



# Opacity Network: Trust but Verify



Vinayak Kurup · [Follow](#)

12 min read · Jul 9, 2024



1



*The first in a series of EV3 Research technical explorations into how cryptographic advancements are reinventing networks and tearing down Web2 walled gardens.*

Here's a question to think about: when logging into your bank account, how do you know that what is appearing on your screen hasn't been compromised? The complex underpinnings of how we access, transact, and engage on the internet are easy to forget about when they happen seamlessly in the background. But offering traditional web2 services augmented by web3 functionality requires solving network communication issues — the protocols we forget about that run in the background.

Over 95% of internet traffic leverages the HTTPS protocol—secured by Transport Layer Security or TLS—to cryptographically ensure that communications between a client and server cannot be tampered with or snooped on. TLS is generalizable and used to secure communications protocols like SMTP (email), FTP (file transfers), and VOIP (real-time audio).

Designed for secure point-to-point communications between exactly *two* counterparties, TLS loses its attractive security properties when third parties are introduced. For example, third parties have a hard time verifying messages secured by TLS because both counterparties share the symmetric key and can unilaterally falsify content (more on this below). To convince a third party that a TLS transcript is correct, therefore, involves ensuring that it was not modified to fit some predetermined criteria.

If third parties could verify TLS-encrypted messages — through zkTLS solutions — opportunities would arise for open networks and new methods of trust authentication. Crucially, this could occur without the historical reliance on and power vested in dominant web2 corporations. Business models based on hoarding private user data and aggregating proprietary social or reputation graphs would be disintermediated. Instead, a new class of business models would use cross-platform composability to enable new forms of collaboration at global scale, fundamentally disrupting the balance of powers in industries like online payments, influencer marketing, and identity verification in favor of end-users (and at the expense of centralized platforms). As on-chain ecosystems enter the mainstream, zkTLS will unlock previously unimaginable flexibility and optionality.

Perhaps most importantly, we believe that zkTLS will empower users to regain ownership of their own data, generate and transfer their own authentications for a fraction of the price, and allow companies to build upon previously inaccessible networks. The best part: this will be done without the need for an API or any buy-in from the companies themselves. zkTLS is a clear solution that disintermediates walled gardens and embodies the ownership of the web3 movement.

Given the scale of value creation by social media giants alone — not to mention gig economy apps and tech companies in health care — on top of private social/reputation graphs over the past decade, we believe such a scenario could drive hundreds of billions of value transfer to new, open networks.

Recent advancements in cryptography and secure computation have made third-party verification of TLS-encrypted messages a possibility in the near-term, though many challenges remain. This is the heart of the issue that cryptographers at Opacity, Clique, Reclaim, and other TLS projects are working on. Let's take a look at the opportunities, structures, and possibilities of the leading on-chain TLS projects.

## **SSL/TLS: A Brief Guide**

When communicating with a server over the internet, it is imperative that a user can trust the validity of queries from the server. Clients use the HTTP protocol to fetch resources such as HTML documents from the server and reconstruct them on screen. SSL, or Secure Socket Layer, is an encryption-based Internet security protocol and the predecessor to modern TLS encryption used today. When a website implements SSL/TLS, it has HTTPS in the URL, which we are often advised to look for, rather than just HTTP.

Here's a good mental model for how SSL/TLS works. The server has a private value and the client has a private value. The client and server use these private values to arrive at a common shared secret used to derive a symmetric key that both the client and server possess. With this symmetric key, both the client and server can encrypt and decrypt each other's messages to the exclusion of anyone else. [1]

This setup, however, is susceptible to a man-in-the-middle attack, where a malicious third party intercepts packets between the client and server, impersonates each party, and can fabricate messages that look legitimate. [2] To overcome this flaw, Netscape implemented the Certificate of Authority in 1999, where the public key of the server is signed by a Certificate authority — a handful of predetermined actors who validate the identity of entities, such as websites. Now, before engaging with the server to create a shared secret, the client requests the server's certificate signature to ensure they are communicating with the right entity.

When convincing a third party of a valid transcript, however, things get more tricky. The challenge is that since both parties have the *same* secret key, the user is capable of fabricating a transcript and signing the server's responses as well, creating the appearance of a valid claim to a third party. In order to trust the provenance of a claim, then, we must complicate the normal network request.

## **The Solutions**

Symmetric keys are useful for a user and server to verify each other's claims, but because the user possesses the same key as the server, an entire transcript can be forged by the user, allowing the user to claim anything they choose. There are a few different current and proposed solutions in the space, elaborated below.

**Using a trusted execution environment:** Clique, a network that closed its recent \$8m Series A, focuses on building out a trusted execution environment (TEE) network for blockchain networks as a whole, but one of the major offerings is a zkTLS solution that utilizes TEE.

A TEE is a secure area in the main processor that is entirely encrypted and cannot be directly opened by even its owner. Owners interact with the TEE through designated and secure interfaces that allow the running of secure authorized applications called trusted applications. The TEE then returns a result and a signature attesting that the result comes from a particular program. Theoretically, this makes the system tamper-proof, even against those running TEEs in the network.

However, as Dr. Andrew Miller at University of Illinois has been demonstrating for years by breaking TEEs (primarily Intel SGX), they aren't a perfect solution. Any large-scale solution relying solely on TEEs has to contend with this challenge, especially when involved with smart contracts in more high-stakes situations, like KYC (Know Your Customer) regulation.

**Using a proxy witness:** Another possible solution is to use a proxy witness that acts as a middleman between the client and server and attests to the accuracy of the transcript. The Reclaim Protocol, whose white paper was published late last year, is the most popular example of this strategy.

A trivial solution is to reveal to a third party, called the *attestor*, the user's TLS private key. This allows the attestor to decrypt the encrypted request, check the correctness of the request, the correctness of the corresponding response, and attest if deemed correct. The challenge, however, is that this approach also reveals private information such as authentication tokens and cookies to the attestor, which gives the attestor login access to the user's account.

The Reclaim Protocol solves this using a three-part construction that involves a key-upgrade mechanism. With only one attestor, however, the collusion problem once again rears its ugly head. If the user and attestor

collude, then anything can be signed. To overcome this, Reclaim includes a subset selection of validators to help randomize and prevent this. However, since the attestor in the Reclaim protocol is responsible for conducting parts of the TLS handshake, their IP address is known by the server. Therefore, it is possible that servers could block IP addresses associated with attestors at scale, reducing the viability of such a system.

### **Using multi-party computation (MPC): TLSNotary and Opacity**

In MPC schemes, two or more parties jointly compute a function over their inputs while keeping the inputs private. The parties learn the result of the function at the end, but none of the participants learns anything about the other's inputs. In the case of TLS, an MPC node and the client would jointly make the TLS request and generate the transcript such that no one knows the shared secret key until after the session is finalized. Crucially, this approach is undetectable by the target server, which, unlike the Reclaim Protocol, is not being bombarded with requests in a short timespan but only has to respond to one. It is indistinguishable from the client logging in by themselves.

A few challenges arise from the MPC-TLS approach. The Shamir Secret Sharing MPC (SSS-MPC) scheme is very effective in scaling the number of parties that are conducting the computation. Since all parties must collude to reveal the shared secret, the more parties conducting the computation, the harder it is to collude — and the greater our security guarantees can be. SSS-MPC is commonly used in MPC wallets, like those in the Lit Protocol.

SSS-MPC, however, is not a good solution for the zkTLS case because of the type of computation required for TLS. The shared secret is the result of an ECDH (Elliptic Curve Diffie-Hellman) key exchange. To hash in SSS-MPC

without any party being able to reconstruct the input or output is challenging because SSS-MPC is fundamentally an algebraic scheme. To make them harder to break, however, the hash functions used in TLS are not algebraic in nature and rely on binary operations (like XOR). This discrepancy creates a prohibitively high communication overhead that scales factorially as new parties are added.

To get around this, Opacity uses garbled circuits and oblivious transfer (OT) schemes. Unlike arithmetic circuits, garbled circuits are boolean in nature and consist of gates (like AND, OR, NOT) that perform binary operations, making them much more efficient for hashing. OT is a cryptographic technique that allows one party (the sender) to send information to another party (the receiver) in such a way that the sender does not learn which specific piece of information the receiver obtained, and the receiver cannot obtain more information than they are entitled to. By using an OT scheme, parties can select and compute on data inputs without revealing which inputs they are interested in, maintaining the privacy of their choices and inputs throughout computation. For garbled circuits, OT can be used to securely transmit keys for the input wires without revealing which keys were chosen. Garbled circuits and OT schemes, however, scale poorly beyond two parties — many of the optimizations developed for two-party garbled circuit/OT computation are unavailable for MPC with more parties.

TLSNotary is the current MPC TLS solution. The scheme works as follows: the user, called the *prover*, and another party, called a *notary*, engage in an MPC together. The output is then signed by the notary, making the data portable. The user can take the signed data and disclose parts to any application-specific verifier. The challenge here is that the notary **must be trusted**. If the notary and the prover collude in MPC, they can reveal the shared secret key and fabricate any transcript between the user and the

server. While it is possible to have the user generate multiple proofs from many notaries before trusting the result, known as proof by committee, such a system would only work if the user cannot change the value between proofs. We can use the bank example to illustrate this further. If we have a committee of five nodes, all of whom have to agree for the server transcript to be considered valid, and a recurring debit hits the account between the third and fourth nodes conducting the MPC, then the last two nodes will disagree with the first three and may be slashed even though they conducted the protocol correctly. An easy solution might be to have all of the nodes conduct the MPC at once, but most servers will flag this as fraudulent behavior since it looks like multiple login attempts to the same account, all at the same time. For a truly general zkTLS solution, proof by committee doesn't work, since price data (a stock price for example) is changing too quickly to allow for multiple parties to conduct the MPC and determine a value without significant margin.

### **Opacity Network:**

Opacity builds from the existing TLSNotary framework with added safeguards and provisions to reduce the trust assumptions inherent in the protocol. It does this by layering multiple security considerations that disincentivize byzantine behavior and collusion. These include an on-chain verification of a web2 account ID, a commit and reveal scheme, a random sampling of the MPC-network, an on-chain verifiable log of attempts, and a whistleblowing process.

Given that account IDs act as the primary identifiers in a web2 company's database, it is standard practice never to change them. This enables a proof by committee to prove ownership of a web2 account, and Opacity's framework expects users to generate ten identical proofs by different nodes



in order to claim a web2 account. Because this account is now mapped to a unique wallet address, a user cannot try different wallets until they find a node willing to collude. This scheme doesn't work for public data not associated with any account such as a weather feed or public stock price. In these cases, oracles can be limited to a trusted subset or required to put up stake, and the protocol can be engineered to be much stricter with the log of attempts (discussed below).

The commit and reveal scheme to further secure against collusion where a user cryptographically commits to a value before a notary node is selected. If a user claims to have \$1 million in a bank account, they would have to commit to the \$1m *before* a node is chosen. Combined with an on-chain record of a user's attempts to generate a signed transcript, Opacity further disincentivizes users from trying the process repeatedly before finding a node willing to collude. For example, even if a user could find a node to collude with on the third attempt, a smart contract reading the signed transcript will still be able to read the on-chain log and see the previously failed attempts, raising red flags.

Additionally, Opacity nodes are required to run their software within Intel SGX (a TEE). Assuming the TEE is not broken, this alone makes collusion impossible. Unlike Clique, we note Opacity's specific solution as a layering of security considerations beyond TEEs. The plan is working closely with the Eigenlayer team to use Eigenlayer's AVS (actively validated service) whereby nodes will be required to restake 32 stETH and are slashed for improper behavior. And because Eigenlayer uses stETH for its restaking, slashing and redistribution of stake takes place immediately rather than waiting until a withdrawal cooldown period is completed.

While we note it's possible that the breaking of the TEE would allow a node to collude in MPC, Opacity's team is currently focusing on implementing a whistleblowing process whereby any user that can submit proof of a notary acting improperly will receive part of the slashed stake (more on that to come).

## Conclusion

The ability to prove TLS transcripts onchain will unlock entirely new functionality as users will have control over their own data without needing permission from large corporations. While there is work to be done, zkTLS companies promise to enable novel use cases that shift power towards the users. And Opacity's thoughtful method of layering security considerations while using MPC is a novel approach that is already being used to drive value across industries.



# Opacity zkTLS AVS Mainnet Launch

x @builddoneigen



## Appendix

# Opacity zkTLS AVS Mainnet Launch

x @builddoneigen



1. The process by which a user and server obtain symmetrical keys is a Diffie-Hellman key exchange to find a shared point on an elliptical curve. The x-coordinate of this shared secret is then HMAC'd (hash-based message authentication code) by both the client and the server to derive a symmetric key. Since both the client and the server use the SAME set of the symmetric keys, they can now both encrypt/decrypt the transcript.
2. In a man-in-the-middle attack, the malicious party performs individual Diffie-Hellman key exchanges with the client and the server while pretending to be the other party, so that it alone can fabricate messages claiming to be the server to the client and vice versa.
3. Opacity is also adding moving to add support for vector oblivious linear evaluation (VOLE), which allows for efficiently constructing a 1 of N oblivious transfer (OT) vs 1 of 2, thereby reducing network overhead by a factor of one to two orders of magnitude.

Zero Knowledge Proofs

Web3

Tls

Cryptography

Opacitynetwork



**Written by Vinayak Kurup**


1 Follower

Follow



## Recommended from Medium

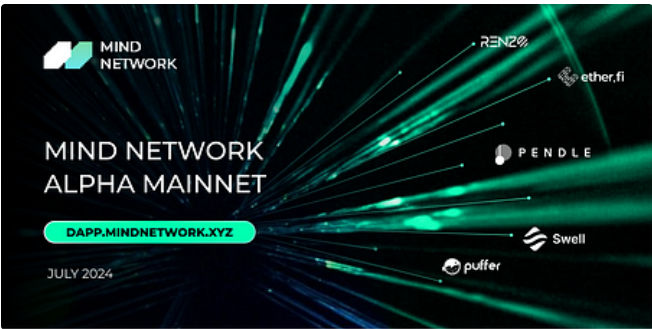



 Vidar Frostbjorn in InfoSec Write-ups

### The Hidden Gem of Pentest Certifications in 2024

Kick-start your penetration test career with this attractive and cheap certification


★ 6d ago 🖱 60 💬 1 



 Mind Network

### Mind Network Alpha Mainnet Launch: A Fully Homomorphic...

Mind Network Alpha Mainnet is now live!  
Check it out: <https://dapp.mindnetwork.xyz/>

Jul 10 🖱 114 💬 3 

## Lists



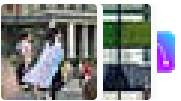
### General Coding Knowledge

20 stories · 1622 saves



### My Kind Of Medium (All-Time Faves)

96 stories · 509 saves



### Generative AI Recommended Reading


52 stories · 1412 saves



### MODERN MARKETING

190 stories · 869 saves




 LearnTheShell

## Reverse Shells: A Practical Guide

In the world of penetration testing, a reverse shell is a crucial concept. It allows an attacke...

★ 3d ago 🖱️ 2





 Ayesha Saddiqa

## \*The Incredible Egg: Nutrition, Health Benefits, and Culinary Uses\*

★ 2d ago 🖱️ 200 💬 3

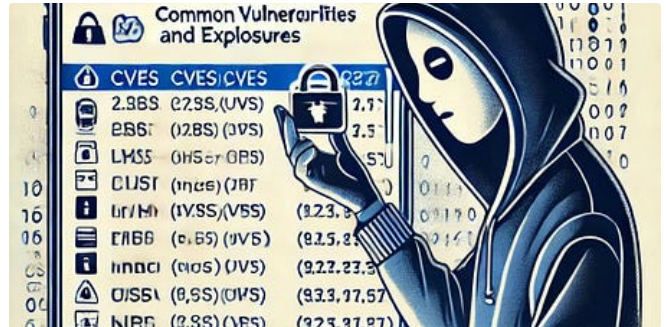


 Aaron Dinin, Ph...  in Entrepreneurship Handbo...

## How Bad Entrepreneurs Reveal Themselves in the First 5 Second...

A few simple words is all it takes to figure out a founder's odds of startup success.

★ Sep 26 🖱️ 3.7K 💬 124



 Jonathan Mondaut

## How ChatGPT Turned Me into a Hacker

Discover how ChatGPT helped me become a hacker, from gathering resources to tackling...

★ Jun 18 🖱️ 1.4K 💬 44



See more recommendations