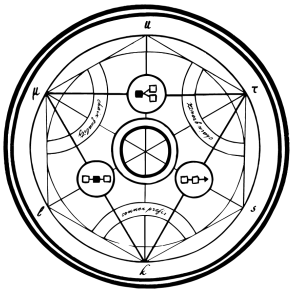
Common Prefix



Evaluation of private networks for Celestia

Pyrros Chaidos, Dimitris Karakostas

23 January 2023

Introduction

The goal of this draft is to propose a solution that resolves the selective disclosure attack against Celestia's by approximating its enhanced network model (§ 5.4).

Model

Clients

We assume *n* clients in total. Each client performs queries about shares of blocks. The choice of queries is done locally, i.e., there is no coordination between clients.

The clients do not act as relays for query replies. For the context of this work, we assume that queries are only being made with regard to a recently disseminated block for which metadata has recently been made available[1].

Block producers

Celestia's block time is 30 seconds.

Attacker

We denote the attacker by A.

A aims at breaking the security (safety and/or liveness) of a specific client-victim, denoted by V.

A has the following minimum capabilities:

- It controls the peer that receives requests from clients
- It knows the IP addresses of all peers

A *does not* have the following capabilities:

- It does not observe the peer's (victim's) *decrypted* traffic
- It does not control the entire communication path with the peer. Specifically:
    - If the peer uses a VPN, A does not control the VPN service.

[1] For historical blocks, a client's progress through the history of the chain will significantly reduce the potential anonymity set, potentially to a single node. At the same time, the attacks we intend to repel seem to be relevant mostly with regard to fresh blocks. We also consider the internals of the DHT layer to be orthogonal to this work.

    - If the peer uses a mixnet, A does not control the entire mixnet.
    - If the peer uses a standard internet connection, A does not control the Internet switches/routers.
- It cannot *eclipse* the peer, i.e., it does not control (entirely) the peer's Internet connections.

In addition to the above, we consider the following adversarial models as relevant:

1. **Passive network attacker.** A can eavesdrop V's encrypted traffic.
2. **Man-in-the-Middle.** A can delay V's packets by up to Δ_A and can alter V's *unencrypted* traffic.
3. **Relay attacker.** A can control t (out of n) mixnet relays.[2]

We note that the third type (relay attacker) is orthogonal to the first two (so A can be a combination of 1-3 or 2-3).

Synchronicity

Clock

We assume that all peers have a Δ-synchronized global clock, i.e., the clocks of any two clients cannot drift by more than Δ.

Depending on the adversarial model, this can be achieved via:

- NTP with Δ=100ms, if A is not of type 2 as above (i.e., it only knows the client's IP and does not control its traffic, but can possibly eavesdrop it) [1]
- NTS with Δ=3s, if A is of type 2 (i.e., it can control the client's encrypted traffic); we suggest chrony as an NTS implementation [2, 3]
- GPS with Δ=1ms, if A can corrupt the NTP & NTS servers to which the peer connects [4]

Latency

We assume a maximum latency of the (vanilla) Internet connection between any two peers of 1 second.

Requirements

We require that given a query for the same, recently disseminated block, an adversarial server cannot tell which user it belongs to, amongst the set of users who recently were made aware of

[2]The number of relays that $\mathcal{A}$ controls depends on the adversarial model. Nonetheless, assuming a few tens of corrupted relays is a reasonable assumption for any realistically strong adversary.

the block. In the literature this is often described as sender/receiver unlinkability or pairwise unlinkability depending on the exact framework used. We do not adopt a single formal definition for this report as we are interested in a broader comparison of different designs.

Tor

Tor offers a client-server architecture.

The client initiates the connection by choosing a path of 3 relays in the Tor network.

If the server is in the clearnet, the Tor path between the client and the server consists of the 3 client-chosen hops. For every connection, the server knows *only* the IP address of the path's exit node (i.e., not the ones of the internal relays).

If the server is in Tor (i.e., is a hidden service), the server also chooses 3 relays and the path between the client and server consists of: i) the 3 client-chosen hops; ii) a rendezvous node; iii) the 3 server-chosen hops. In this case, neither the client nor the server know the internal nodes of the other's path.

The median circuit build time is ~1.3 seconds. The round-trip latency to a public server is roughly 1400 ms; assuming a standard Internet (e.g., home) connection, which observes between 40-100ms RTT latency, Tor adds a latency of ~1300ms.

Note: In Celestia, one share is 512 bytes, and the corresponding proof comprises two merkle paths of 8 hashes each. This produces a sum[3] of 1024 bytes for a block together with its prof. To download approx. 50 shares[4] with proofs over Tor (~50KB) requires approx. between 1-3 seconds. This should be taken into account, in conjunction with the timeout rate of Tor, which is ~3%.

The number of exit nodes is ~1500.

Solution 1

The client opens a circuit to the server. For every block, it sends the queries over the circuit to the server, using a single message (which contains all the queries).

Hazards

[3] We assume the roots have been obtained beforehand, as suggested in § 5.2.
[4] According to the security analysis report, a client should request at least 50 shares, s.t. the probability of a selective disclosure attack being successful drops below 10^{-6}.

Since all queries are bundled in a single message, the minimum attacker A can link the queries to a single origin. Following, A can pick a client (i.e., an origin) at random and perform the selective disclosure attack on it. In this setting, A breaks the safety of *one* client and, with probability *1/n*, this client is V (equivalently, A is successful in attacking V after *n* tries).

Solution 2

The client opens a circuit to the server. For every block, it sends a single message per query in an asynchronous manner (i.e., with a time difference between each query in the order of μs), as soon as possible. The client can open a separate TCP/IP connection for each query over the established circuit, with the intention of mixing queries by user choosing the same exit node.

Hazards

The minimum attacker A can do the following.

First, it can link all queries to the same exit node, i.e., it can (with high probability)[5] cluster all queries to their origin, even without knowing which client corresponds to which exit node. This enables the same attack as solution 1 above (i.e., breaking safety with probability *1/n*).

Second, A can cluster queries based on the timestamp when it received them. The difference in the receiving timestamps depends on Tor's latency. So, two queries that were sent with a few μs difference can be received with a time difference up to ~700ms (half the max round-trip latency). Therefore, the queries of any two clients that initiate their querying with a time difference more than ~700ms can be distinguished, even if the same exit node is used. Using this timing mechanism, A can again cluster queries to their origin and perform the safety attack with probability *1/n*.

Third, if the adversary is additionally able to monitor the network, then by selectively delaying its replies to queries, the adversary can attempt to link answered queries to IPs. Whilst this does not allow the adversary to change its answer to the particular query it can be used to (1) inform answers to subsequent ones and (2) temporally isolate peers by enforcing delays.

Solution 3

For every query (of every block), the client opens a (separate) circuit to the server and sends a message for the query over this circuit as soon as the circuit is ready.

[5] Assuming the Tor client picks an exit node uniformly at random, the probability that no two clients pick the same exit node is $\Pi_{i=0}^{k-1}(1500-i) / 1500^k$ where k is the number of clients.

Hazards

Using a single circuit per query now prevents the minimum attacker A from directly clustering queries to exit nodes (and thus their origin). However, A can again perform the timing attack as in solution 2.

Specifically, the first option is that the client builds all circuits in parallel. Building a circuit takes between 100-2000ms, whereas a circuit's latency ranges between 50-2000ms. As a result, A can again distinguish the messages of any two clients that are desynchronized with a time difference more than approx. *4 seconds*.

The second option is that the client builds the circuits serially. However, assuming *k* queries, this introduces a latency of up to *k*2 seconds* (i.e., to build *k* circuits); e.g., if *k=50*, this latency is approx. *2 minutes*.

However, the passive network attacker A' can use timing to deanonymize V's queries with better probability. Specifically, A' can typically identify when each of the client's circuits is built. As a result, assuming that V does not use Tor for any other operations within the same time period, A knows the exact time when each of V's queries is issued. Following, observing the received requests on the server and taking into account the latency of ~700ms, it can identify V's queries with high probability, assuming other clients don't send their queries within 700ms (Tor's latency) before and after V's queries.

Note: The above time frames assume the generic timing measurements for Tor. However, by observing V's traffic, the passive network attacker A can be informed about V's specific parameters. For instance, after consistently observing small circuit build times (e.g., in the order of 300ms), A can conjecture that the latency of V's messages is low, so A can reduce the expected latency (from the average of 700ms) when performing its timing attack.

Solution 4

For every query (of every block), the client opens a circuit to the server and sends a message for the query over this circuit, after introducing an artificial delay chosen uniformly at random between $[0, t_d]$ seconds.

Hazards

In this setting, A can perform the same attacks as in solution 3. Although, on the face of it, it is unclear how much artificial delays benefit anonymity, they possibly increase the desynchronization assumptions about the clients.

Specifically, the minimal attacker can distinguish the messages of any two clients that are desynchronized by more than $4+t_d$ *seconds*.

However, the passive network attacker can observe the exact timestamp when V sends (encrypted) messages, so with high probability it can deanonymize V's messages if no other client sends queries within 700ms of V's messages.

Solution 5

The client behaves as in solution 4, but also introduces dummy traffic. Specifically, for every circuit towards the server, the client opens a second circuit with the same entry node and sends messages over that circuit to other services, after introducing artificial delay using the same parameters as the delay used for Celestia queries.

Hazards

Here, the minimal attacker behaves the same as in Solution 4.

However, the passive network attacker cannot pinpoint the exact time when V sends its Celestia queries. However, depending on the (randomized) values of the dummy traffic's delay, A can identify a small time frame within which the Celestia queries were sent (i.e., the time frame within which all encrypted messages, including the dummy traffic and the Celestia messages, were sent). Therefore, the larger the dummy traffic, the less accurate A's timing estimates are, although this would also increase the client's bandwidth requirements.

Solution 6

The client behaves as in solution 4, but also uses Snowflake[6]. Briefly, Snowflake masks Tor traffic by relaying it through Tor nodes via WebRTC and makes it indistinguishable from other WebRTC services, such that a passive eavesdropper (e.g., the ISP) cannot identify that the client is using Tor.

Hazards

The minimal attacker again behaves as in Solution 4.

[6] For more information see https://snowflake.torproject.org and https://arxiv.org/pdf/2008.03254.pdf.

However, the passive network attacker can no longer pinpoint the client's Tor traffic and it cannot identify when the client uses Tor. Therefore, this attacker now has the same capabilities as the minimal attacker.

VPN

We assume that each client uses a VPN service of their choice.

The server receives the peer's queries via the VPN, i.e., it only learns the VPN exit node's IP (even though the client knows the server's IP).

A commercial VPN adds a latency of ~300ms, while some VPN services might increase latency (or slow down the connection by decreasing bandwidth) by 10-20%. Unless a client uses multiple VPNs or if multiple clients opt to use the same VPN servers, clustering is very easy even for the minimal adversary. As such, the ease of availability of VPNs cannot be leveraged as they need to be used in a non-standard way.

Loopix (NYM)

Loopix is a packet-based privacy solution that uses cover traffic and built-in delays to maintain privacy even against attackers with wide scale monitoring capabilities (Global passive adversaries), extended and implemented by NYM. Latency is quoted to be as low as 3 sec.

One obstacle to a Loopix solution is the requirement for a known rate of payload messages from the sender. Whilst this rate can be easily determined with regards to caught-up clients who are following in near real-time, it is less predictable with regards to clients who are catching up after a long offline period. Furthermore, since the communication is packet-based, senders need to include single-use reply blocks, to be used by the receiver for replies. Given the size difference between queries and server replies, it may also be appropriate to partition the communication network in the two directions, otherwise padding and/or splitting packets becomes uneconomical. A final concern is that Loopix models traffic as poisson whereas actual client traffic is for the most part contained in small, regularly-spaced bursts. This is possible to handle as outgoing traffic is buffered, but implies an increase in real world latency. Two different implementations are available: one by the authors and an independent one here. The NYM project also implements a commercial service based on Loopix.

Streams

Streams is a packet-based privacy solution with a split approach to mixing and processing messages. In each step, a Funnel node serves to provide a single point where every pair of messages may be (in principle) swapped, whilst processing nodes share the cost of applying public-key operations needed to unwrap each message. Thus, two messages belonging to the same round can be assumed to be perfectly mixed if there exists one step where the funnel node as well as the processing nodes for each message are honest. For system parameters provided in the paper (32 hops) latency is estimated at ca. 30 sec.

As Streams is packet-based, similar concerns apply compared to Loopix. Furthermore, the funnel node for each round is a single point of failure, and the costs of running funnel nodes scale linearly with the amount of traffic (by design, each funnel must handle the entirety of traffic). A prototype implementation is provided by the authors via Google Drive.

cMIX

cMix is a message-based mixnet utilizing a tailored encryption scheme that aims to increase performance by limiting expensive operations to a preprocessing phase. Whilst performant, its design implies that message sizes are correlated with group element sizes, requiring a limited redesign for the return path.

HOPR

HOPR [whitepaper] is a privacy service that implements a mixnet with similar characteristics to Loopix (dummy traffic, sphinx packet encapsulation but not delays and route limitations). The publicly available documentation is vague wrt technical details. A report by Loopix coauthor A. Piotrowska comparing simulations of Loopix, HOPR and Elixxir is available.

Solutions not covered

- Atom (high latency)
- Riposte (high latency)
- Express (low throughput)
- Dandelion (one-to-many, one-way communication)

Future work

In this report we provide a high level overview of the available network anonymity solutions. Specifically, we provide a series of candidate solutions, based on Tor, each protecting against a wider range of attacks and adversaries with particular capabilities, at possible tradeoffs of latency/bandwidth. Also we review alternatives based on VPN, Loopix (and its implementation in NYM), Streams, cMix, and HOPR.

This report does not provide a rigorous, scientific analysis of these solutions' anonymity guarantees w.r.t. Celestia's requirements. Such analysis would require significant effort to express Celestia's setting in a formal model, describe each solution in the same model and, using cryptographic tools (such as the Universal Composability framework) provide proofs of security when possible (or describe specific attacks in a more rigorous manner than presented here). Due to the time effort needed to do this, we leave it for future work.

Another aspect that we do not touch in this report is information discovery. Specifically, Celestia relies on IPFS's Kademlia DHT for data discovery. It is unclear how much information is leaked to an adversary by this DHT implementation and how this affects the data retrieval (w.r.t. the selective disclosure attack and the anonymity process covered in this report). Furthermore, it is unclear if an adversary can arrange for a (malicious) block to only become known to a small set of clients. Since data discovery is orthogonal to this report's scope and due to the time commitment for analyzing the DHT implementation, we defer such analysis to future work.

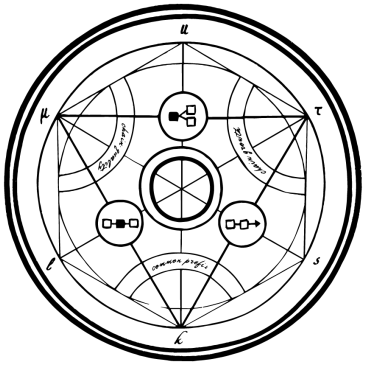
Further reading

- https://github.com/Attacks-on-Tor/Attacks-on-Tor
- https://arxiv.org/pdf/2009.13018.pdf
- [2011.08536] SoK on Performance Bounds in Anonymous Communication
- https://eprint.iacr.org/2017/954.pdf

About Common Prefix

Common Prefix is a blockchain research, development, and consulting company consisting of a small number of scientists and engineers specializing in many aspects of blockchain science. We work with industry partners who are looking to advance the state-of-the-art in our field to help them analyze and design simple but rigorous protocols from first principles, with provable security in mind.

Our consulting and audits pertain to theoretical cryptographic protocol analyses as well as the pragmatic auditing of implementations in both core consensus technologies and application layer smart contracts.



https://www.commonprefix.com