Seldom: An Anonymity Network with Selective Deanonymization

ERIC WAGNER, Fraunhofer FKIE, Germany and University of Luxembourg, Luxembourg
ROMAN MATZUTT, Fraunhofer FIT, Germany
MARTIN HENZE, RWTH Aachen University, Germany and Fraunhofer FKIE, Germany

While anonymity networks such as Tor provide invaluable privacy guarantees to society, they also enable all kinds of criminal activities. Consequently, many blameless citizens shy away from protecting their privacy using such technology for fear of being associated with criminals. To grasp the potential for alternative privacy protection for those users, we design Seldom, an anonymity network with integrated selective deanonymization that disincentivizes criminal activity. Seldom enables law enforcement agencies to *selectively* access otherwise anonymized identities of misbehaving users while providing technical guarantees preventing these access rights from being misused. Seldom further ensures *translucency*, as each access request is approved by a trustworthy consortium of impartial entities and eventually disclosed to the public (without interfering with ongoing investigations). To demonstrate Seldom's feasibility and applicability, we base our implementation on Tor, the most widely used anonymity network. Our evaluation indicates minimal latency, processing, and bandwidth overheads compared to Tor; Seldom's main costs stem from storing flow records and encrypted identities. With at most 636 TB of storage required in total to retain the encrypted identifiers of a Tor-sized network for two years, Seldom provides a practical and deployable technical solution to the inherent problem of criminal activities in anonymity networks. As such, Seldom sheds new light on the potentials and limitations when integrating selective deanonymization into anonymity networks.

CCS Concepts: • **Security and privacy → Pseudonymity, anonymity and untraceability**; **Privacy protections**.

Additional Key Words and Phrases: Tor, Exceptional Access, Translucent Ledger, Threshold Encryption

ACM Reference Format:
Eric Wagner, Roman Matzutt, and Martin Henze. 2026. Seldom: An Anonymity Network with Selective Deanonymization. *ACM Trans. Priv. Sec.* 1, 1 (January 2026), 30 pages. https://doi.org/10.1145/3794848

1 Introduction

Tor [20] and other anonymity networks provide invaluable services to, *e.g.*, journalists, whistleblowers, and militaries, who are among its millions of daily users [54]. One may argue that Tor anonymizes too well, as criminals misuse Tor to establish a hotspot of highly illegal activities, such as sharing Child Sexual Abuse Material (CSAM) [39]. Reports showing that over 10 % of Ahmia.fi search sessions seek CSAM [39] and over 50 % of marketplace listings are related to illegal drugs [28] underpin the need for reevaluating the societal downsides of *unconditionally* anonymous Internet use.

Collectively, privacy advocates, such as researchers and some governmental agencies, may accept this fact as a necessary evil for the irreplaceable service Tor provides to society. On an individual

Authors' Contact Information: Eric Wagner, Fraunhofer FKIE, Germany and University of Luxembourg, Luxembourg, eric.wagner@uni.lu; Roman Matzutt, Fraunhofer FIT, Germany, roman.matzutt@fit.fraunhofer.de; Martin Henze, RWTH Aachen University, Germany and Fraunhofer FKIE, Germany, henze@spice.rwth-aachen.de.

ACM 2471-2574/2026/1-ART
https://doi.org/10.1145/3794848

level, however, people do have moral concerns regarding such misuse of Tor. Studies highlight that Tor users are concerned about illegal activities within the network [25, 58]. Moreover, a study suggests that 61 % of US citizens "would like to do more" to improve their privacy, while only 2 % of participants have used anonymity networks such as Tor in the past [46]. Consequently, there exists a potentially large untapped user base for anonymity services. We argue that some of these users, potentially with only a passing understanding of Tor, shy away from it due to its close association with illegal activities they intend to stay away from. After all, each user indirectly supports all Tor activities—by increasing everyone's anonymity [19]—and (exit) relay operators become active targets of criminal investigations.

With this paper, we strive to investigate the potential advantages and limitations of an alternative anonymity network that offers integrated **sel**ective **d**ean**o**ny**m**ization of suspicious users (Seldom). Seldom empowers Law Enforcement Agencies (LEAs) to effectively prosecute crime by investigating suspicious activities within the anonymity network only with acknowledged due cause and under public oversight, while technically limiting these capabilities to prevent abuse of power at the same time. Seldom thus provides those who do not want to be associated with Tor an alternative to adequately protect their anonymity on the Internet. As the provided anonymity grows with the number of users [19], wide acceptance of Seldom can offer strong anonymity even when LEAs can deanonymize selected suspicious identities based on publicly agreed-upon rules.

Seldom realizes its functionality to deanonymize a fraction of communication flows based on a novel *oblivious authentication protocol* that links all outbound traffic to threshold-encrypted user identities. A trustworthy consortium of impartial parties then decides about LEAs' deanonymization requests for individual communication flows. To provide public transparency without impeding ongoing investigations, we propose an immutable but delayed public release of all activities via a *translucent ledger*. Comparing the performance of Seldom to Tor, we measure only an imperceptible latency and processing overhead. Seldom would thus offer a Tor-like experience without the downside of supporting crime when merely using the network. If significantly outgrowing Tor, Seldom could even reduce network latency and thus address one of the major drawbacks of onion routing.

Contributions. To provide a thorough understanding of the potential *and* implications of selective deanonymization in anonymity networks, we make the following contributions:

- We devise an oblivious authentication protocol that links each outgoing communication flow to a unique threshold-encrypted identity of the anonymized client, and we integrate this protocol into our anonymity network Seldom.
- We design a translucent deanonymization process that only reveals the client identity to the requesting LEA.
- Our evaluation shows that Seldom has an imperceptible performance impact on the users' experience and imposes only well-manageable data storage requirements on centralized databases.
- Finally, we discuss the challenges, limitations, and implications of deploying an anonymity network with exceptional deanonymization capabilities.

2 Acceptable Exceptional Access

Providing data access to an LEA in legitimate circumstances is often referred to as *exceptional access*. Any system providing exceptional access naturally faces public scrutiny due to its potential for abuse. It is thus vital to technologically protect individual privacy against LEAs by enforcing regulation, transparency, and accountability. However, LEAs need to be able to identify unfolding crimes and act accordingly. To combine these conflicting requirements, we lay out the prerequisites to successfully strike a balance in the safety-privacy dilemma. Overall, we identify the following aspects to be essential for a widely acceptable form of exceptional access:

Nobody But Us. Exceptional access must be restricted to authorized entities and must especially not be accessible to criminals or privacy-invasive third parties.

Translucency. LEAs should operate transparently and inform the public of targeted surveillance activities. However, timely access to such data also helps criminals evade surveillance. Hence, *translucency*, which allows for critical information to remain *temporarily* classified, is needed.

Accountability. If an LEA abuses its given privileges, its exceptional access privileges must be revocable in a way that is transparent to the system's users.

Ethical Decision-Making. Individual LEAs might target different persons of interest. For instance, government-critical journalists or whistleblowers might be subjected to surveillance without posing a harm for public safety [52]. Any decision-making process on how to grant an LEA exceptional access must thus reflect this ethical issue and allow for denying access on these grounds.

Timely and Flexible Access. LEAs must be able to swiftly access data in case of imminent danger.

Limited User Overhead. When using a system offering exceptional access, citizens should not be unnecessarily affected by overhead stemming from enabling this access.

Overall, these essential prerequisites ought to ensure that surveillance of digital communication (i) is limited to exceptional cases, (ii) is effective in prosecuting crime on the network and thereby disincentivizing it, and (iii) generates wide societal acceptance by strengthening public safety.

3 Related Work

In the following, we give an overview of previous approaches to enabling exceptional access. We notice that related work has evolved and considerably broadened its scope since the initial proposals during the 1990s crypto wars.

Partial Key Escrow. Early works focused on *(partial) key escrow*, which involves the government maintaining a global store of (parts of) all used encryption keys [10, 18]. Hence, the government would be able to retrieve keys and brute-force partial keys if needed. However, such systems require strong trust in key holders; even the protection of partial keys decays over time as computation power rises [5]. Similarly, the idea of obliviously escrowing encryption keys at random peers [13] was quickly abandoned due to overhead and trust concerns. Recent work looks at the use of distributed ledgers as trusted parties for key escrow [23, 26]. These systems, however, require active adoption from users, while it remains unclear how users with criminal intent would be motivated to use such key escrow systems.

Translucent Cryptography. Orthogonally to key escrow schemes, *translucent cryptography* [11] is based on a variant of oblivious transfers [44] called Non-Interactive Fractional Oblivious Transfers (NFOTs). In this scheme, any message also holds an NFOT, which allows an LEA to decrypt wiretapped messages with a predefined probability. However, at least one communication partner must remain honest in this scheme. Further, evaluating a wiretapped NFOT leaves no evidence, which prevents transparency or accountability. Finally, a purely probabilistic per-message scheme does not hold up with today's communication patterns and surveillance capabilities. Namely, being able to decrypt only a tiny fraction of messages still allows one to peek into a large fraction of conversations but prohibits gaining all relevant information from any rightfully investigated individual.

Crypto Crumple Zones. As a continuation of the idea of partial key escrow, *crypto crumple zones* [60] are designed to augment the generation of ephemeral keys with exceptional access in mind. At their core, they enable powerful entities to invest computational resources to retrieve full keys akin to Bitcoin's proof of work. Overall, this approach has improved rate-limiting guarantees compared to partial key escrow, but it neither restricts access to LEAs nor provides transparency.

Lawful Device Access. Another proposal considers the design of special storage hardware that can be decrypted based on prolonged physical access, *e.g.*, hours to days [48]. LEAs would then have to prove continual physical access before being able to read the storage. This way, transparency, accountability, and rate-limited access are established by requiring traditional seizures of hardware. Namely, this approach could have provided a legitimate framework for accessing the device at the center of the controversy around the San Bernardino shooting [55]. However, the roll-out of such technology would be slow due to its reliance on the wide adoption of specialized storage hardware, and it does not consider encrypted containers stored on such hardware.

Privacy-preserving Content Scanning. Orthogonally, client-side scanning wants to identify illegal material such as CSAM on clients' devices. In this vein, *privacy-preserving content scanning* attempts to realize client-side scanning while respecting the user's privacy [9, 36]. However, current approaches suffer from limitations, *e.g.*, the uncertainty of whether detection remains restricted to harmful material and will not be abused for censorship or surveillance of political opponents [4]. Indeed, recent analysis shows how the underlying perceptual hashing algorithms to compare images are susceptible to adversarial images, *i.e.*, images in the database that look like CSAM to humans but have hashes that collide with other secretly tracked content [29, 31, 32].

Deanonymization in Onion Routing. A wide range of approaches strives to enable deanonymizations of activities in anonymity networks through traffic analysis, such as website fingerprinting on the connection between client and entry relay, *e.g.*, to derive which website a client likely visited. To this end, different approaches claim to identify visited websites based on traditional machine learning [14, 22, 27, 41, 42, 57], deep learning [12, 45, 47, 49, 51], or generative artificial intelligence [40]. However, besides various issues with website fingerprinting itself, such as unrealistic or oversimplified assumptions [35] or limited scalability [15, 41], these approaches neither provide correctness guarantees nor allow deanonymizing clients when observing activity related to the server side, *e.g.*, who accessed a certain website. To the best of our knowledge, BackRef [8] is the only proposal to provide such functionality by enabling relays to iteratively prove that they are not the origin of suspicious traffic. BackRef [8], however, creates significant storage overhead for all relays and requires them to prove their innocence, which becomes problematic if relay operators cannot provide evidence, *e.g.*, due to data loss. Finally, BackRef [8] is neither transparent in terms of scale nor purpose of requests.

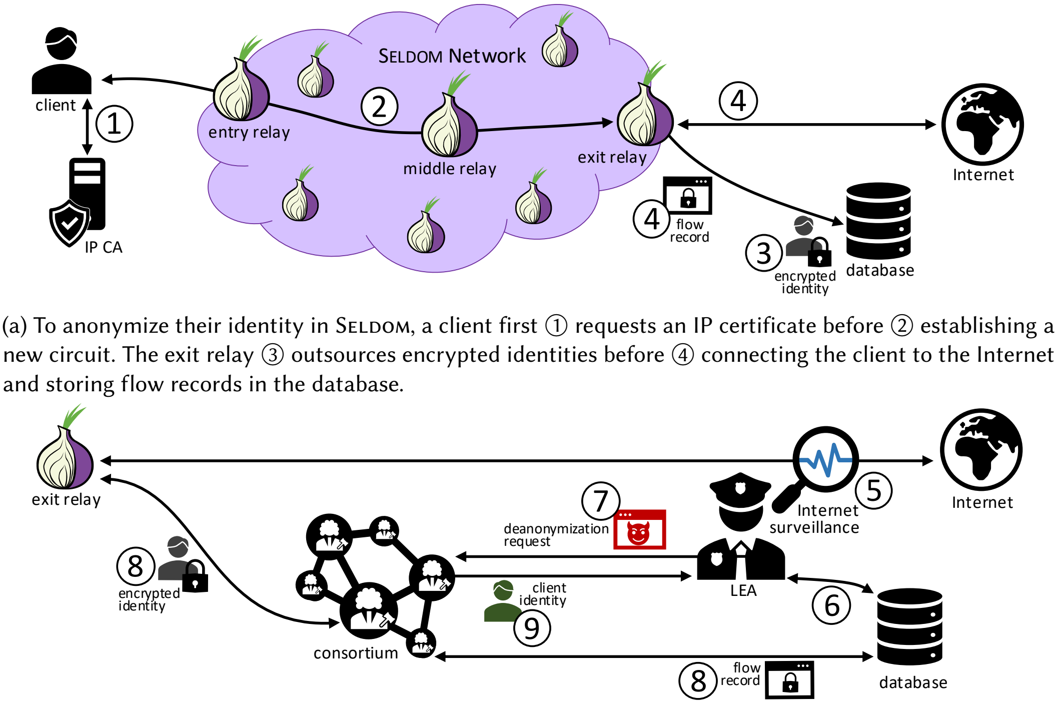
Past work on exceptional access thus aims to provide LEAs with tools to intrude on an individual's privacy to ultimately protect public safety. In contrast, we are the first to propose using exceptional access capabilities to increase privacy by providing a new service to an untapped user base currently browsing the Internet without anonymity.

4 High-Level Overview of Seldom

In this section, we give a high-level overview of Seldom's design and the assumed threat model before we describe Seldom's technical details in greater depth in the following Sections 5 and 6.

4.1 Design Overview

Seldom is an anonymity network based on onion routing that is extended with exceptional access capabilities according to our prerequisites formulated in Section 2. We design Seldom by extending Tor's design. Figure 1 provides a high-level overview of Seldom's architecture. Like Tor, the Seldom network uses an ensemble of directory authorities to manage the relays in the network and also supports bridge relays that do not need to be listed publicly.



(a) To anonymize their identity in Seldom, a client first ① requests an IP certificate before ② establishing a new circuit. The exit relay ③ outsources encrypted identities before ④ connecting the client to the Internet and storing flow records in the database.

(b) LEAs monitor the Internet and, upon ⑤ identifying suspicious traffic, an LEA can ⑥ look up the corresponding flow record and ⑦ request a deanonymization. Then, the consortium ⑧ obtains the required information and votes to ⑨ reveal the client's identity to the LEA.

Fig. 1. Overview of Seldom's anonymization (Figure 1a) and deanonymization (Figure 1b) procedures.

Clients use Seldom just like Tor, but Seldom securely stores data related to the clients' identities in a database under shared control of a *consortium* of impartial parties. This process allows an LEA observing suspicious Internet traffic leaving the Seldom network to file a substantiated *deanonymization request* with the consortium. It is the consortium that then *jointly* decides whether it should reveal the client's identity to the requesting LEA, with the knowledge that this decision and the attached justification will be released to the public in the future.

The anonymization process of Seldom is illustrated in Figure 1a. First, the client must ① obtain a certificate based on their current IP address as a means for potential identification. To then ② establish a circuit, the client propagates the IP certificate to the exit relay such that all relays can *obliviously validate* its authenticity. At the end of the circuit establishment, the exit relay ③ stores the encrypted client identity in a, for now, government-sponsored central database. These identities are only threshold-decryptable by the consortium. Afterward, the circuit is established successfully, and the client can ④ access the Internet. Simultaneously, the exit relay ④ stores symmetrically encrypted flow records with searchable metadata for each connection in the database.

We assume that LEAs are monitoring the Internet. If they ⑤ identify suspicious activity coming from the Seldom network, they proceed as shown in Figure 1b. They first ⑥ query the database for the corresponding flow record and ⑦ file a duly substantiated deanonymization request with the consortium. The consortium logs the request and potential authorization through a smart contract on a translucent ledger. After receiving proof of a logged and approved deanonymization request, the exit relay ⑧ shares the corresponding circuit's *encrypted* identity with the consortium. Then, the consortium ⑨ decrypts the threshold-encrypted identity and reveals the identity only to the requesting LEA.

Immutably logging *all* requests and decisions is crucial for maintaining transparency toward the users. However, Seldom has to ensure that criminals under investigation do not gain an advantage via this channel. To address this issue, we introduce a *translucent ledger* (*cf.* Section 6.2) that is maintained by the consortium and allows for a *delayed* full disclosure while providing a general overview of ongoing activities in real time. With this form of public oversight, citizens can notice *patterns* of potential power abuses within Seldom and abandon the network in cases of government overreach. In case of a successful deanonymization, users also only lose exactly the anonymity gained through the use of the network, *i.e.*, their browsing is as safe or unsafe as if no anonymity network was ever used.

4.2 Threat Model and Trust Assumptions

The threat model for Seldom and the trust assumptions toward the different entities involved are guided by the core goal of Seldom:

Mission Statement of Seldom
Seldom's purpose is to provide a crime-free anonymization network to encourage more users to use such networks. To deter malicious activities, the anonymity provided by Seldom is revocable in justified cases, *e.g.*, if public safety is at risk. This revocation power is protected against abuse from any involved entity by relying on strong cryptographic primitives that only enable policy-based and translucent deanonymization. Further, deanonymization is non-invasive in that it is strictly limited to the service provided by Seldom. It is an explicit goal that a client establishing a TLS connection through Seldom can be deanonymized while the transmitted traffic remains encrypted. If backdoored encryption should ever be desired, it falls outside of Seldom's scope.

In the following, we first introduce our threat model before discussing the trust assumptions toward the different entities in the network.

4.2.1 Threat Model. We build on Tor's threat model [20] and extend it to capture the deanonymization capabilities introduced by Seldom. Hence, we assume that there exists no global passive adversary who is capable of matching all incoming and outgoing traffic at every relay. Instead, we consider adversaries that can observe a fraction of all Internet traffic and possibly operate a portion of relays. This direct passive observation is one of the ways for LEAs to identify critical traffic on the Internet that may require deanonymization. Alternatively, LEAs can request (usually through a warrant) traffic metadata, *e.g.*, IP addresses and timestamps, from Internet Service Providers (ISPs) or website operators. While a single person or entity can take on multiple roles in the network, it cannot take over a significant share of *e.g.*, relay operators. An LEA can, for example, take on the role as client and relay operator in the network at the same time. Further, a government, as one entity, could even act as an LEA and a consortium member. With these capabilities, the goals of an adversary can be twofold. They can either evade or prevent deanonymizations to hide suspicious activities, frame an innocent client, or simply disrupt the network to hinder adoption. On the other hand, an adversary can also attack Seldom to conduct unauthorized deanonymizations.

4.2.2 Trust Assumptions about the Different Roles. In the following, we list the entities involved in the operation of SELDOM and which trust assumptions are made about them.

Client. Anyone can operate one or multiple SELDOM clients, but a client only has access to information about itself.

IP Certificate Authority. IP CAs are independent entities that have to be designated when SELDOM is set up. Deanonymization results of SELDOM can only be trusted by the requesting LEA to the degree they trust the CA.

SELDOM Relay. Any entity can operate one or multiple relays. As for Tor, clients need to trust that no single entity controls a large fraction of all relays, as proper anonymization is not guaranteed otherwise. Relays are expected to follow the SELDOM protocol and will be excluded from the network if misbehavior is detected.

SELDOM Exit Relay. Exit relays are relays that are exposed to the Internet and thus the initial suspicion points for LEAs. In Tor, this risk deters many relay operators. On the other hand, SELDOM, with its automated deanonymization process, can better protect exit relays. In turn, exit relays are expected to be available for deanonymizations.

Directory Authority. SELDOM, like Tor, has a set of highly trusted directory authorities that manage the list of relays.

Consortium. The members of the consortium must be carefully chosen, and only a small number of changes over time are expected. It must collectively be trusted to act honestly. For SELDOM to function properly, the consortium thus has to be trusted by clients, relays, and LEAs alike, while each individual may distrust different members within the consortium.

Database Operator. We assume that all entities trust the database operator(s). The involved risk is that they can see the metadata of all outgoing connections through flow records. As detailed later in Section 5.3 (Figure 3), database operators do not see which flows are associated with the same circuit. In practice, the database could be operated by the directory authorities, by governments to redistribute costs if the privacy implications are accepted, or distributed among multiple entities.

5 Oblivious Authentication

SELDOM is built upon an *oblivious authentication protocol* integrated into onion routing. This protocol authenticates a client to an exit relay without revealing its identity. Clients have to obtain *temporal IP certificates* (Section 5.1) before being able to establish a circuit (Section 5.2) so that exit relays are convinced that any outgoing traffic can be irrefutably linked to the client (Section 5.3). In Appendix A, we detail how we integrate this protocol with the existing Tor handshake.

5.1 Temporal IP Certificates

SELDOM needs a way to reliably represent an individual's identity if deanonymization is required later on. Regular, CA-issued certificates are not applicable to SELDOM, as (i) obtaining them is costly and not always possible [8] and (ii) they would share excessive personal information with entry relays. Instead, we propose to identify users via *temporal IP certificates*, which link an ephemeral public key[1] of a client to a timestamped IP address.[2] While not necessarily allowing recovery of a criminal's true identity directly, deanonymizable temporal IP certificates effectively cancel

[1] SELDOM uses the ed25519 signature scheme for its speed and small signatures.

[2] Since designing and implementing SELDOM, Let's Encrypt announced the rollout of IP Address Certificate [24], which would satisfy SELDOM's requirements.

out the added privacy provided by the anonymity network only for LEAs bringing forward good reasons for further investigation. Only the entry relay obtains the temporal IP certificate in the clear for validation purposes. If the deanonymization reveals an invalid certificate, this serves as an indication for LEAs that the client and entry relay colluded, pointing to the entry relay for further investigation.

Temporal IP certificates are issued in an automated manner by one or more CAs that are trusted by LEAs. The client generates a public-private key pair for any new Internet connection (*e.g.*, ISP-triggered reassignment of IPv4 addresses) and sends the public key to a CA for certification. In the simplest case, the CA extracts the client's IP address from the request and creates the temporal IP certificate from that IP address, the public key, and the current timestamp. We further discuss adaptations to this process for users relying on bridge relays, *e.g.*, for censorship evasion, in Appendix B.

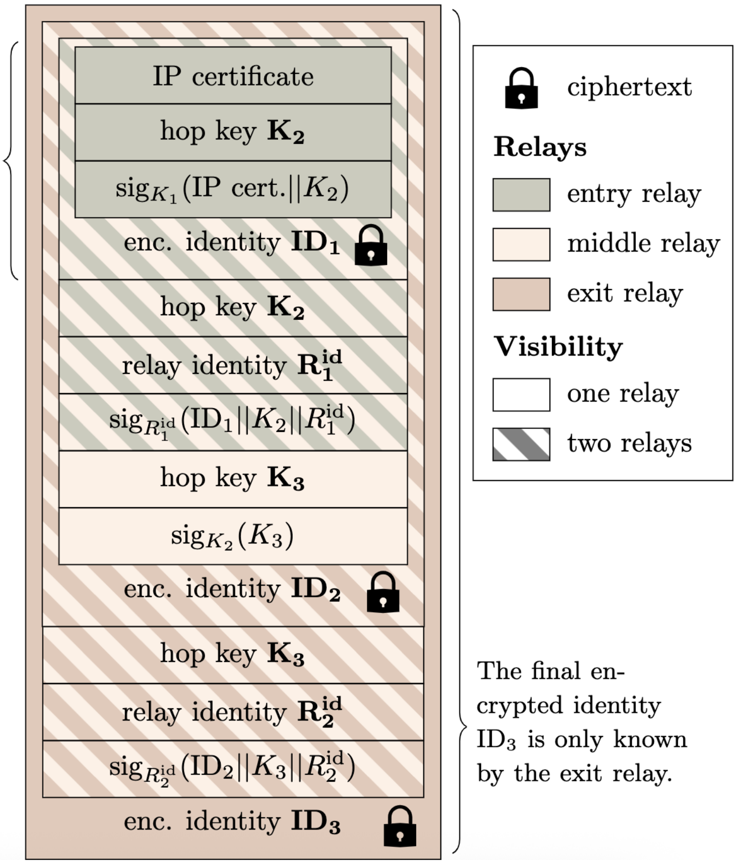
5.2 Oblivious Authentication Protocol

We now present our oblivious authentication protocol. This protocol distinguishes three cases, detailed in the following: authenticating the client to their entry relay (Section 5.2.1), propagating authentication via middle relays (Section 5.2.2), and finalizing authentication at the exit relay (Section 5.2.3). Figure 2 illustrates the final derived encrypted identity for a circuit consisting of three relays R_1, R_2, and R_3.

5.2.1 Authentication to the Entry Relay. The purpose of our authentication protocol is to bind a temporal IP certificate to the hops within a circuit in a way that relays can only obtain information related to their direct predecessor. We hence refer to the ed25519 public keys to authenticate individual hops as *hop keys* K_n.

After obtaining a valid temporal IP certificate (*cf.* Section 5.1), the client confidentially sends that certificate to the entry relay R_1, which extracts the client's first hop key K_1 from the certificate. The entry relay validates K_1 by verifying the IP certificate's signature by the CA and ensuring that the client uses the certified IP address. Failures to validate the IP address will surface during deanonymizations; hence, R_1 is incentivized to perform this check honestly. Next, the client generates a second hop key pair (with the associated public key K_2) to prepare the authentication forwarding between relays R_1 and R_2. By signing K_2 using the validated K_1, R_1 can in turn validate the authenticity of K_2. Finally, R_1 threshold-encrypts the client's IP certificate, K_2, and the signature under K_1 using a CCA-secure Threshold Public Key Encryption (TPKE) scheme. Thus, only the consortium can access the encrypted data from now on. We refer to the resulting ciphertext as the client's *encrypted identity* ID_1. Our concrete implementation relies on a hybrid threshold-encryption scheme based on Shoup's threshold RSA signatures [50] and AES in GCM mode. We published the scheme as open source under the name *thRSAhold* and present it in more detail in Appendix C.

5.2.2 Oblivious Authentication Forwarding. Assuming that an obliviously authenticated partial circuit has been established up to relay R_n, we now describe how to extend the authentication via another hop to R_{n+1}. On a high level, R_n forwards its view on the state of current authentication, ID_n and K_{n+1} as well as its own relay identity key R_n^{id} to R_{n+1}. R_n^{id} is usually an ed25519 public key and published via the relay's descriptor [20]. Other relays, such as unlisted bridge relays, can be identified pseudonymously via additional, SELDOM-specific certificates retrieved from a bridge authority. Only in cases where LEAs must identify a relay operator, bridge authorities would reveal the certificate owner. R_n further signs the transmitted information with its identity key, yielding the signature $\mathrm{sig}_{R_n^{\mathrm{id}}}(\mathrm{ID}_n || K_{n+1} || R_n^{\mathrm{id}})$. This signature is transferred to, and verified by, R_{n+1}. Verifying the identity of R_n reassures R_{n+1} that LEAs can discover R_n if necessary, and thus any deanonymization process can be properly delegated to R_n.


IP certificate
hop key $\mathbf{K_2}$
$\mathrm{sig}_{K_1}(\text{IP cert.}||K_2)$
enc. identity $\mathbf{ID_1}$
hop key $\mathbf{K_2}$
relay identity $\mathbf{R_1^{id}}$
$\mathrm{sig}_{R_1^{\mathrm{id}}}(\mathrm{ID}_1||K_2||R_1^{\mathrm{id}})$
hop key $\mathbf{K_3}$
$\mathrm{sig}_{K_2}(K_3)$
enc. identity $\mathbf{ID_2}$
hop key $\mathbf{K_3}$
relay identity $\mathbf{R_2^{id}}$
$\mathrm{sig}_{R_2^{\mathrm{id}}}(\mathrm{ID}_2||K_3||R_2^{\mathrm{id}})$
enc. identity $\mathbf{ID_3}$
ciphertext
Relays
entry relay
middle relay
exit relay
Visibility
one relay
two relays

IP certificate, hop key K_2, and the signature $\mathrm{sig}_{K_1}(\text{IP cert.}||K_2)$ are only visible to the entry relay. This information is then threshold-encrypted to create the encrypted identity ID_1, which is revealed to the entry relay and the middle relay.

The final encrypted identity ID_3 is only known by the exit relay.

Fig. 2. SELDOM's oblivious authentication protocol hides the client's identity behind multiple layers of threshold encryption (entries marked with a padlock), which can only be decrypted through collaboration of the consortium. The encrypted identity of a 3-hop circuit is approximately 500 B long.

Following this step, the client establishes a new hop key K_{n+2} between itself and R_{n+1} without revealing it to R_n. To link this new hop key K_{n+2} to the circuit, the client signs this new hop key with the old hop key K_{n+1} and sends the new key and this signature, *i.e.*, $\mathrm{sig}_{K_{n+1}}(K_{n+2})$, to R_{n+1}. This signature is the only link between two hop keys. To prevent collusion attacks between relays, it is, therefore, crucial that this signature is only revealed to the relay also knowing the corresponding keys, *i.e.*, R_{n+1}. After validating the signature, R_{n+1} computes ID_{n+1} by threshold-encrypting all now-established data.

These steps can be repeated to establish longer circuits. Only the final extension step, to the exit relay, has to be treated differently, as we detail next.

5.2.3 Oblivious Authentication to the Exit Relay. In the last step, the exit relay obliviously authenticates the client. Without loss of generality, we now assume that a circuit has length three, *i.e.*, the exit relay is R_3. Then, R_3 receives the following from R_2: the current encrypted identity ID_2, the hop key K_3, the middle relay's identity R_2^{id}, and its signature $\mathrm{sig}_{R_2^{\mathrm{id}}}(\mathrm{ID}_2||K_3||R_2^{\mathrm{id}})$. As R_3 is the last hop within the circuit, it only has to threshold-encrypt the received data one last time to yield the final encrypted identity ID_3. In case of future deanonymization, a client-circuit binding is identified

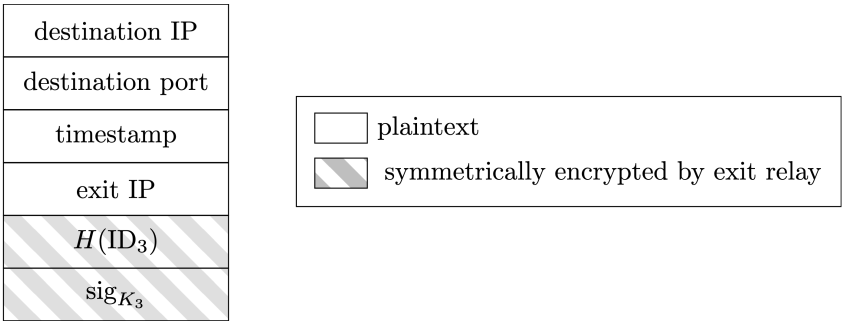


Fig. 3. Flow records are searchable by LEAs. The link to the encrypted identity is however encrypted by the exit relay to prevent the matching of multiple flows to the same circuit. Storing a flow ultimately produces 134 B of data.

by the last hop key K_3, which we also call the *client identification key*, and a digest $H(\mathrm{ID}_3)$ of the encrypted identity (SELDOM uses SHA-3 as its cryptographic hash algorithm H).

However, storing available encrypted identities and associated metadata of outgoing traffic flows quickly becomes expensive for the exit relay. Thus, SELDOM outsources this data to external databases. The threshold encryption of ID_3 prevents unlawful access by individual parties and outsourcing this information unburdens exit relays and ensures data availability to LEAs at the same time. Therefore, exit relays are expected to upload their data periodically in batches to mitigate traffic analysis attacks and improve bandwidth usage. The uploaded data is retained only for a predetermined period, *e.g.*, for one year. Afterward, the consortium and exit relays refuse to support deanonymizations.

5.3 Linking Internet Traffic to Circuits

Following an established circuit, the exit relay's outgoing traffic must be indisputably linked to a client. Here, SELDOM takes a flow-based approach. Therefore, whenever the client makes the exit relay establish a new flow, it signs this *flow record* and the database key to its encrypted identity $H(\mathrm{ID}_3)$ with the client identification key. The flow record, illustrated in Figure 3, consists of the destination's IP address, the targeted port, the exit relay's IP address, and a timestamp. Only the client can generate this signature, and the corresponding public key is stored in the encrypted identity ID_3. In case of a deanonymization request, the flow record, this signature, and the encrypted identity shift the blame toward the client.

In practice, when opening a new connection, the client creates the signature over the flow record and includes it in the first transmitted cell. The exit relay verifies this signature and then establishes the connection. Afterward, the client can communicate over this flow without additional overhead through the circuit. To outsource the storage of this flow record, the exit relay symmetrically encrypts the signature and the digest $H(\mathrm{ID}_3)$. The resulting data layout is displayed in Figure 3. We made the design choice to symmetrically encrypt this sensitive data and rely on the exit relay to decrypt it in case of a deanonymization request. Even if the exit relay were offline, it would be traceable through the consensus or IP address. With another threshold encryption, we could avoid this involvement of the exit relay, but this would reduce oversight. Most importantly, the active involvement of exit relays ensures the correctness of the statistics and disclosure of deanonymization requests even for a dishonest consortium.

In theory, two different clients could open a connection through the same exit relay in close succession. In this case, it would be hard to match observed Internet traffic to a specific flow record.

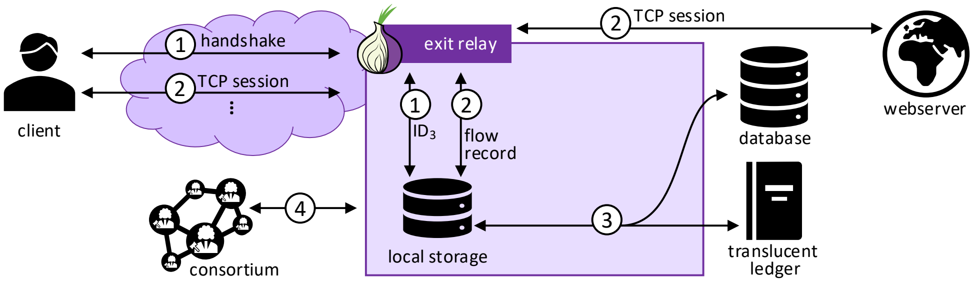


Fig. 4. During the establishment of a circuit, a Seldom exit relay first ① performs the Seldom handshake and stores the encrypted identity of the client. Then, ② one or multiple connections are opened by the client and the flow records stored by the exit relay. The exit relay ③ periodically publishes its locally stored data to the database and ④ remains available for possible collaboration with the consortium during deanonymizations.

This challenge stems from the fact that the client cannot reliably know the port used by an exit relay; hence, the client cannot include this information in its signature. In practice, we expect such similar flow records to rarely and mostly happen to widely known and benign websites, *e.g.*, news sources. In the worst case, LEAs would have to investigate two or more circuits, which could come with higher scrutiny by the consortium deciding on the deanonymization request.

An additional challenge comes from the fact that no single database operator is trusted by everybody. It would, however, put additional strain on the exit relays if they have to migrate their database entries to multiple deployed databases. We offload these synchronization costs to the operators while simultaneously benefiting from the better connectivity among them. However, if the operators synchronize the databases themselves, one operator must be sure that they know about all data distributed by the exit relays. Hence, we implemented a smart contract (deployed on the translucent ledger introduced in Section 6.2) through which exit relays inform about the digest of migrated database entries. Each database operator thus learns which data other instances should distribute to them. All in all, Seldom thus allows the distributed storage of the necessary data for deanonymizations without overburdening the exit relays or other volunteers with additional computations or costs.

5.4 A Seldom Circuit from the Perspective of an Exit Relay

We now present the anonymization procedure of Seldom from the perspective of an exit relay to summarize the system. As shown in Figure 4, the exit relay's involvement ① begins with the last part of the handshake to establish a new circuit for the client. It validates the client identification data, computes the encrypted identity ID_3, and assigns the extracted client identification key to the circuit as described in Section 5.2.3. Then, the exit relay stores ID_3 locally.

Next, the client ② establishes TCP sessions through the circuit. While establishing the connection, the client submits a flow record signed by its identification key to the exit relay. The exit relay extracts this flow record and verifies it. Then, the flow record is stored locally by the exit relay, which then establishes the connection. The client can now send data like in Tor.

Finally, the exit relay ③ regularly publishes its locally stored encrypted identities and flow records to the database. To ensure consistency in the database, this also involves writing the hash of each update to a smart contract on the translucent ledger. Moreover, the exit relay ④ remains available for potential deanonymization requests by the consortium.

6 Deanonymization Process

In the following, we look at SELDOM's deanonymization process, which is illustrated by Figure 5. SELDOM relies on a consortium of trusted parties to approve and execute deanonymizations. While this paper focuses on the technical aspects, we nevertheless give some considerations about the potential composition of such a consortium in Section 6.1. To ensure public oversight, we introduce *translucent ledgers* in Section 6.2 to manage this consortium. The use of a translucent ledger enables the public to see live statistics about attempted and successful deanonymization requests, ensures the delayed information disclosure, and allows the consortium to grant selective data access to prove an approved request to an exit relay. Afterward, we discuss the deanonymization process in Section 6.3. Finally, we see how SELDOM probes relays to preemptively detect rogue relays without compromising anyone's privacy in Section 6.4.

6.1 Assembling a Trusted Consortium

We identify three roles that should be represented in SELDOM's consortium: *governments*, *privacy advocates*, and *privacy-focused corporations*. First, we need representatives from the judicial branch of participating governments. These entities have the knowledge to assess deanonymization requests based on the potential benefit to public safety and their legality. Secondly, the consortium needs to incorporate parties that are known to strongly advocate for people's privacy. These parties are essential to generate trust in the system. They can block excessive requests or such requests that clearly do not benefit public safety, *e.g.*, if the deanonymization of a whistleblower would be requested. This group could include the operators of Tor's directory authorities and other well-known privacy advocates. Thirdly, we see a need to include corporations that are known to prevent governments from excessively accessing encrypted data, *e.g.*, *Mozilla* and *Apple*. These are powerful entities relative to typical privacy advocates and thus are better equipped to oppose political pressure.

Besides the type of representatives in the consortium, its size must also be determined, as well as the threshold to approve requests. This threshold can be lower than the threshold set for the threshold decryption, assuming that non-agreeing entities still cooperate in the decryption once a vote has passed. Moreover, the interval between the renewing of threshold-encryption keys must be decided on, such that honest consortium members can ensure that no data can be accessed beyond its intended retention period. Overall, the consortium should be relatively static, and changes should be planned in advance to happen with scheduled key updates, but they also require updating the smart contract. Considerations regarding the consortium's size and the voting procedure (*e.g.*, simple majority, majority within each role, etc.) remain important aspects to SELDOM's deployment but go beyond the scope of this work.

6.2 Translucent Ledgers

We propose the concept of *translucent ledgers* for the coordination among the consortium while ensuring public oversight. Generally, a translucent ledger is a private blockchain where a new block (deanonymization requests in SELDOM's case) is accepted by the consensus of the consortium members and is cryptographically linked to its predecessor. For SELDOM, we use an Ethereum [59] blockchain with the Proof of Authority (PoA) consensus mechanism, which requires minimal modifications to be used as the translucent ledger. What makes a ledger translucent is that the consortium grants access to selected data for specific clients and the public. The three pillars of translucency are (i) delayed disclosure, (ii) peeks into real-time data, and (iii) real-time statistics. These three pillars allow keeping ledger data secret as much as necessary while providing guarantees of the correct execution of processes to the public.

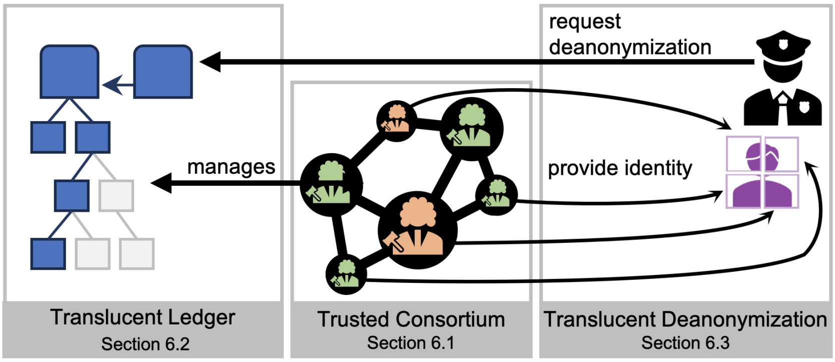


Fig. 5. SELDOM relies on a trusted consortium. During operation, LEAs can make deanonymization requests on a translucent ledger, which will be voted on by the consortium and, if accepted, collaboratively executed by the consortium.

Delayed Disclosure. The first building block of translucent ledgers is the notion of delayed disclosure, *i.e.*, the delayed release of all stored information to the public such that they can be verified by everyone. For SELDOM, this delayed release is based on a fixed delay, as already exists for the declassification of secret documents by democratic governments. Still, the consortium may unanimously agree on delaying the release of specific data, *e.g.*, if a case is still ongoing. Here, a single honest entity in the consortium suffices to enforce the timely data disclosure, while the blockchain properties ensure data integrity. Importantly, this step only reveals which deanonymizations have been requested (including a reasoning) and which of those succeeded, while the client's identity is never written to the ledger and thus also not revealed to the public.

Peeks into Real-time Data. While disclosure to the public is delayed, interactions with the ledger must happen in real time. Especially exit relays must be able to read the ledger when they are authorized to do so, *i.e.*, to verify the approval of a deanonymization request. Yet, no additional information should leak about the ledger's current state to unauthorized entities. To enable such fine-grained interactions, the operators of translucent ledgers can generate verifiable claims of the current state of parts of the ledger. We build our translucent ledger based on the Ethereum blockchain, such that we can use Merkle proofs as proposed in EIP1186 [34] to reveal the current state of each smart contract variable individually by linking it to the public header. These Merkle proofs link the revealed data to the already distributed and immutable block headers known to the public. Thus, authorized outsiders are enabled to peek into specific parts of the ledger's state.

Real-Time Statistics. Delaying information disclosure for a significant time can make a process appear opaque, making users more skeptical of its trustworthiness. To counteract this possibility, translucent ledgers can distribute real-time statistics about what is being stored and processed on the ledger. To this end, we use the Bloom filters in Ethereum [59] that usually are used to let clients quickly search for past events. Events are introduced by smart contract programmers and are triggered when certain conditions are met during code execution. Upon being fired, these events write logs to the ledger. Additionally, the specific event that was triggered and its arguments are inserted into the Bloom filter of the current block header. Thus, by strategically specifying these events, a smart contract developer can control what real-time statistics are leaked into the block header.

Translucent ledgers are thus a strong building block for the management of exceptional access that is neither opaque nor completely transparent to the public.

6.3 Translucent Selective Deanonymizations

We now discuss how we use a translucent ledger to create a translucent deanonymization process. After identifying suspicious SELDOM-anonymized traffic, an LEA requests a deanonymization from the consortium. Therefore, the consortium first assesses this request through a translucent voting. If the request is approved, the consortium members, with the help of the exit relay, jointly remove the layers of threshold encryption from the client's encrypted identity. The final decryption shares are then only shared with the requesting LEA, such that the identity of the suspect is not even revealed to the consortium.

6.3.1 Translucent Voting. We assume that LEAs monitor the Internet or request logs from websites to identify suspicious traffic that has been anonymized by SELDOM when deanonymization is warranted to uphold public safety. As a first step, the LEA then searches the database for the flow record (*cf.* Figure 3) that matches the observed traffic. For now, consider that this database is fully accessible to LEAs, such that LEAs have a list of all communication flows exiting SELDOM. To further strengthen privacy, this database could be based on searchable encryption [16] (and rate-limited through cryptographic puzzles or the database being operated by the consortium) where each exit relay logs to a different index with an encryption key only known to the relay. This index could still be searched by LEAs to match concrete flow records, while no information about all traffic leaks.

After identifying the appropriate flow record, the LEA can request a deanonymization (and supply evidence) from the consortium by calling the corresponding function of the smart contract deployed on the translucent blockchain. This deanonymization request contains one suspicious flow record (referenced by its hash) and justification for the consortium to approve the request, which are both written to the translucent ledger. What would be accepted as valid justification must be agreed upon by the consortium, but it could involve a warrant or the evidence collected by the LEA. Appendix D shows an exemplary smart contract for a consortium with five members and basic majority voting. This function call triggers an event in the smart contract which reveals that a deanonymization request occurred immediately to the public, while the suspicious flow record and justification will only later be revealed to the public.

Then, each consortium member can call the smart contract to vote in favor of the deanonymization. If enough positive votes are collected, the request is accepted; this approval triggers an on-chain event visible to all SELDOM users, *i.e.*, the public. The delayed disclosure then ensures that the individual votes of each consortium member will eventually be fully published. This event also starts the deanonymization procedure by the consortium. Following a successful vote, however, only the exit relay can decrypt the key in the flow record linking it to the encrypted identity.

6.3.2 Linking Flows to Encrypted Identities. The consortium needs to request the exit relay to link the identified flow record to an encrypted identity. Therefore, the exit relay needs to be informed about the flow record and convinced that the request was approved and logged by the consortium. The consortium provides this proof through an EIP1186 Merkle proof [34], which proves the deanonymization request (without justification) and voting record to the exit relay by revealing the hash chain that connects the smart contract state variables to the already public blockchain header. This communication between the consortium and the exit relay is confidential and realized over a TLS channel between one consortium member and the exit relay.

Thus, only the exit relay is informed about this request, as SELDOM would otherwise extensively interfere with ongoing investigations. After verifying the request, the exit relay is thus sure that the consortium agreed on the necessity of this deanonymization and that this will eventually be disclosed. Then, the exit relay decrypts the digest $H(\mathrm{ID}_3)$, which is the database key for the encrypted client identity ID_3, as well as the flow record signature by the client identification key. The exit relay then transfers this data to the consortium.

The consortium can now retrieve the associated encrypted identity ID_3 (*cf.* Figure 2) and remove the first threshold-encryption layer to verify the honesty of the exit relay. For this verification, several requirements must be fulfilled. First, the data pointed to by the database key has to be retrievable. Second, the signature of the flow record must be verifiable with the client identification key K_3 revealed by the decryption. This signature ensures that the outer encrypted identity is indeed the origin of the investigated communication, as only the client knows the corresponding private key and would not sign flow records of unknown connections. Third, the consortium has to check that the indicated predecessor relay has a known identity R_2^{id}. Knowing the middle relay's identity means that either a valid certificate is provided or, more commonly, the relay's identity is stored in Seldom's consensus. Finally, the consortium ensures that this middle relay did indeed vouch for the correctness of the next layer's encrypted identity by verifying the signature $\mathrm{sig}_{R_2^{\mathrm{id}}}\left(\mathrm{ID}_2 || K_3 || R_2^{\mathrm{id}}\right)$.

Together, these verification steps convince the consortium that the exit relay received the claimed inner encrypted identity ID_2 from the indicated predecessor relay and that the exit relay acted according to the protocol. If any of the preceding verifications fail, this would indicate a malicious exit relay and thus likely collusion with the client.

6.3.3 Extracting the Traffic's Origin. After verifying the exit relay's honesty, the consortium proceeds to similarly verify the other relays' honesty, ultimately leading to the identification of the client's identity. Therefore, the consortium first jointly decrypts the inner encrypted identity ID_2. Then, the consortium verifies the honesty of the middle relay with the same process as for the exit relay. Additionally, the consortium verifies that the hop key K_3 matches that of the outer encrypted identity layer and that this key is correctly signed by $\mathrm{sig}_{K_2}(K_3)$. Again, if any verification fails, the middle relay behaved maliciously, and the consortium reports this as likely collusion with the client to the LEA. Otherwise, the middle relay acted according to the Seldom protocol.

If a circuit consists of more than three relays, the same procedure can be used to iteratively unveil further intermediate layers of the encrypted identity. Only the last layer is treated differently. While the encrypted identities are handled by the consortium, the plaintext origin of the traffic is only communicated to the requesting LEA. Therefore, the consortium members only share the final decryption shares with the LEA. Thus, no member of the consortium learns this identity. This restriction maximizes privacy, as only the requesting LEA learns the client's identity, while the consortium only knows the relays.

The LEA then still has to verify the correctness of the decrypted client identity to ensure that the entry relay did not manipulate this information. Therefore, the LEA first verifies that the IP certificate is valid and that a trusted CA issued it. Additionally, the LEA verifies the correctness of the hop key K_1, analogously to this verification by the consortium as previously discussed. To this end, each consortium member shares the hop key K_2 claimed in the previous layer with the LEA along its decryption share. If both checks succeed, the revealed identity is indeed that of the user from whom the suspicious traffic originated. Otherwise, the entry relay attempted to cover up the true identity, and the LEA should investigate this. Through Seldom's iterative process, the requested identity is thus only revealed to the requesting LEA, while misbehavior (*i.e.*, sending no or false data) during the anonymization process can be clearly attributed to one relay.

6.4 Probing Relay Honesty

Deanonymizations reveal misbehaving relays. Such misbehavior could occur due to collusion with the client, general malice toward the network, or from genuine errors. Therefore, the relay's protocol abidance under normal circumstances should be verified without risking individuals' privacy, such that misbehaving nodes can be fixed or removed from the network. Regular probing by the directory authorities reduces the likelihood that an exit relay's key loss in a critical situation is coincidental. This feature significantly strengthens our assumption that failures during deanonymization are due to a relay colluding with a client, warranting further investigation.

To generate probe traffic for verifying the honesty of relays, Seldom relies on cryptographic commitments. We use Pedersen commitments [43], but other commitment schemes could also be employed. To create a commitment, a directory authority, *i.e.*, the prober, first generates a key later used as the client's identification key K_3 during the probe. Then, the prober hashes this key, the probed exit relay, a traffic destination, and a timeframe during which the probe is active. We refer to the resulting digest as probe p. To then create a Pedersen commitment to the probe p, the prober generates a random value r and computes the commitment $C = pG + rH$ on an elliptic curve with generators G and H.

The prober then invokes a smart contract on the translucent ledger to store C. Afterward, the prober establishes a new circuit with the probed relay as the last onion router and sends traffic as specified in the commitment and using the pre-generated client identification key. Afterward, the prober opens its commitment C and discloses the preimage of p on the ledger. Then, the prober performs a deanonymization request, using the probe as justification for the deanonymization. The consortium accepts this request, and the exit relay is asked to initiate the deanonymization process.

At this point, the exit relay does not know that the deanonymization request is only for probe traffic. It learns that the vote for the request was successful but does not see the reasoning behind it, which is only revealed during the delayed disclosure of the request to the public. The exit relay thus is expected to cooperate like for ordinary deanonymization. Afterward, the consortium verifies the signature of the database key $H(\mathrm{ID}_3)$ with the public key indicated in the commitment. If this verification is successful, the exit relay passed the probe. Otherwise, the directory authority no longer accepts this exit relay. The other directory authorities can adopt this assessment but are encouraged to verify it through independent probes. Otherwise, a single directory authority would have the power to remove relays at will by claiming they failed a probe. However, a malicious relay may try to detect such additional probes and behave honestly for them to not be removed from the network. Hence, once suspicious, a relay should be probed with a higher frequency at random intervals to ensure any misbehavior was indeed a one-off event. The consortium can also further investigate the encrypted identity to identify non-exit relays that are misbehaving. Through regular probing, the consortium and the LEAs can be much more confident that each relay behaves correctly during deanonymization.

In general, there are two concerns that dictate the selection of a sensible probing frequency: strain on the network and detectability. The exit bandwidth consumption of one probe is dominated by the size of the accessed web page, where a median web page is 2.8 MB in size [30]. Having each directory authority probe each exit relay once per day would thus result in not even 1 MB/s of exit bandwidth consumption. To put this overhead into perspective, we can look at bandwidth authorities [6], which consume orders of magnitude more bandwidth for relay capacity estimations [17]. Moreover, to avoid that probes are detected, the respective deanonymization needs to take time to match the manual decision-making within the consortium. Given that executing multiple probes in parallel likely adds limited value, a probing frequency above one probe per day is unrealistic.

Overall, probing is expected to make up the bulk of all requests. As the public is informed about each request immediately, all probers are encouraged to reveal that a probe took place immediately after conclusion. Therefore, they can call a special function in the smart contract that triggers an event to inform the public, such that the public always knows how many requests actually deanonymize a user.

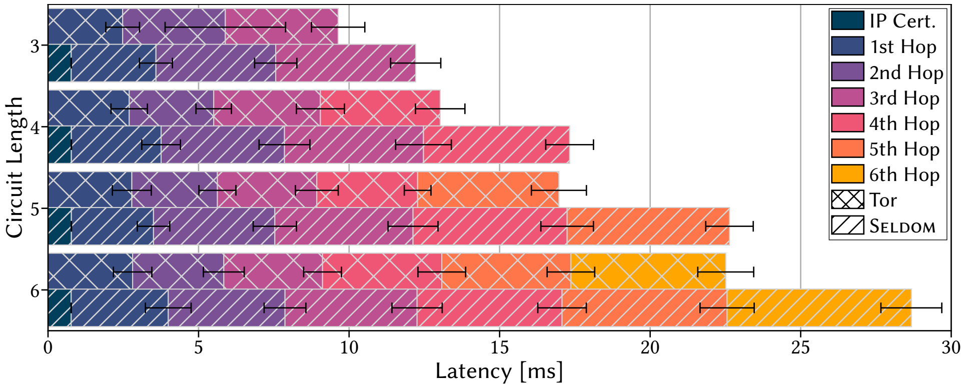


Fig. 6. During circuit establishment, Seldom adds only single digit milliseconds of delay, which is overshadowed by the latency of a global network. Error bars show 95 % confidence intervals. The processing overhead of Seldom is not noticeable to users and thus unlikely to cause an issue.

7 Performance Evaluation

By comparing Seldom's performance to Tor, we now assess the costs of realizing anonymity with translucent selective deanonymization capabilities.

7.1 Evaluation Setup

Most new Tor features are evaluated through simulations. However, simulation abstracts from Tor's cryptographic processing [33], which is precisely the part impacted by Seldom (and not the behavior of the network). Thus, we evaluate Seldom in a local deployment. Since our changes do not impact the network as a whole, but rather individual relays or circuits, this still gives accurate results on Seldom's overhead.

We modify Tor version 0.4.1.5 to implement Seldom. As our target host for connection establishments, we run an *apache2* web server. We additionally host a local CA for IP certificates and a *MongoDB* database to store deanonymization information. The consortium members' behavior is expressed in Python, and the translucent ledger is a permissioned Ethereum blockchain [59] modified to realize the translucency properties. Our server contains two Intel® Xeon® Silver 4116 processors (24 CPU cores @ 2.1 GHz) and 204 GB of RAM. We run nine relays: one directory authority, three exit relays, and five non-exit relays. Each process (*e.g.*, databases, Tor clients, Tor relays) is pinned to one CPU core to prevent cross-process interference.

7.2 Circuit Establishment

First, we measure the overhead of Seldom during circuit establishment. All communication happens through the loopback interface, which means that we do not include latency between different entities. During the evaluation, the client periodically establishes new circuits and logs whenever a circuit is extended by a new hop. For Seldom, the client obtains a new IP certificate before each circuit establishment (usually not necessary, as these certificates are reusable and can be obtained in advance). We establish approx. 900 circuits in Tor and Seldom for each evaluated circuit length.

We compare the latency to extend a circuit by one hop at a time in Figure 6. The x-axis shows the elapsed time, while the different horizontal bars represent circuits of various lengths in Seldom or Tor, respectively. The cumulative bars represent the total time to establish a circuit.

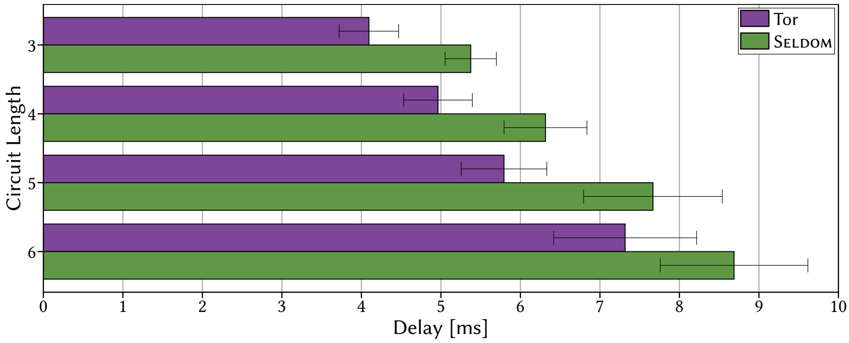


Fig. 7. The average latency to retrieve the first byte when connecting to a new host through an established circuit shows barely perceivable overhead by Seldom.

We measure a processing overhead to extend the circuit by one hop that increases from 7.4 % for the first hop to 19.6 % for the sixth hop. This increase arises from the larger encrypted identities of longer circuits. Overall, we observe a resulting latency increase of 3.2 ms to 4.3 ms, depending on circuit length. The receipt of an IP certificate causes an overhead of 0.8 ms. The remaining overhead stems from signature processing, threshold encryption, and transferring the partially encrypted identity to the next hop. All these steps require more time the more previous hops exist (as more data has to be processed). The observed variance in timings results from occasional failures of the path selection algorithm due to the small network size. In a global overlay network, additional latency in the range of multiple 100 milliseconds and jitter would be added by the network and make this overhead insignificant. Overall, Seldom thus does not perceivably impact circuit establishment.

7.3 Connection Establishment

Analyzing the impact of Seldom on connection establishment is essential, as additional latency in this step directly affects the user experience. To assess this overhead, we request a web page from the locally hosted web server through *cURL*, which uses *socks5* to communicate with the Tor client. We measure the time to first byte, *i.e.*, the time between issuing the request and receiving the first byte through the established circuit. We repeat this measurement for different circuit lengths (approx. 700 measurements each) and compare a vanilla Tor deployment to Seldom. Our results are shown in Figure 7. We observe that Seldom takes, on average, 1.5 ms longer than Tor to establish a connection and receive the first byte of the response, independent of the circuit length. This delay stems from the generation and verification of the flow record signature as well as writing this flow record to a local database before the connection is established. Again, these delays are overshadowed by the latency of the overlay network, which is two orders of magnitude higher.

7.4 Bandwidth and Storage Requirements

While Seldom barely impacts performance, the additional data that is transferred and stored risks becoming the real bottleneck. To quantify this overhead, we first measure how much bandwidth exit relays consume to transfer all deanonymization data to a state-sponsored database. During our evaluation, the client establishes circuits with a length of three and a fixed exit relay. A fixed number of connections is established to random IP addresses and ports through each circuit before it is discarded. The exit relay waits for a predefined amount of closed circuits before it migrates the combined local storage for these circuits to the state-sponsored database. When migrating, *mongodump* is executed on the local *MongoDB* database, and the resulting folder is serialized

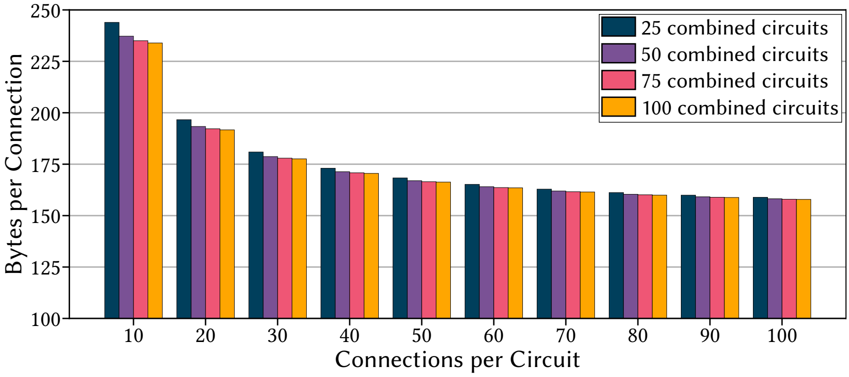


Fig. 8. Considering all data that is transmitted when flushing deanonymization data to the database, each connection consumes between 156 and 244B of data. This overhead shrinks the more connections are established per circuit and as more circuit's data is flushed together.

and compressed using *zlib*. The resulting archive is sent over a TLS channel to a state-sponsored database. Simultaneously, its hash is sent to the translucent ledger to ensure consistency across multiple governmental databases.

In Figure 8, we show how bandwidth usage depends on the number of connections established per circuit and how often the local buffer is flushed. Flow records are approximately an order of magnitude smaller than encrypted identities, which are only generated for each circuit and not each connection. Furthermore, the more data is migrated to the database in one go, the more the local buffer can be compressed. However, the benefit of sending more data at once levels off quickly, indicating that frequent flushing is desirable. While using an established circuit for more connections is beneficial, such a change directly influences the susceptibility to deanonymization attacks. Recent measurements by Mani *et al.* estimate the average number of established connections per circuit to be about 20 [37]. An exit relay that migrates its database after 100 circuits have been established and used to establish 20 connections each would thus produce 191.7 byte of data per connection.

The measurements of Mani *et al.* also allow us to predict the storage requirements of a Tor-sized SELDOM network. For this prediction, we project the amount of generated deanonymization data based on the activity in the Tor network over time. According to Mani *et al.*, an average of 2.1 billion (95%-CI:[1.7 billion; 2.5 billion]) exit streams were created every 24 hours in April 2018 [37]. Over the same period, the Tor Project reported an average of 111.55 Gbit/s of consumed bandwidth by the Tor network [53]. For our estimations, we assume a constant relationship between the amount of consumed bandwidth and the amount of exit streams over the last six years. This is a worst-case assumption, as the size of web pages grew by over 29 %, while the number of TCP connections per web page shrunk by about 47 % over the same period [30]. Thus, in practice, the additional bandwidth consumed by Tor is, in part, caused by larger web pages that are downloaded over fewer connections.

Figure 9 thus quantifies the bandwidth overhead of SELDOM in a Tor-sized network according to those estimations. Here, the blue line shows the global bandwidth consumed by all exit relays to write data into the state-sponsored database over time with 95 % confidence intervals (blue area). In January 2020, all exit relays combined would consume 67.7 Mbit/s (95%-CI:[54.8 Mbit/s; 80.6 Mbit/s]) to write deanonymization data to the state-sponsored database. To put these numbers into perspective, this would mean a mere 0.11 % (95%-CI:[0.09 %; 0.013 %]) increase in data

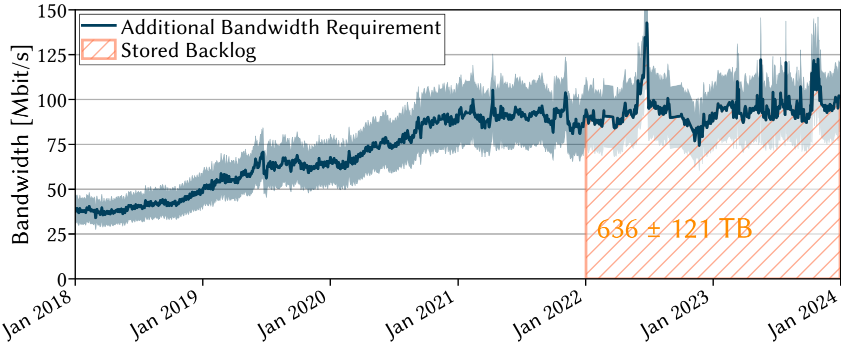


Fig. 9. SELDOM increases exit bandwidth by 0.1% to store deanonymization data. The total storage requirement for a retention period of two years amounts to 636 ± 121 TB.

transmitted by exit relays compared to Tor. A SELDOM exit relay advertising 50 Mbit/s of exit traffic per second would thus insert approx. 56 kB/s of deanonymization data into the database. This estimate also means that this same relay has to, on average, verify 385 ed25519 signatures and compute 18 hybrid threshold encryptions per second, a processing overhead hardly impacting exit relays. Figure 9 also highlights the amount of data that would be retained in the state-sponsored database, assuming a retention period of two years. The database would thus have to store 636 TB (95%-CI:[515 TB; 757 TB]) of deanonymization data, which can easily be provided by government organizations.

SELDOM can thus easily scale to Tor's current size. If SELDOM were to significantly outgrow Tor, there exist several strategies to improve scalability. First, websites that are unlikely to be abused for illegal activities (*e.g.*, news websites) could be whitelisted, *i.e.*, exit relays would not store flow records (and encrypted identities if only such websites are visited). Secondly, SELDOM could rely on *probabilistic deanonymization*, where only a randomly selected subset of identities is deanonymizable. Thus, the consortium could retroactively select a fraction of records that are persistently stored through provable randomness. Thus, bandwidth and storage overhead are unlikely to limit SELDOM's scalability.

7.5 Costs of Deanonymizations

Finally, we measure the processing overhead and data transfers during the deanonymization process. Here, we look at an increasing threshold of required consortium members participating in the deanonymization. The total size of the consortium does not influence the performance. For each measurement, we consider a new SELDOM-anonymized circuit and start with the deanonymization request by an LEA. We repeat each measurement 40 times. We ignore network latency (negligible) and assume instantaneous decision-making by humans in the loop (which would likely take hours to days in real life).

We plot the time required to decrypt the layer-encrypted identities in Figure 10 (stacked bars on the left of each group). Here, the time for the first decryption layer also includes the time for the flow record. Potentially surprisingly, the last step in the decryption is notably faster than the previous steps. This speedup results from decryption shares being collected only by the LEA and not all consortium members. Overall, we observe a linear growth in processing with an increase in consortium members. Still, even for 75 members, processing on a single CPU core per participant requires only 32.8 seconds (95%-CI:[32.5 s; 33.2 s]). Thus, processing is orders of magnitude faster than human processes and does not constitute a processing bottleneck.

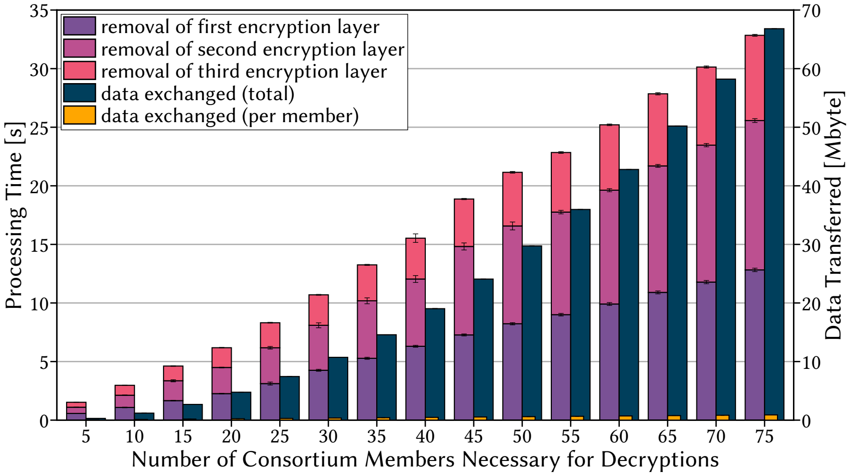


Fig. 10. Even large consortia deanonymize an identity in under a minute with minimal data exchange. Humans in the loop are expected to cause significantly larger overhead.

The other relevant performance metric is the amount of exchanged data. We thus report the total (dark blue bar) and per-consortium member (yellow bar) amount of application layer data sent and received by each party during deanonymization in Figure 10. For the exchanged data per consortium member, we observe linear growth with the number of participants. As all data is broadcasted to all participants, we observe quadratic growth in the total amount of exchanged data. Still, the total amount of exchanged data is well below 100 MB and thus insignificant. Overall, we conclude that the deanonymization process is efficient, and manual processes outside SELDOM, such as the identification of suspicious traffic and the assessment of requests, are significantly slower.

8 Security Discussion

After having evaluated its performance, we now discuss how SELDOM protects against potential threats. SELDOM's threat model (*cf.* Section 4.2.1) discusses the two potential goals of an adversary concerning its deanonymization capabilities: (i) evade deanonymizations, possibly by misattributing suspicious traffic to an innocent client, or (ii) conduct unauthorized deanonymizations.

To analyze how SELDOM thwarts attempts to bypass its deanonymization procedure, we examine the individual actors within the network. While the database operators and the consortium as a collective are considered trustworthy, relays, CAs, and individual consortium members may act maliciously (*cf.* Section 4.2.2). In the following, we analyze how SELDOM technically prevents individuals and colluding entities from evading deanonymizations (Section 8.1) or performing unauthorized deanonymizations (Section 8.2). Finally, we discuss additional forms of collusion (Section 8.3) that lie outside the scope of SELDOM's direct technical protections.

8.1 Evading Deanonymizations

First, we look at how SELDOM prevents entities from evading deanonymizations. No relay, consortium member, or client should be able to tamper with deanonymization without being clearly identified as the culprit. Malicious relays, as well as consortium members, could try to mix deanonymization data across different circuits to mount an attack. Likewise, clients could route traffic through a self-hosted relay to cover their tracks. In the following, we see how the SELDOM protocol protects against such attacks.

8.1.1 Reusing Encrypted Identities. A misbehaving relay may try to hinder deanonymizations by reusing the encrypted identity of a previous client, thereby protecting the real client's anonymity and even blaming an innocent client for potentially suspicious traffic. Here, the attacker's idea is that during deanonymizations, the wrong encrypted identity is decrypted. Therefore, the relay could include an encrypted identity ID_n during two different relay extension handshakes. However, in this case, the hop key K_{n+1} in the encrypted identity would not match the advertised hop key K_{n+1} in the following encrypted identity ID_{n+1}. This mismatch can be undeniably blamed on the relay during deanonymizations.

Alternatively, the relay could advertise an encrypted identity ID_n, the hop key K_{n+1}, relay identity R_n^{id}, and its signatures on two different relay extensions. In this case, the next relay would, however, never receive a valid signature $sig_{K_{n+1}}(K_{n+2})$ during the next step of the circuit extension. Hence, the circuit would never be established. In neither case is it possible for a malicious relay to undetectably reuse an encrypted identity to extend another client's circuit.

8.1.2 Selectively Misbehaving Self-Hosted Relay. Similarly, malicious clients could attempt to cover up their traces by hosting their own relay. This relay would behave honestly for all other clients but incorrectly process the encrypted identity when a client creates a circuit through its own relay. In this case, the deanonymization process cannot identify the client. However, the deanonymization process ensures that the self-hosted relay is identified as misbehaving. Hence, the relay operator and consequently the client are identified as suspicious and will be further investigated by the LEA.

8.2 Deanonymization Attacks

While some adversaries may want to prevent deanonymizations through Seldom, others may aim to perform unauthorized deanonymizations of clients. In the following, we look at different attacker strategies, ranging from classical traffic analysis to collusion attacks, and how Seldom defends against these strategies.

8.2.1 Traffic Analysis Attacks. Traffic analysis is a common attack to deanonymize users of anonymity networks [38, 42]. While Tor does not explicitly protect against global adversaries [20], *i.e.*, an entity that intercepts all traffic, analyzing traffic between two or multiple nodes is often possible for adversaries. For our discussion, we focus particularly on the additional information that may be leaked about the client in Seldom compared to Tor. An observer intercepting the outgoing traffic of a client not using a bridge relay learns two things: the hop key K_1 of the circuit and the client's intention to use Seldom. The hop key K_1 does not help in deanonymizing a circuit, and the intention to use the network is also leaked in Tor.

As the relays communicate through TLS channels, even during circuit establishment, no additional information is leaked besides the amount of transmitted data. However, this information is unlikely to give an adversary an advantage in detecting circuit extensions, as, usually, many circuits are multiplexed over a single link. In the end, such attacks can also be executed against traffic patterns of different websites [27, 41, 42], which is possible for both Tor and Seldom. All traffic leaving exit relays is either anonymized client traffic or encrypted deanonymization information. Neither leaks any additional information.

8.2.2 Mismatching Encrypted Identities. A malicious consortium member could potentially use partial deanonymizations to leak information about users. Therefore, the malicious member could set up a circuit that includes the threshold-encrypted identity of a targeted user. The consortium could be tricked to decrypt this identity, *e.g.*, by marking the circuit as probe traffic.

However, such an attack would be detected after the consortium removes one layer of threshold encryption due to an invalid signature. The corresponding relay would be flagged as suspicious, but the malicious entity, as part of the consortium, would learn the identity of the preceding relay in the targeted circuit. The attacker could repeat this step and eventually learn the client's identity. However, the consortium would notice the second iteration of this attack as the plaintexts are from different layers of the same circuit and thus clearly identify an attack from within the consortium. Thus, the client's identity remains secure, as long as the attacker does not operate the entry relay of the targeted user. Additionally, the rest of the consortium and eventually also the public would learn about this one-off attack and could abandon the network or hold the consortium member accountable.

8.2.3 Collusion among Relays. SELDOM enables the exit relay to learn the encrypted client identity (*cf.* Figure 2) and indisputably link connections to it while preserving client anonymity (*cf.* Section 5). To show that this claim holds, we analyze which relays have access to which data and derive whether any relay has unintended access to client-identifying data. Here, we consider information that can be correlated within one circuit and across different circuits.

During circuit establishment, the entry relay learns the client's IP certificate, the hop keys K_1 and K_2, as well as the signature created with K_1. The certificate (and consequently the reuse of K_1 across multiple circuit establishments) contains no new information about the client, as the entry relay knows the client's IP address and the current time.

The hop key K_2 and, consequently, the signature created with K_1, are unique per circuit and derived from a secure source of randomness. Thus, they do not allow any relay to gain information about clients from other circuits. The n-th middle relay knows the hop keys K_n and K_{n+1}, the identity of the previous relay, and signatures by the previous relay and K_n. All other information is threshold-encrypted by the $(n-1)$-th relay. This nested encryption ensures that no earlier relay can match any information to deanonymize the client. In Tor, intermediary relays also know their direct neighbors. As K_n is only shared with the preceding relay and K_{n+1} only with the following relay, non-connected relays cannot notice that they are part of the same circuit. Crucially, the signature as the only link between these two keys is not shared with any other relay.

Finally, all signatures are derived from different keys over different data; thus, no information is leaked. The exit relay also learns about the traffic sent through the circuit (as in Tor) and stores this information in encrypted form. Additionally, matching different flow records to the same circuit is also not possible, as the circuit-identifying information is encrypted. Thus, as we prevent information leakage for each data field known to the different relays, the client remains anonymous.

8.3 Other Collusion Scenarios

Beyond the threats that SELDOM explicitly defends against, additional collusion scenarios remain outside the scope of direct technical protection. Nonetheless, collusion between a CA and an entry relay as well as collusion among consortium members, is unlikely to materially compromise user anonymity.

8.3.1 CA and Entry Relay Collusion. A potential protocol-based attack arises from the possible collusion between a CA and an entry relay. Here, the malicious entities could create a fictional client with any IP address that is certified by the CA. If the circuit is then deanonymized for some reason, *e.g.*, suspicious activities by the fictional client, the deanonymization process reveals the fake IP address. In this case, a powerful adversary could thus use SELDOM to incriminate any person browsing the Internet. However, such incrimination would also be possible with other means by such adversaries. Most importantly, SELDOM only removes anonymity and gives LEAs hints for further investigations. IP addresses, allegedly used for illegal activities with SELDOM or on other platforms, need to be supported by further evidence, as has been concluded by several courts [1–3].

Thus, such collusion by powerful entities does not yield them capabilities they would not have without the existence of Seldom.

8.3.2 Collusion Within the Consortium. To ensure that the risk of being covertly deanonymized in Seldom through collusion of a majority of consortium members and an exit relay is comparable to Tor, the consortium must be assembled accordingly. Here, it is important to recall that Tor does not fully protect the anonymity of its users but only decreases the chances that an attacker can learn a client's identity [7]. One attack to deanonymize a client in Tor is through the collusion of the three relays that make up a client's circuit. Thus, Seldom's deanonymizations should be more resilient to collusions than collusion of three randomly selected relays. As the exit relay is involved in the deanonymization attack in Tor and Seldom, we can remove it from the equation.

Hence, we compare the risk of collusion in Seldom's consortium to the risk of collusion of two randomly selected relays. There are multiple reasons why Seldom's consortium is significantly less likely to collude than these two relays: while anyone can deploy massive numbers of relays, the identities of the consortium members are publicly known. Specifically, it would be necessary to convince a significant fraction of privacy advocates, *e.g.*, some of the directory authorities, and foreign governments to partake in such an attack. Even attempting this could result in significant repercussions, which makes a successful collusion attack among the consortium members highly unlikely. We also designed Seldom in such a way that a higher number of members must participate in a decryption than have to accept a deanonymization vote, *i.e.*, some disagreeing members have to accept the consensus and cooperate, to further minimize the risk of collusion.

9 Ethical and Technical Concerns

The safety-privacy dilemma remains particularly divisive, with LEAs and privacy advocates maintaining seemingly fundamentally incompatible positions regarding a problem that requires some sort of compromise [56]. Fostering the understanding of technical means for enabling exceptional access without the dooming vision of privacy-invasive legislation or hidden backdoors for LEAs, we address frequent concerns regarding Seldom's design in the following.

Q1 – Do we advertise for Seldom to replace Tor? No. Seldom relies on translucency and public oversight, which means that observed misbehavior by Seldom's consortium must be met with revocations of these privileges, *i.e.*, a migration of the user base back to Tor. In the far future, Seldom might attract a significantly larger user base than Tor and thus offer better anonymity even when accounting for selective deanonymization capabilities, especially if users start abandoning Tor. For Seldom's security concept to work, there must, however, always exist a quick and easy way to migrate back to an anonymity network without deanonymization capabilities, even if temporarily deserted. This could be achieved with a browser that lets the users easily, and potentially collaboratively, toggle whether to use Tor or Seldom.

Q2 – Can citizens trust the consortium? The underlying assumption for Seldom is that a majority of consortium members are trustworthy and value privacy by default, except for extraordinary situations that warrant deanonymization. Those members therefore have little incentive to partake in collusions, especially considering the high risk of such attempts being revealed to the public through the several translucency properties of Seldom. Further, Seldom protects against individual malicious consortium members. Namely, the majority of honest consortium members will faithfully delete their secret shares after encrypted identities reach the end of the mandated retention period or in extreme events, such as an imminent government takeover. This way, the consortium renders older encrypted identities unusable for malicious members even in the case of an anticipated dishonest majority gaining control over Seldom. However, maintaining *transparency* *about the goals and non-goals* of Seldom remains important to avoid creating false expectations about Seldom's protection.

Q3 – Can individual relays misbehave? Once a relay misbehaves and does not comply with deanonymization requests, they will become suspicious of the requesting LEA. Hence, they would reveal themselves as targets for further investigation. To reduce the number of misbehaving relays, Seldom can further use active probing (*cf.* Section 6.4) and remove any relays that misbehave.

Q4 – Can investigations fail then? In exceptional cases, LEAs will not be able to find the true root of harmful traffic using Seldom alone. For instance, misbehaving relays may cover up specific bad actors, or bad actors may establish circuits via compromised devices. However, all these cases lead LEAs to a distinct location worth of further forensic investigation. Hence, Seldom will not always guarantee an immediate investigation success, but it provides crucial pointers to narrow down additional steps.

Q5 – Can Seldom support hidden services? Technically, Seldom can integrate hidden services. Each introduction and rendezvous point would be associated with an encrypted identity, which could be deanonymized if deemed necessary. The consortium must, however, consider the increased privacy implications of agreeing to such a request. If a Seldom-like system were to be deployed, it should, however, be carefully considered whether hidden services should be supported or not.

Q6 – Could Seldom interfere with Tor's service? Initially, Seldom targets the population of privacy-conscious users who currently shy away from using Tor due to having negative associations by tying Tor to cyber-criminality. Only if Seldom is capable of establishing a reputation as the infrastructure of choice for privacy-seeking, law-abiding citizens, a potential user migration from Tor to Seldom could take place. As a positive effect, this would actively harm the achievable anonymity for cybercriminals, as they cannot hide among a mass of benign traffic. On the other hand, it becomes harder to abandon Seldom if a deserted Tor network offers little anonymity. To ensure that such a situation does not encourage extensive surveillance by the consortium, it must be easy for the user base to switch back to Tor at once, *e.g.*, by offering a Tor and Seldom-compatible client that, by default, switches to Tor if Seldom is collectively deemed unsafe.

10 Conclusion

With this paper, we investigate the potential of selective deanonymizations for anonymity networks. We find that Seldom's design barely impacts performance in comparison to Tor, while data overhead remains manageable. On the other hand, Seldom relies on a translucent trusted consortium and potentially encourages the migration of users from Tor, which would weaken the anonymity provided by Tor. With this paper we thus contribute to understanding the advantages and disadvantages of selective deanonymizations in anonymity networks like Seldom. These insights should encourage future research to mitigate drawbacks, and, most importantly, they should foster more open, objective, and technical discussions in inevitable future political debates about exceptional access.

References

[1] 2011. Media CAT Limited v. Adams & Ors . (EWPCC). http://www.bailii.org/ew/cases/EWPCC/2011/6.html.

[2] 2012. K-Beech, Inc. v. John Does. 2:11-cv-03995 (E.D.N.Y.). https://cite.case.law/frd/296/80/.

[3] 2016. Cobbler Nev., LLC v. Gonzales. 17-35041 (D. Or.). https://law.justia.com/cases/federal/appellate-courts/ca9/17-35041/17-35041-2018-08-27.html.

[4] Harold Abelson, Ross Anderson, Steven M Bellovin, Josh Benaloh, Matt Blaze, Jon Callas, Whitfield Diffie, Susan Landau, Peter G Neumann, Ronald L Rivest, et al. 2024. Bugs in our pockets: The risks of client-side scanning. *Journal of Cybersecurity* 10, 1 (2024). DOI: 10.1093/cybsec/tyad020.

[5] Harold Abelson, Ross Anderson, Steven M Bellovin, Josh Benaloh, Matt Blaze, Whitfield Diffie, John Gilmore, Matthew Green, Susan Landau, Peter G Neumann, et al. 2015. Keys under doormats: mandating insecurity by requiring

government access to all data and communications. *Journal of Cybersecurity* 1, 1 (2015). DOI: 10.1093/cybsec/tyv009.

[6] Mashael AlSabah and Ian Goldberg. 2016. Performance and Security Improvements for Tor: A Survey. *ACM Computing Surveys (CSUR)* 49, 2 (2016).

[7] Arma. 2014. Traffic correlation using netflows. https://blog.torproject.org/traffic-correlation-using-netflows?page=1 Last accessed: May 17, 2020. https://blog.torproject.org/traffic-correlation-using-netflows.

[8] Michael Backes, Jeremy Clark, Aniket Kate, Milivoj Simeonovski, and Peter Druschel. 2014. BackRef: Accountability in Anonymous Communication Networks. In *International Conference on Applied Cryptography and Network Security (ACNS).* DOI: 10.1007/978-3-319-07536-5_23.

[9] James Bartusek, Sanjam Garg, Abhishek Jain, and Guru-Vamsi Policharla. 2023. End-to-end secure messaging with traceability only for illegal content. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques (Eurocrypt).* DOI: 10.1007/978-3-031-30589-4_2.

[10] Mihir Bellare and Shafi Goldwasser. 1997. Verifiable Partial Key Escrow. In *Proceedings of the 4th ACM conference on Computer and communications security (CCS).* DOI: 10.1145/266420.266439.

[11] Mihir Bellare and Ronald L Rivest. 1999. Translucent Cryptography—An Alternative to Key Escrow, and Its Implementation via Fractional Oblivious Transfer. *Journal of cryptology* 12, 2 (1999). DOI: 10.1007/PL00003819.

[12] Sanjit Bhat, David Lu, Albert Kwon, and Srinivas Devadas. 2019. Var-CNN: A Data-Efficient Website Fingerprinting Attack Based on Deep Learning. *Proceedings on Privacy Enhancing Technologies* 4 (2019).

[13] Matt Blaze. 1996. Oblivious Key Escrow. In *International Workshop on Information Hiding.* DOI: 10.1007/3-540-61996-8_50.

[14] Xiang Cai, Xin Cheng Zhang, Brijesh Joshi, and Rob Johnson. 2012. Touching from a distance: Website fingerprinting attacks and defenses. In *Proceedings of the 2012 ACM conference on Computer and communications security.*

[15] Giovanni Cherubin, Rob Jansen, and Carmela Troncoso. 2022. Online website fingerprinting: Evaluating website fingerprinting attacks on tor in the real world. In *31st USENIX Security Symposium (USENIX Sec' 22).*

[16] Reza Curtmola, Juan Garay, Seny Kamara, and Rafail Ostrovsky. 2006. Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions. In *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS '06).*

[17] Hussein Darir, Hussein Sibai, Chin-Yu Cheng, Nikita Borisov, Geir Dullerud, and Sayan Mitra. 2022. MLEFlow: Learning from History to Improve Load Balancing in Tor. *Proceedings on Privacy Enhancing Technologies (PETS '22)* (2022).

[18] Dorothy E Denning and Dennis K Branstad. 1996. A Taxonomy for Key Escrow Encryption Systems. *Commun. ACM* 39, 3 (1996). DOI: 10.1145/227234.227239.

[19] Roger Dingledine and Nick Mathewson. 2006. Anonymity Loves Company: Usability and the Network Effect. In *In Proceedings of the Fifth Workshop on the Economics of Information Security (WEIS).*

[20] Roger Dingledine, Nick Mathewson, and Paul Syverson. 2004. Tor: The Second-Generation Onion Router. In *13th USENIX Security Symposium (USENIX Sec'04).* DOI: 10.5555/1251375.1251396.

[21] Arun Dunna, Ciarán O'Brien, and Phillipa Gill. 2018. Analyzing China's Blocking of Unpublished Tor Bridges. In *8th USENIX Workshop on Free and Open Communications on the Internet (FOCI '18).* https://www.usenix.org/system/files/conference/foci18/ foci18-paper-dunna.pdf.

[22] Kevin P Dyer, Scott E Coull, Thomas Ristenpart, and Thomas Shrimpton. 2012. Peek-a-boo, i still see you: Why efficient traffic analysis countermeasures fail. In *2012 IEEE Symposium on Security and Privacy.*

[23] Valerie Fetzer, Michael Klooß, Jörn Müller-Quade, Markus Raiber, and Andy Rupp. 2023. Universally composable auditable surveillance. In *International Conference on the Theory and Application of Cryptology and Information Security (Asiacrypt '23).* DOI: 10.1007/978-981-99-8724-5_14.

[24] Aaron Gable. 2025. We've Issued Our First IP Address Certificate. https://letsencrypt.org/2025/07/01/issuing-our-first-ip-address-certificate/ Last accessed: July 4, 2025.

[25] Kevin Gallagher, Sameer Patil, and Nasir Memon. 2017. New Me: Understanding Expert and Non-Expert Perceptions and Usage of the Tor Anonymity Network. In *Thirteenth Symposium on Usable Privacy and Security (SOUPS '17).* 385–398.

[26] Matthew Green, Gabriel Kaptchuk, and Gijs Van Laer. 2021. Abuse resistant law enforcement access systems. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques (Eurocrypt '21).* DOI: 10.1007/978-3-030-77883-5_19.

[27] Jamie Hayes and George Danezis. 2016. k-fingerprinting: A robust scalable website fingerprinting technique. In *25th USENIX Security Symposium (USENIX Sec '16).* DOI: 10.5555/3241094.3241186.

[28] Naoki Hiramoto and Yoichi Tsuchiya. 2023. Are illicit drugs a driving force for cryptomarket leadership? *Journal of Drug Issues* 53, 3 (2023).

[29] Ashish Hooda, Andrey Labunets, Tadayoshi Kohno, and Earlence Fernandes. 2024. Experimental Analyses of the Physical Surveillance Risks in Client-Side Content Scanning. (2024). DOI: 10.14722/ndss.2024.241401.

[30] http archive. 2024. State of the Web. https://httparchive.org/reports/state-of-the-web. Last accessed: June 28, 2025.

[31] Shubham Jain, Ana-Maria Crețu, Antoine Cully, and Yves-Alexandre de Montjoye. 2023. Deep perceptual hashing algorithms with hidden dual purpose: when client-side scanning does facial recognition. In *IEEE Symposium on Security and Privacy (Oakland '23).* DOI: 10.1109/SP46215.2023.10179310.

[32] Shubham Jain, Ana-Maria Crețu, and Yves-Alexandre de Montjoye. 2022. Adversarial Detection Avoidance Attacks: Evaluating the robustness of perceptual hashing-based client-side scanning. In *31st USENIX Security Symposium (USENIX Sec '22).*

[33] Rob Jansen and Nicholas Hopper. 2012. Shadow: Running Tor in a Box for Accurate and Efficient Experimentation. In *Proceedings of the 19th Symposium on Network and Distributed System Security (NDSS).*

[34] Simon Jentzsch and Christoph Jentzsch. 2018. EIP-1186: RPC-Method to get Merkle Proofs. https://github.com/ethereum/EIPs/issues/1186.

[35] Marc Juarez, Sadia Afroz, Gunes Acar, Claudia Diaz, and Rachel Greenstadt. 2014. A critical evaluation of website fingerprinting attacks. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security.*

[36] Anunay Kulshrestha and Jonathan Mayer. 2021. Identifying Harmful Media in End-to-End Encrypted Communication: Efficient Private Membership Computation. In *30th USENIX Security Symposium (USENIX Sec '21).* https://www.usenix.org/system/files/sec21-kulshrestha.pdf.

[37] Akshaya Mani, T Wilson-Brown, Rob Jansen, Aaron Johnson, and Micah Sherr. 2018. Understanding Tor Usage with Privacy-Preserving Measurement. In *Proceedings of the Internet Measurement Conference (IMC '18).* DOI: 10.1145/3278532.3278549.

[38] Steven J Murdoch and George Danezis. 2005. Low-cost traffic analysis of Tor. In *IEEE Symposium on Security and Privacy (Oakland).* DOI: 10.1109/SP.2005.12.

[39] Juha Nurmi, Arttu Paju, Billy Bob Brumley, Tegan Insoll, Anna K Ovaska, Valeriia Soloveva, Nina Vaaranen-Valkonen, Mikko Aaltonen, and David Arroyo. 2024. Investigating child sexual abuse material availability, searches, and users on the anonymous Tor network for a public health intervention strategy. *Scientific Reports* 14, 1 (2024), 7849.

[40] Se Eun Oh, Nate Mathews, Mohammad Saidur Rahman, Matthew Wright, and Nicholas Hopper. 2021. GANDaLF: GAN for data-limited fingerprinting. *Proceedings on Privacy Enhancing Technologies* 2021, 2 (2021).

[41] Andriy Panchenko, Fabian Lanze, Andreas Zinnen, Martin Henze, Jan Pennekamp, Klaus Wehrle, and Thomas Engel. 2016. Website Fingerprinting at Internet Scale. In *Proceedings of the 23rd Annual Network and Distributed System Security Symposium (NDSS).* DOI: 10.14722/ndss.2016.23477.

[42] Andriy Panchenko, Lukas Niessen, Andreas Zinnen, and Thomas Engel. 2011. Website fingerprinting in onion routing based anonymization networks. In *Proceedings of the 10th annual ACM workshop on Privacy in the electronic society (WPES).* DOI: 10.1145/2046556.2046570.

[43] Torben Pryds Pedersen. 1991. Non-interactive and information-theoretic secure verifiable secret sharing. In *Annual international Cryptology Conference (CRYPTO '91).*

[44] Michael O Rabin. 1981. *How To Exchange Secrets with Oblivious Transfer.* Technical Report. https://eprint.iacr.org/2005/187.

[45] Mohammad Saidur Rahman, Payap Sirinam, Nate Mathews, Kantha Girish Gangadhara, and Matthew Wright. 2020. Tik-Tok: The Utility of Packet Timing in Website Fingerprinting Attacks. *Proceedings on Privacy Enhancing Technologies* 2020, 3 (2020).

[46] Lee Rainie and Mary Madden. 2015. Americans' privacy strategies post-Snowden. (2015). Last accessed: June 1, 2024. DOI: 20.500.12592/jwvf83.

[47] Vera Rimmer, Davy Preuveneers, Marc Juarez, Tom Van Goethem, and Wouter Joosen. 2018. Automated Website Fingerprinting Through Deep Learning. In *25th Annual Network and Distributed System Security Symposium.*

[48] Stefan Savage. 2018. Lawful Device Access without Mass Surveillance Risk: A Technical Design Discussion. In *Proceedings of the 25th ACM SIGSAC Conference on Computer and Communications Security (CCS '18).* DOI: 10.1145/3243734.3243758.

[49] Meng Shen, Kexin Ji, Zhenbo Gao, Qi Li, Liehuang Zhu, and Ke Xu. 2023. Subverting website fingerprinting defenses with robust traffic representation. In *32nd USENIX Security Symposium (USENIX Sec '23).*

[50] Victor Shoup. 2000. Practical Threshold Signatures. In *International Conference on the Theory and Applications of Cryptographic Techniques (Eurocrypt '00).*

[51] Payap Sirinam, Nate Mathews, Mohammad Saidur Rahman, and Matthew Wright. 2019. Triplet fingerprinting: More practical and portable website fingerprinting with n-shot learning. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS '19).*

[52] David Smith. 2019. Trump condemned for tweets pointing to name of Ukraine whistleblower. https://www.theguardian.com/us-news/2019/dec/27/trump-ukraine-whistleblower-president.

[53] The Tor Project. 2024. Tor | Metrics. Last accessed: February 8, 2024. https://metrics.torproject.org/research.html.

[54] The Tor Project, Inc. 2024. Who uses Tor? https://www.torproject.org/about/torusers.html.en Last accessed: June 5, 2024.

[55] Sam Thielman. 2016. Apple v the FBI: what's the beef, how did we get here and what's at stake? https://www.theguardian.com/technology/2016/feb/20/apple-fbi-iphone-explainer-san-bernardino Last accessed: June 5, 2024.

[56] Leon Twenning and Harald Baier. 2024. Towards arbitrating in a dispute - on responsible usage of client-side perceptual hashing against illegal content distribution. In *European Interdisciplinary Cybersecurity Conference (EICC '24)*.

[57] Tao Wang, Xiang Cai, Rishab Nithyanand, Rob Johnson, and Ian Goldberg. 2014. Effective attacks and provable defenses for website fingerprinting. In *23rd USENIX Security Symposium (USENIX Sec' 14)*.

[58] Philipp Winter, Anne Edmundson, Laura M Roberts, Agnieszka Dutkowska-Żuk, Marshini Chetty, and Nick Feamster. 2018. How do tor users interact with onion services?. In *27th USENIX Security Symposium (USENIX Security '18)*. 411–428.

[59] Gavin Wood et al. 2019. Ethereum: A Secure Decentralised Generalised Transaction Ledger. *Ethereum Project Yellow Paper* (2019). https://ethereum.github.io/yellowpaper/paper.pdf.

[60] Charles Wright and Mayank Varia. 2018. Crypto Crumple Zones: Enabling Limited Access Without Mass Surveillance. In *IEEE European Symposium on Security and Privacy (EuroS&P '18)*. DOI: 10.1109/EuroSP.2018.00028.

A Integration into Tor's Handshake

We now discuss how Seldom integrates its oblivious authentication into Tor's circuit establishment. Figure 11 depicts the extensions made to Tor's circuit establishment. The data exchange of the original Tor handshake is compiled in the cell types, which are indicated in capital letters. Seldom extends the relevant cell types, *i.e.*, Create2, Extend2, and Extended2 cells with the data necessary for the oblivious authentication. The clients and relays append the IP certificates or encrypted identities to Create2 cells. This data, however, does not always fit in a single cell. Therefore, the data is fragmented, and the first fragment is appended to the Create2 cell. All further fragments are sent in the newly added Create2_Addata cells. Seldom dispatches these cells immediately after transmitting Create2 cells. As soon as a relay receives the Create2 cell (not necessarily all Create2_Addata cells), it executes the Tor handshake and sends a Created2 cell back. After a relay has sent all Create2_Addata cells, it sends a standard Extended2 cell back to the client to inform it about the successful circuit extension. Once all Create2_Addata have been received by the extending relays, they perform all checks as explained in Section 5. Overall, Seldom's oblivious authentication handshake can thus be integrated seamlessly into Tor's circuit extension without introducing any additional round trips.

B Opaquely Retrieving IP Certificates

The simple form of retrieving temporal IP certificates is not feasible if the network operator does not tolerate the usage of Seldom, *e.g.*, how the Chinese Firewall blocks regular Tor traffic [21]. Certificate requests are easily interceptable and blockable, or worse, the requester could be prosecuted. Therefore, we discuss in the following how someone who wishes to use Seldom could securely request IP certificates under these circumstances. In the past, much effort was invested into making Tor traffic opaque to traffic analysis. Similarly to how bridge relays work, CAs and clients can communicate in a way that does not resemble classical certificate requests. For this, either the CA needs access to enough different IP addresses that not all of them can be blocked simultaneously, or the CA can accept certificate requests through proxies. The process to request certificates in the latter case works as follows. Because the CA cannot verify the origin of a request, the host of the proxy server could request certificates for whatever IP address they desire. Therefore, the certificate itself is not distributed over the same proxy as the request. Instead, the proxy is only informed about other servers that will host shares of the certificate and communicates this to the requesting client. The client can then establish covert connections to these servers and retrieve shares of its certificate. Each of these new proxies verifies the IP address of the client, which ensures that certificates are only created for the user controlling the certified IP address. A single honest proxy suffices to ensure correctness and also identify misbehavior of other proxies.

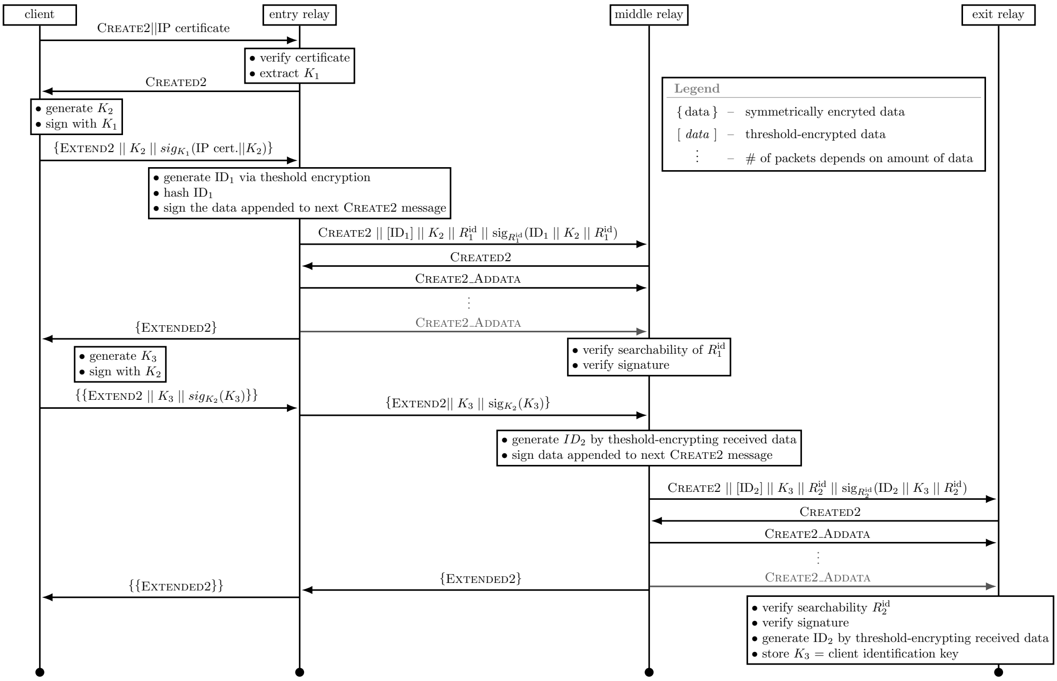

client
entry relay
middle relay
exit relay
CREATE2||IP certificate
• verify certificate
• extract K_1
CREATED2
Legend
{ data } – symmetrically encryted data
[data] – threshold-encrypted data
⋮ – # of packets depends on amount of data
• generate K_2
• sign with K_1
$\{$EXTEND2 $|| K_2 || sig_{K_1}$(IP cert.$||K_2)\}$
• generate ID_1 via theshold encryption
• hash ID_1
• sign the data appended to next CREATE2 message
CREATE2 $|| [\mathrm{ID}_1] || K_2 || R_1^{\mathrm{id}} || \mathrm{sig}_{R_1^{\mathrm{id}}}(\mathrm{ID}_1 || K_2 || R_1^{\mathrm{id}})$
CREATED2
CREATE2_ADDATA
⋮
{EXTENDED2}
CREATE2_ADDATA
• verify searchability of R_1^{id}
• verify signature
• generate K_3
• sign with K_2
$\{\{$EXTEND2 $|| K_3 || sig_{K_2}(K_3)\}\}$
$\{$EXTEND2$|| K_3 || \mathrm{sig}_{K_2}(K_3)\}$
• generate ID_2 by theshold-encrypting received data
• sign data appended to next CREATE2 message
CREATE2 $|| [\mathrm{ID}_2] || K_3 || R_2^{\mathrm{id}} || \mathrm{sig}_{R_2^{\mathrm{id}}}(\mathrm{ID}_2 || K_3 || R_2^{\mathrm{id}})$
CREATED2
CREATE2_ADDATA
⋮
{{EXTENDED2}}
{EXTENDED2}
CREATE2_ADDATA
• verify searchability R_2^{id}
• verify signature
• generate ID_2 by threshold-encrypting received data
• store K_3 = client identification key

Fig. 11. SELDOM's oblivious authentication protocol integrates nicely into Tor's circuit establishment without causing additional round trips.

C Hybrid Threshold Encryption

To efficiently implement SELDOM, we designed and implemented a hybrid threshold-encryption scheme based on RSA and AES, which is published as the thRSAhold Python library.[3] Our thRSAhold library is compatible with standard RSA and AES implementations, such that, *e.g.*, *openssl* can be used for fast encryption, and the slower Python code is only used for deanonymizations.

Our threshold encryption closely follows Shoup's threshold RSA signatures [50]. All plaintexts are padded to be at least as long as the modulus of the public RSA key to prevent common attacks against RSA. If the plaintext does not fit into a single RSA ciphertext, a random AES key is generated and prepended to the plaintext. This key and the beginning of the original plaintext message are threshold-encrypted, while the remaining plaintext is encrypted with AES in GCM mode to ensure confidentiality *and* data authenticity.

During decryption, the first part of the message, clearly defined by the size of the public RSA modulus, is threshold decrypted by the consortium. Only an entity that collects enough decryption shares can then reconstruct the first part of the plaintext, which only requires knowledge about the public encryption key. Thus, the requesting LEA can decrypt the identity of the suspicious user in the final deanonymization step, without revealing this information to the consortium itself. If the ciphertext is longer than the RSA modulus, the first 32 B of the decrypted plaintext are interpreted as an AES key to decrypt the remaining ciphertext. The key can then be discarded afterward. Finally, the padding is removed from the plaintext to reveal the original plaintext.

[3] thRSAhold is available on PyPi and on GitHub at https://github.com/eric-wagner/thRSAhold.

D Consortium Smart Contract

Listing 1 shows an exemplary Solidity smart contract to manage a static consortium of five members with a basic majority vote to accept a deanonymization request. The highlighted lines of code show the variables and events that are revealed to the involved exit node to prove that the deanonymization request for a specific flow record has been accepted and to the public (either in real time or through delayed disclosure).

```
pragma solidity >=0.8;

contract Voting {

  Case[] private cases;
  uint256 next_case_id=0;
  uint256 constant VOTING_THRESHOLD = 3;
  address[] consortium = [
    0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 , 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2 ,
    0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db , 0x78731D3Ca6b7E34aC0F824c42a7cC18A495cabaB ,
    0x617F2E2fD72FD9D5503197092aC168c91465E7f2
  ];

  event Accepted(uint256 indexed index , bytes32 key);
  event NewCase(uint256 indexed index , bytes32 key);

  struct Case{
    bytes32 hash;
    int256 accepted; // 32 byte such that the variable can be revealed individually
    mapping(address=>bool) votes_by_consortium;
    string reasoning;
  }

  function request_deanonymization(bytes32 hash , string memory reasoning) public returns
       ↪ (uint256 id){
    Case storage c = cases.push();
    c.hash = hash;
    for(uint i=0; i<consortium.length; i++){
        c.votes_by_consortium[consortium[i]] = false;
    }
    c.accepted = -1;
    c.reasoning = reasoning;
    emit NewCase(next_case_id, cases[id].hash);

    return next_case_id++;
  }

  function vote_in_favor(uint256 id) public returns (bool){
    cases[id].votes_by_consortium[msg.sender]=true; // track all votess

    uint256 count = 0;
    for(uint i=0; i<consortium.length; i++){
      if(cases[id].votes_by_consortium[consortium[i]]){ // only count votes by members
        count++;
      }
    }

    if(count>=VOTING_THRESHOLD && cases[id].accepted!=0){
      cases[id].accepted = 0;
      emit Accepted( id, cases[id].hash );
    }

    return cases[id].accepted==0;
  }
}
```

Listing 1. A minimal smart contract to manage a fixed consortium of size 5 with an threshold to accept a deanonymization request of 3. The information leaked in real time to the public is highlighted in orange, the information revealed to convince exit nodes of a successful vote are highlighted in purple, and the additional information revealed during the delayed disclosure to the public is highlighted in green.