Proposed Improvements to the Tor Handshake

Akhil Kammila
Mentor: Kyle Hogan

January 2022

Abstract

Tor is the world's largest anonymous communication network. It conceals its users' identities by sending their traffic through three successive Tor relays. To establish connections between users, relays, and destinations, Tor uses a unique two-staged handshake. The first stage is a modified version of TLS 1.2 and the second stage is a fully encrypted exchange of Tor cells. The two-stage process enables both parties to authenticate while masking the differences that the Tor's handshake has from standard TLS. The Tor handshake has multiple shortcomings when compared to widely-used cryptographic protocols like TLS and QUIC. It has high latency that detracts from the user experience and increased complexity that makes maintenance challenging. The first stage of the handshake also only supports TLS 1.2 despite TLS 1.3's release in 2018. Our work presents an analysis of Tor's handshake and proposes improvements. We find messages in the second stage of the Tor handshake that are redundant. Most notably, the responder sends a certificate that is not necessary for authentication. Removing these messages reduces the data transferred in the handshake without compromising the key exchange or authentication. Further, we find that removing backward compatibility from the Tor handshake allows for the trivial use of TLS 1.3 in the first stage. This reduces the round-trips and improves the security of the Tor handshake.

1 Introduction

As technology continues to advance, the internet is becoming increasingly accessible and widely used. Rising internet usage, however, comes with more attackers seeking to steal or spy on users' information. Online anonymity is becoming more desirable as a result. Users are using the Tor network to achieve anonymity, keeping their online activity unlinked from themselves [TOR].

Internet users may desire anonymity for a multitude of reasons. Anonymity provides censorship resistance, which is useful to users who want to bypass online restrictions in their country [TOR]. It also allows users to communicate or voice their opinions without being traced. Whistleblowers, for instance, may choose to access the internet anonymously to send messages while hiding their identity. As public awareness of privacy issues rises, many users seek anonymity simply for a higher level of security than HTTPS provides [TOR].

Tor is an anonymity network that conceals its users' identities using onion routing [SDM04]. When using onion routing, a client's traffic passes through a system of relays called a circuit before reaching the server. Tor uses three relays: a guard node, a middle node, and an exit node. Because the connection passes through multiple relays, no one party knows the identity of both the client and the server. The guard node, for example, knows the identity of the client and the middle node. The middle node knows the identity of the guard node and the exit node. Finally, the exit node knows the identity of the middle node and the server. Even if one relay is malicious, the origin of traffic sent to a server remains hidden.

Every pair of parties in the Tor circuit must have an encrypted and authenticated connection [SDM04]. This means that four secure connections must be established: client to guard node, guard node to middle node, middle node to exit node, and exit node to server. The connections must be secure so that eavesdroppers cannot observe any exchanged data. Tor establishes connections using a modified version of the TLS handshake. The TLS handshake has been proven to be secure [CJJ+19]. To the best of our knowledge, Tor's handshake lacks a formal security analysis.

In this paper, we will propose and evaluate potential changes to the handshake that would improve its latency and security. Our hope is that our changes are implemented and Tor becomes more usable and secure for all users.

2 Background

This section will be focused on the specifics of the Tor handshake protocol. It will explain the purpose and goals of the handshake and why it differs from typical TLS. It will also explain the technical specifications of the handshake, including what packets/cells both parties exchange.

2.1 Goals

Tor's handshake has has four primary goals: authentication, establishing a connection, censorship resistance, and backwards compatibility. Authentication, establishing a connection, and backwards compatibility are standard for any handshake. Censorship resistance is a key reason for why Tor has a unique handshake and does not use TLS alone.

2.1.1 Authentication

Authentication is necessary in any handshake to ensure that the participating parties are who they claim to be. Without authentication, malicious adversaries could hijack the handshake and pretend to be one or both parties in what is called a man in the middle attack [MAST19]. Consider a handshake between client Alice and server Bob, with malicious adversary Eve. Eve could intercept Alice's messages to Bob and send her own messages in their place during the key exchange process. She could similarly intercept Bob's messages to Alice. By pretending to be both parties, Eve can effectively receive all exchanged information.

In a standard handshake, a client connects directly to the server. While mutual authentication is supported by protocols like TLS, the server is typically the only party that authenticates. This is because the server is usually not interested in the client's identity. For instance, a public website allows any internet user to access it, so client authentication is unnecessary [TLS].

Tor differs from the standard because mutual authentication is usually necessary [SWZ16]. The Tor circuit consists of four connections: client to guard node, guard node to middle node, middle node to exit node, and exit node to server. The first connection only requires one-way authentication. This is because similarly to typical TLS, the client's identity is unimportant. The remaining three connections in the circuit, however, require mutual authentication. This is because relays must always authenticate to show that they are part of the Tor network and that they are the client's intended node [SWZ16].

2.1.2 Establishing a connection

Establishing a connection refers to both parties negotiating a shared key. The shared key allows messages to be encrypted but still readable by the intended recipient. Malicious observers are unable to read the encrypted data [TLS].

There are two types of shared keys: asymmetric keys and symmetric keys. Handshakes seek to establish symmetric keys because they allow the fastest form

of encrypted communication. Once a symmetric key is established, a sender encrypts all his messages with the key before sending them and the recipient decrypts all messages with the same key upon receiving them [TLS].

To establish a shared key, both parties must follow a specific key exchange algorithm. They cannot simply send each other the shared key because malicious observers could see it also [TLS]. As the Tor handshake uses a modified version of TLS in its first stage, it negotiates key exchange algorithms with ClientHello ciphersuites just as TLS does [Spe].

2.1.3 Censorship Resistance

Censorship resistance is concerned with hiding the fact that a client is connecting to Tor. If a Tor connection stands out, it can easily be blocked. Anonymous users can be censored in this way. To achieve censorship resistance, Tor users must be indistinguishable from the typical internet user [TOR].

Because Tor users must be unrecognizable, the Tor handshake must look like a typical client's TLS handshake. As was mentioned in 2.1.1, however, Tor often requires mutual authentication - and mutual authentication is uncommon. This is the primary reason why Tor has a unique handshake and does not simply use TLS. If TLS was used, observers could easily see mutual authentication in the handshake and find Tor users.

To hide the mutual authentication, the latest version of Tor uses a unique handshake. It begins with a TLS layer that attempts to mimic a normal internet user. The mutual authentication is hidden in a secondary Tor layer where all data is encrypted.

2.1.4 Backwards Compatibility

A final goal of Tor's handshake is backwards compatibility. The handshake has been ratified multiple times to improve the censorship resistance. Support remains so that relays that are not up-to-date can still support the service. Backwards compatibility has become less necessary, as nearly 0% of relays do not support the latest Tor handshake version [Met].

2.2 Handshake Versions

Tor has three versions, with version 3 being the most recent. The newer versions are progressively less distinguishable from typical TLS, improving the censorship resistance. All three versions are still supported [Spe, line 292]. Tor clients and

relays signal which versions they support with covert channels such as cipher suites and certificate fields. This section will explain each version.

2.2.1 Version 1

Version 1 is the "Certificates-Up-Front" handshake. In this version, a typical TLS handshake is performed. Tor relays send a 2-certificate chain that includes a long-term server identity key and a short-term link key in order to authenticate. As usual, Tor clients do not authenticate and send no certificate [Spe, line 292]. Figure 1) illustrates the handshake.

This handshake is easily recognizable because the 2-certificate chain that relays use to authenticate is unusual. There is also no attempt made at hiding the mutual authentication [Spe, line 292].

2.2.2 Version 2

Version 2 of the handshake uses renegotiation in an attempt to become less detectable. The initiator of the handshake, whether it is a client or a relay, does not send a certificate. The responder always sends a single certificate. This portion of the handshake is similar to typical TLS. To achieve mutual authentication, the initiator renegotiates. In the renegotiation handshake, the parties exchange their true certificates [Spe, line 304]. See Figure 1) for a full handshake illustration.

While Version 2 was effective at the time of its implementation, the usage of renegotiation handshakes drastically dropped. This made the version 2 handshake easy to profile, as renegotiation became unusual [TOR].

2.2.3 Version 3

Version 3, the in-protocol handshake, attempts to make the TLS portion of the handshake as generic as possible by eliminating any renegotiation. Instead, authentication is done using Tor cells after the TLS portion of the handshake is fully complete [Spe, line 314]. Figure 1 shows the full handshake.

The protocol begins with a typical TLS handshake. A TCP connection is first established. The responder (Bob) then sends a certificate to the initiator (Alice) in the TLS portion, and both parties securely agree on a symmetric key [Spe, line 314].

When Bob is a Tor relay, he sends a self-signed certificate with a long-term

identity key. Alice verifies this certificate by checking a live consensus that contains the expected identity key [Spe, line 377].

In the second portion of the handshake, both parties exchange Tor cells and seek to authenticate if they are Tor relays. Both parties first send a VERSIONS cell. This is a variable-length cell which is used to negotiate a commonly supported tor protocol version [Spe, line 330].

Following the VERSIONS cells, Bob sends a CERT cell. This cell contains multiple certificates that contain Bob's long-term and short-term RSA or Ed25519 keys [Spe, line 334]. Upon receiving the CERTs cell, Alice uses the live consensus to check the certificates and authenticate Bob [Spe, line 382].

Bob then sends an AUTH-CHALLENGE cell and a NETINFO cell. The AUTH-CHALLENGE cell contains a randomly generated string that Alice must use if she is authenticating. The NETINFO cell contains time and relay address information [Spe, line 334].

Alice's response depends on if she is a client or a Tor relay. If she is a client, she does not authenticate. She sends a NETINFO cell to conclude the Tor handshake [Spe, line 334].

If Alice is a Tor relay, she must authenticate. She sends a CERT cell and an AUTHENTICATE cell. The AUTHENTICATE cell contains a HMAC of information from the TLS portion of the handshake such as the client random, the server random, and the TLS master secret [Spe, line 334].

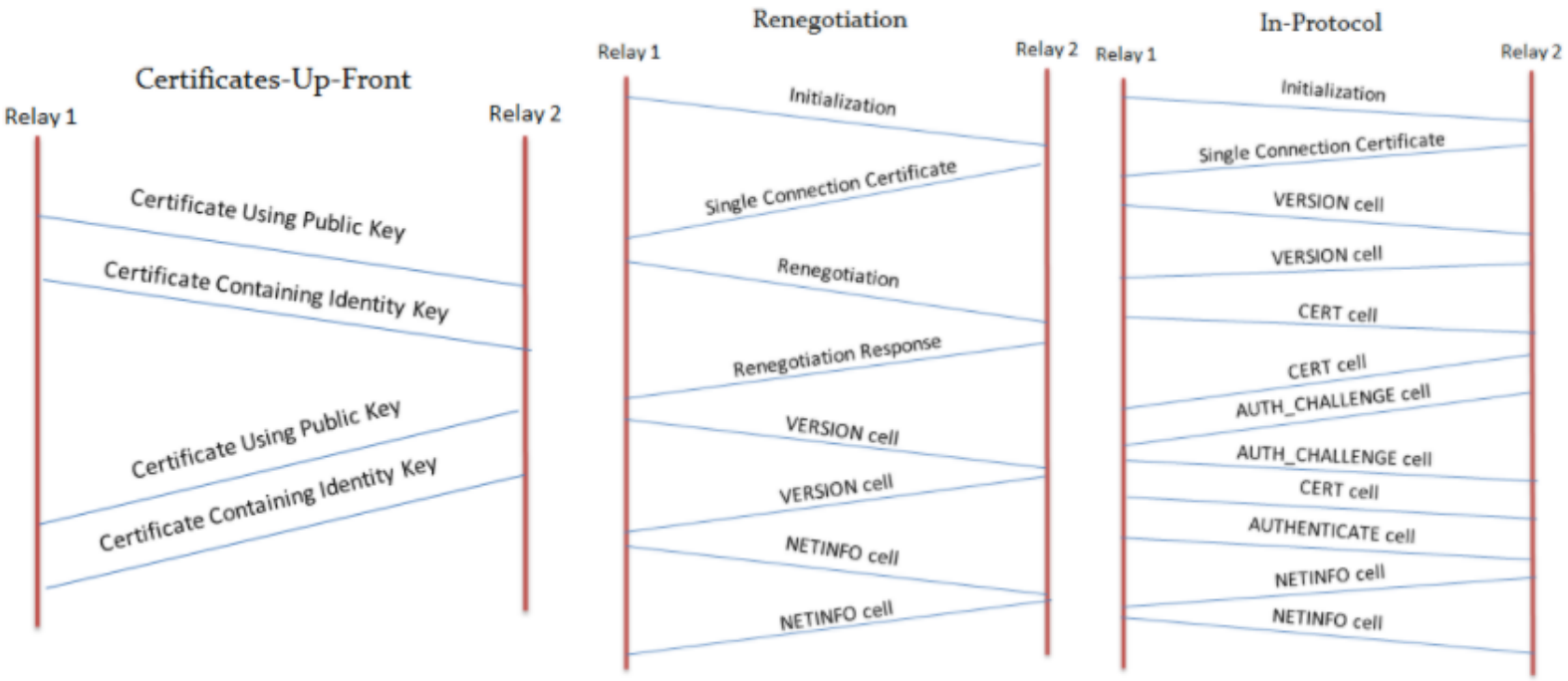

Figure 1: V1, V2, and V3 Handshakes

3 Design

While the Tor handshake has been developed for many years, there remain areas for improvement. Its latency is high due to redundancy and the presence of two stages. Its censorship resistance and security is also weak in v1 and v2. This section will provide a new design for the Tor handshake, which improves latency, censorship resistance, security, and complexity.

3.1 Overview

The new design has two primary changes.

First, a certificate sent by the responder in the second stage of the handshake is removed. The initiator instead uses data from the first stage of the handshake to verify authentication. This change decreases the complexity and the amount of data exchanged.

Second, backwards compatibility to v1 and v2 is removed. Restrictions on the client's ClientHello and the responder's TLS certificate are also removed. The change allows for native TLS to be trivially supported, as modifications for backwards compatibility are no longer needed. The removals do not negatively impact Tor's usability because about 0% of relays support solely v1 and v2 [Met].

3.2 Design

This model is derived from the specification of TLS and Tor, modified to implement the above changes.

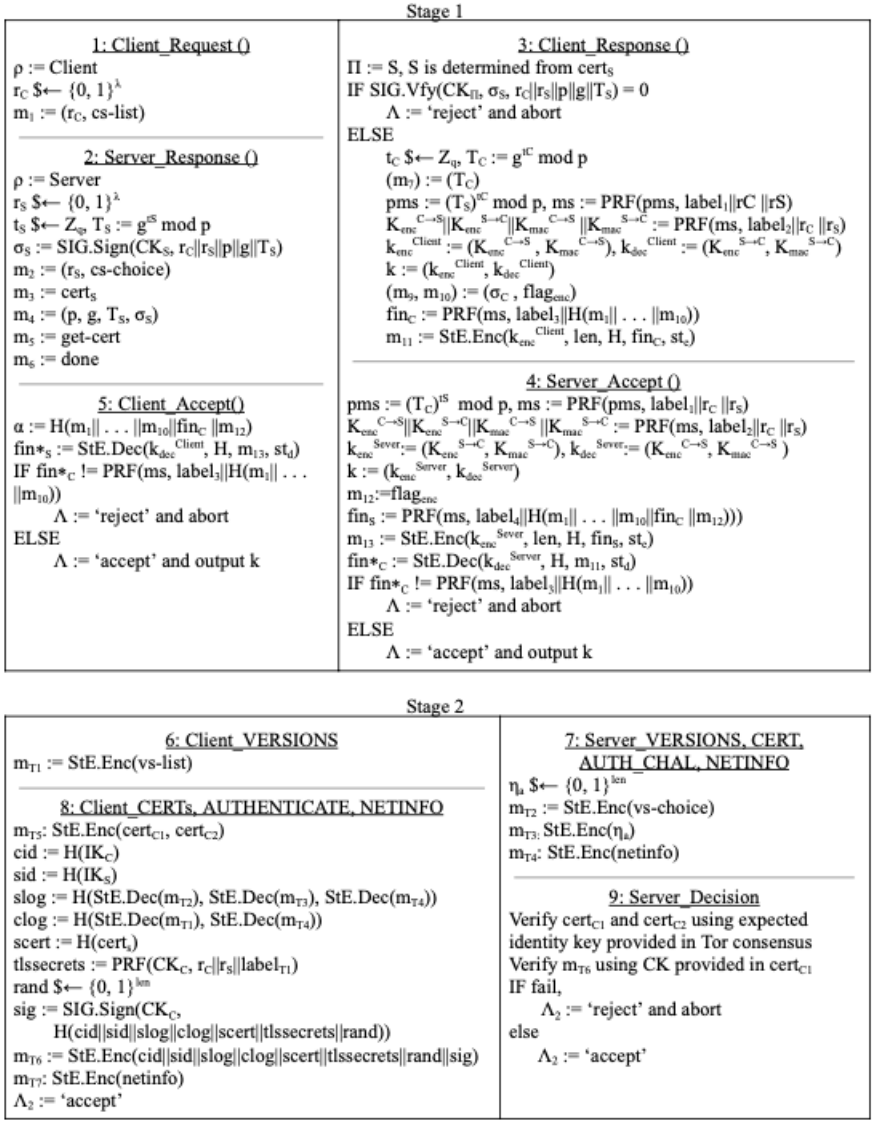

Figure 2: Tor Handshake Improved Design

3.3 Improvements

The design has the following improvements over Tor's current implementation:

- Bandwidth improvement: Less data is sent because the responder no

longer requires a CERT cell in the second stage of the handshake. This data savings is multiplied because there are four connections in each Tor circuit.

- Checks: Removal of v1 and v2 backwards compatibility means that the following checks are no longer performed, thus simplifying the handshake.

    1. In stage 1, the responder no longer checks if the initiator's client hello contains specific cipher suites. Previously, cipher suites were used to indicate a v1 handshake, as was mentioned in Section 2.2.
    2. In stage 1, the initator no longer checks if the responder's certificate is self-signed or has a certain commonname field. Previously, fields in the cert were used to indicate support of v2 (2.2).

- Native TLS Support: The new design supports TLS 1.3, which was previously not usable because it lacked the renegotiation feature needed for v2. It also supports native TLS, rather than modified versions with covert channels for backwards compatibility. This results in the following improvements:

    1. Security. TLS 1.3 has improved security guarantees compared to TLS 1.2, which Tor currently uses. This is because it removes insecure features like SHA-1 and RC4 that TLS 1.2 still supports [1.3].
    2. Latency. Support for TLS 1.3 results in a reduced round-trip time for each of four connections in every tor circuit. This is because TLS 1.3 takes one round-trip-time, while TLS 1.2 takes two [1.3].
    3. Censorship Resistance. Tor handshakes may be identifiable during TLS due to their use of a modified protocol. Supporting native TLS makes the type and order of packets exchanged in stage 1 of Tor indistinguishable from a non-tor TLS handshake.
    4. Future-Proofing. Using a modified version of TLS forces Tor volunteers to create their own updates when new TLS versions come out. Creating this updates is time-consuming, which leads to older versions being used for a significant period of time. For instance, Tor still uses the outdated TLS 1.2 even though TLS 1.3 was released in 2018. Using native TLS will allow Tor to trivially support the most up-to-date one, leading to optimized latency and security.

4 Discussion

This section will discuss the effects of the proposed changes in relation to threats to Tor. It will also discuss potential problems with Tor that remain and future work to address them.

4.1 Threats

This section will cover the two threats to Tor that this paper is concerned with: an adversary that identifies and blocks Tor connections, and an adversary that compromises relationship anonymity for the client and destination.

4.1.1 Adversary Blocking Connections

A primary function of Tor is to resist censorship. It is often used by whistle blowers and people under restrictive governments to bypass restrictions [TOR].

To achieve censorship resistance, Tor's handshake must look similar to the typical internet user's handshake. Otherwise, it is trivial for onlookers to identify Tor users and block their connection. Much of the design behind Tor's handshake is based on this censorship resistance goal. Tor could simply use TLS and send two-certificate chains in the clear instead of having a unique handshake. This is not done, however, because the unusual certificates would make the handshake too easy to identify [SDM04].

4.1.2 Adversary Compromising Anonymity

A Tor circuit consists of a client, an entry node, a middle node, an exit node, and a destination. Each entity in the circuit only knows the two "adjacent" parties that it connects to. For instance, the entry node only knows the identity of the client and the middle node.

The circuit is compromised if both the entry node and exit node are corrupted by a single adversary. This is because the entry node knows the identity of the client, and the exit node knows the identity of the destination. As the relays in each circuit are chosen randomly, Tor's aims to support enough relays such that it is highly unlikely for an adversary to control multiple nodes in a single circuit.

Changes to the Tor handshake that improve the user experience thus indirectly improve security. More users leads to more relays, which strengths Tor's resistance to an adversary that compromises a client's anonymity.

4.2 Design Improvements

While the improvements to bandwidth, checks, security, and future-proofing mentioned in Section 3.3 are standard, the reduced latency and censorship resistance have different effects on the usability of Tor than they would on a typical

internet connection. This section will discuss the importance of these changes in relation to the aforementioned threat models.

4.2.1 Censorship Resistance

The new design's use of native TLS greatly improves Tor's censorship resistance. Previously, Tor's modified TLS 1.2 had restrictions on certificate fields for backwards compatibility purposes. These fields could be analyzed by an adversary to selectively block Tor connections. As the new design uses native TLS, certificate fields no longer have restrictions and the threat of an adversary identifying and blocking Tor connections is reduced.

4.2.2 Latency

For Tor, reduction in latency not only means faster connections, but protection against an adversary that corrupts users' circuits. Reducing latency improves Tor's practical usability, encourages more clients to use it. Tor is a volunteer-based network, and so its popularity affects its ability to maintain a healthy number of relays. More relays subsequently decreases the chances that an adversary controls both an entry node and an exit node in a Tor circuit [SDM04].

5 Conclusion

The redesign of the Tor handshake has substantial benefits and limited downsides. The primary changes made were the removal of a redundant certificate and the backwards compatibility to v1 and v2. The design brings bandwidth improvement, a simplified handshake, and native TLS support. The native TLS support further improves security, latency, censorship resistance, and future updates. The design loses support of the few v1 and v2 relays, but these comprise about 0% of all nodes.

5.1 Future Work

Though Tor's TLS stage will be native under the new design, there may remain ways to identify its handshakes. Both parties must begin the Tor portion of the handshake immediately following the completion of the TLS portion. They must send a specific series of cells such as VERSIONS and CERTs. Though these messages are encrypted, onlookers may be able to identify Tor handshakes by

analyzing the volume of traffic sent following the TLS portion. Traffic analysis could be used to test the effectiveness of this type of attack in the future.

Additionally, the Tor handshake may be further optimized to reduce latency. The VERSIONS cell could potentially be sent by the server before the client in the second stage. This would reduce a round-trip time in the connection, greatly improving latency. Future work could include investigating the feasibility of changing this message order.

6 Acknowledgements

I would like to express my gratitude for my mentor Kyle Hogan, who supported and guided me throughout the research process. I would also like to thank the MIT PRIMES program for providing me with this research opportunity.

References

[1.3] Tls 1.3 - enhanced performance, hardened security. `https://www.cloudflare.com/learning-resources/tls-1-3/`. Accessed September 20, 2021.

[CJJ+19] Shan Chen, Samuel Jero, Matthew Jagielski, Alexandra Boldyreva, and Cristina Nita-Rotaru. Secure communication channel establishment: Tls 1.3 (over tcp fast open) vs. quic. Cryptology ePrint Archive, Report 2019/433, 2019.

[MAST19] Avijit Mallik, Abid Ahsan, Mhia Shahadat, and Jia-Chi Tsou. Man-in-the-middle-attack: Understanding in simple words. 3:77–92, 01 2019.

[Met] Tor metrics. `https://metrics.torproject.org/`. Accessed September 20, 2021.

[SDM04] Paul Syverson, Roger Dingledine, and Nick Mathewson. Tor: The secondgeneration onion router. In *Usenix Security*, pages 303–320, 2004.

[Spe] Tor protocol specification. `https://github.com/torproject/torspec/blob/main/tor-spec.txt`. Accessed June 20, 2021.

[SWZ16] John M Schanck, William Whyte, and Zhenfei Zhang. Circuit-extension handshakes for tor achieving forward secrecy in a quantum world. *Proc. Priv. Enhancing Technol.*, 2016(4):219–236, 2016.

[TLS] What is tls? https://www.cloudflare.com/learning/ssl/transport-layer-security-tls/. Accessed January 14, 2021.

[TOR] Tor project. https://www.torproject.org/. Accessed January 10, 2021.