TLS 1.3 Handshake Analyzer

Alexandre Augusto Giron[1,2], **Frederico Schardong**[1,3], **Ricardo Custódio**[1]

[1]Computer Security Lab – Graduate Program on Computer Science,
Department of Informatics and Statistics,
Federal University of Santa Catarina (UFSC)
Florianópolis – SC – Brazil

[2]Federal University of Technology - Paraná (UTFPR)
Toledo – PR – Brazil

[3]Instituto Federal do Rio Grande do Sul (IFRS)
Rolante – RS – Brazil

`alexandregiron@utfpr.edu.br, frederico.schardong@rolante.ifrs.edu.br`
`ricardo.custodio@ufsc.br`

Abstract. *The Transport Layer Security (TLS) protocol is the de facto standard for global Internet security. Despite its performance and security improvements over previous versions, there are still challenges regarding user privacy and security against future quantum attackers. Although there are several initiatives to address these challenges, there is no specialized tool to analyze these new features. This paper presents the TLS 1.3 Handshake Analyzer, a tool for security and performance analysis of TLS connections. Users can obtain security information about their connections, and server administrators can use it to validate the effects of TLS in their network applications, such as handshake timings and the sizes of transferred cryptographic objects.*

1. Motivation

The Secure Data Network System (SDNS), which is regarded one of the antecedents of today's Transport Layer Security (TLS) protocol, was developed by a collaborative initiative of US agencies in 1986. TLS gained prominence under a different name, Secure Sockets Layer (SSL), which was initially published in 1995. TLS 1.3 [Rescorla 2018] was released to the public after 23 years, and it is now the most recent version of the Internet's most-used security protocol.

Several sources indicate the widespread usage of TLS on the Internet. Web applications and Internet browsing are typical use cases, however TLS can also be used in embedded devices, automobile electric charging systems, and microservices [Naylor et al. 2014, Chan et al. 2018, Paracha et al. 2021]. Undoubtedly, Internet users will continue to utilize TLS in the near future.

Throughout the years, the ongoing improvement of TLS has resolved security vulnerabilities, such as the Bleichenbacher attack (solved in TLS 1.3 [Rescorla 2018]). Still, TLS 1.3 does not adequately protect user privacy, and it is vulnerable to the threat of quantum computers, according to security experts. In addition to these problems, non-expert users may be unaware of whether their connections leak information (such as the websites they browse) or whether they are employing quantum-vulnerable algorithms. An adversary could record TLS packets today and break TLS confidentiality in the future, even if there is no known quantum computer capable of breaching today's security parameters. This assault is known as *record-now-decrypt-later* [Bindel et al. 2019], and it poses the greatest threat to Internet security from quantum computers.

Fortunately, initiatives exist to address both the quantum threat and privacy concerns. One example is the TLS Encrypted Client Hello (ECH) extension for user privacy [Rescorla et al. 2022] and the Open-Quantum Safe (OQS) Project [Stebila and Mosca 2016] to safeguard users against future quantum assaults. However, most users have little to no knowledge about the security of their TLS connections. Moreover, these approaches may have a negative impact on network performance. The impact is mostly driven by the larger amount of data that must be sent by these more recent methods. Even though these methods are not yet standardized in TLS, it is likely that they will be used in the near future. ECH is being implemented and quantum-safe experiments have been conducted by major internet corporations [Braithwaite 2016, Patton 2022].

We introduce the TLS 1.3 Handshake Analyzer tool, which is designed to help end users to better understand the security and performance issues of TLS connections. In addition, web-application managers can assess the impact of ECH and quantum-safe algorithms on performance in comparison to current TLS handshakes. By examining TLS packets with our tool, users will be able to determine whether or not their connections employ privacy and quantum-safe techniques. The key features are:

- Supports analysis of TLS 1.3 handshakes, with or without TLS keylog files;
- Indicates whether quantum-safe algorithms from the OQS project are being utilized and if ECH extensions are present in the handshake;
- Based on CIPHERSUITE.INFO webservice [Rudolph and Grundmann 2022], displays whether insecure symmetric ciphersuites were used; and
- Displays handshake statistics and the size (in bytes) of exchanged TLS objects.

The structure of this document is as follows: Section 2 provides the necessary context. In Section 3, we describe the design of our tool. In Section 4, the tool is demonstrated in detail. Section 5 contains conclusions and future work.

2. Preliminaries

This section provides background information on TLS, the Encrypted Client Hello extension, Post-Quantum Cryptography (PQC), and existing SSL/TLS analyzers.

2.1. TLS 1.3 Overview

The TLS 1.3 protocol offers a confidential and authenticated channel for network applications. The security properties of TLS include: Authenticated Encryption with Associated Data (AEAD); Forward Secrecy (FS), where past communications are not compromised if a long-term cryptographic key is compromised; Downgrade protection; Non-replayability; and others [Rescorla 2018]. TLS 1.3 has three components: the Handshake Protocol, responsible for establishing TLS sessions; the Record Protocol, which protects application data with symmetric cryptography; and the Alert Protocol, which specifies error conditions and counteractions.

The handshake protocol specifies the client's first interaction with the server. The client initiates sending a `ClientHello` message, containing an ephemeral public key and additional metadata, such as the Server Name Indication (SNI). The server then responds with a `ServerHello` (including the corresponding ephemeral key), a `Certificate` (usually in the x509 standard format), a `CertificateVerify` (i.e., a digital signature), `EncryptedExtensions`, and a `Finished` message. These messages conclude the handshake (from the perspective of the server). If the client verifies and accepts those messages, he sends his `Finished` message, accompanied by encrypted application data, and then the session is established between the two parties. Additionally, the handshake protocol also specifies the client authentication scenario and connection resumption mode [Rescorla 2018].

2.2. Encrypted ClientHello Extension

One problem with the TLS 1.3 handshake is that it discloses (in clear) information about the connection, such as SNI and Application-Layer Protocol Negotiation (ALPN) lists, raising privacy concerns. SNI, for instance, displays the user's connection destination. Knowing this problem, different entities proposed solutions, but none of them are standardized by the Internet Engineering Task Force (IETF) yet. One promising solution is the Encrypted ClientHello (ECH) extension [Rescorla et al. 2022], which is already being adopted by companies such as Cloudflare [Patton 2022].

As the name suggests, the solution encrypts the `ClientHello` message under the public key of a co-located server (called a *client-facing server*). The purpose of this co-located server is to be in the middle of the communication (like a proxy), hiding the desired termination server that the client wants to connect to. So, the public key is distributed using a DNS-over-HTTPS (DoH) service, and thus clients have to obtain this public key prior to the communication. After obtaining the public key, the clients then create an ECH-inner message (encrypted) and an outer `ClientHello` message. The outer contains the inner, and the *client-facing server* can process which termination server the client wants to connect to. Therefore, the ECH extension reveals only that a client is connecting to a particular service provider, but it does not reveal the desired termination server. The termination server responds in the same way as the regular TLS 1.3 handshake but with some limitations. For instance, TLS 1.2 servers cannot use the ECH extension.

2.3. Post-Quantum Cryptography (PQC)

Several network protocols (such as TLS) rely on classical public-key cryptography not only for peer authentication but also for Key Exchange (KEX) mechanisms. Since classical cryptography is known to be vulnerable when a Cryptographically-Relevant Quantum Computer (CRQC) [Mosca 2018] is available, TLS (and the other protocols) are also vulnerable. Mainly, public-key schemes like RSA and ECDSA are vulnerable to Shor's algorithm [Shor 1994], but also symmetric primitives are affected. In theory, Grover's algorithm [Grover 1996] reduces the security of symmetric primitives by half.

In order to solve this issue, researchers started to study new schemes of cryptography, called Post-Quantum Cryptography (PQC) or Quantum-Safe Cryptography (QSC) [Stebila and Mosca 2016]. These schemes can be executed on standard computer architectures (non-quantum) and they are based on different mathematical problems which are not known to be vulnerable to quantum computers.

The most notorious effort related to PQC is the standardization process conducted by the US National Institute of Standards and Technology (NIST) [NIST 2016]. Recently, the end of the third round of the process defined which algorithms will be standardized:

- **Kyber**, for Key Exchange and Public-Key Encryption; and
- **Falcon**, **Dilithium** and **Sphincs+**, for authentication (digital signatures).

Additionally, NIST defined a fourth round in which other algorithms will be analyzed for standardization. BIKE, Classic McEliece, HQC, and SIKE are KEX mechanisms that passed to the fourth round (although SIKE was broken shortly after [Castryck and Decru 2022]). In this fourth round, NIST expects new proposals for digital signatures, with a particular interest in schemes in which the signatures have small sizes (compared to other PQC schemes). Nevertheless, the standardization process has not yet concluded. It is expected that, from now on, network protocols will gradually start adopting PQC schemes in their designs.

2.4. Existing Solutions

There are several tools available to test SSL/TLS connections [Geekflare 2022]. Table 1 provides a qualitative comparison of existing solutions. Some are open source, and most of them display security information (at least ciphersuites available). The majority of the tools do not show performance information and none mention ECH support.

Table 1. Comparison of existing SSL/TLS tools

Tool	Open source?	Shows security information?	Checks PQC/ECH Support?	Shows performance information?	User Interface
ssl-handshake	✓	✗	✗	Yes	CLI
howsmyssl	✓	✓	✗	✗	Web
SSL Labs	✗	✓	✗	✗	Web
SSL Checker	✗	✓	✗	✗	Web
TLS Scanner	✗	✓	✗	✗	Web
SSL Tester	✗	✓	✗	Yes (server-to-server)	Web
Observatory	✓	✓	✗	✗	Web/CLI
CryptCheck	✗	✓	✗	✗	Web
SSLChecker.com	✗	✓	✗	✗	Web
OQS-OpenSSL	✓	✓	PQC only	Yes	CLI
TLS 1.3 Handshake Analyzer (This work)	✓	✓	✓	Yes	Web/CLI

In particular, the OQS-OpenSSL fork [Stebila and Mosca 2016] allows checking security information and measuring performance using CLI commands. It includes PQC algorithms but no ECH support is provided (at this time of writing). Another interesting tool is the SSL Tester [Wormly 2022], since it provides security and performance information, obtaining handshake times for each ciphersuite supported by the server. It also shows the handshake sizes when choosing different ciphersuites. However, all timings are obtained from where the tool is hosted to the target website. Therefore, it does not reflect the handshake time considering the geographical location of the user. In the next section, we describe the tool proposed in this work.

3. Design and Implementation

Figure 1 shows the high-level overview of the tool. The user interacts through a web interface, but a CLI is also provided. The tool starts parsing the input capture file (`.pcap` or `.pcapng` files), with or without the corresponding TLS keylog file, aided by the `pyshark` module [Green 2022]. The tool retrieves handshake information and compares it with a local Object ID (OID) database, which is a list of quantum-safe algorithm OIDs based on the OQS project [Stebila and Mosca 2016]. Optionally, it checks for the ECH extension and also makes an HTTP POST request to CIPHERSUITE.INFO [Rudolph and Grundmann 2022] to retrieve security information about the ciphersuites in use: recommended, secure, weak, or insecure. The difference between weak and insecure is that the latter means a ciphersuite known to be vulnerable or "broken with minimal effort"; and the former means that it is an old ciphersuite and is not recommended by the IETF anymore. After retrieving all the required information, the tool computes handshake statistics (performance information) and presents them to the user.

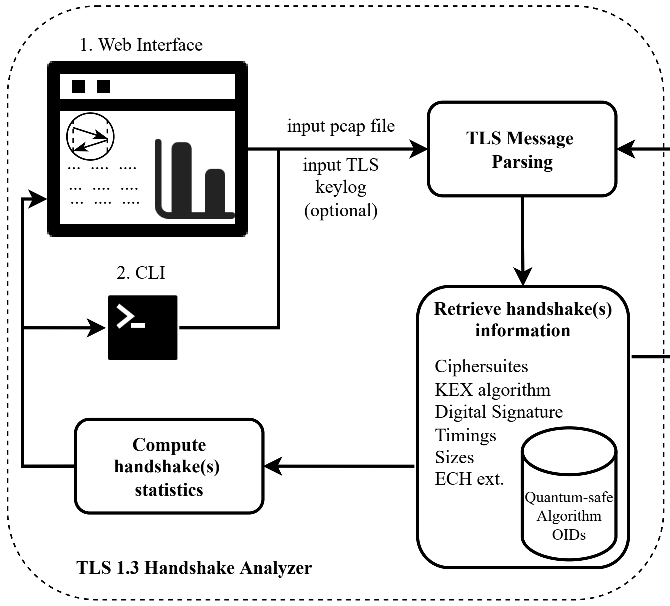

Figure 1. High-level overview of the tool design

The tool parses each TLS 1.3 complete handshake in the capture file. Currently, older versions of TLS and the resumption handshake are not supported. Normally, the resumption uses a Pre-Shared-Key (PSK) mode that can be used after a full handshake. Therefore, we focused on the complete handshake. After parsing, the following information is displayed to the user:

- Security Information:
  - Cryptographic algorithms used: the handshake metadata contains the KEX algorithm negotiated between the parties. Also, it includes the symmetric ciphersuite and, when the TLS keylog file is provided, the tool is able to decrypt `CertificateVerify` and `Certificate` messages, recovering the digital signature (and certificates) used for authentication. It is worth noting that the tool indicates whether or not quantum-safe algorithms are in use.

    - Ciphersuite check: optionally, the tool checks if the selected ciphersuite is secure, based on the information provided by CIPHERSUITE.INFO [Rudolph and Grundmann 2022].
    - ECH check: optionally, the tool parses the handshake looking for the ECH extension message in `ClientHello`. The presence of the extension indicates that this privacy-preserving method is supported.
- Performance Information: two metrics are used to evaluate performance, namely handshake timings and sizes. Regarding timings, they are based on the data provided in the capture file and are computed by subtracting the timestamps of the server's `Finished` message minus the `ClientHello` message. When more than one handshake is found, it shows average and standard deviation statistics. For sizes, the tool sums all handshake messages (in bytes) and plots their sizes.

The TLS 1.3 Handshake Analyzer is developed using the Python language (required version 3.9 or above). The web interface uses `Dash` [Plotly 2022], which is based on the `Flask` framework. Our tool can be executed directly or built using a Docker file, allowing easier deployment and usage. Additional documentation, such as README file and UML class diagrams, is provided at our repository [Giron et al. 2022].

4. Demonstration

We demonstrate our tool by analyzing two capture files: one handshake using PQC and the other with classical cryptography. Both captures were performed using the TEST.OPENQUANTUMSAFE.ORG webserver, which allows one to benchmark post-quantum TLS. Table 2 summarizes the handshake characteristics, and Figure 2 shows the per-message size graph created by our tool.

Table 2. Information retrieved from sample capture file

HS	Ciphersuite	KEX Algo.	Auth algo.	ECH Support?	HS Time (ms)
1	TLS_AES256_GCM_SHA384	kyber512	dilithium2	No	306.5
2	TLS_AES256_GCM_SHA384	x25519	ecdsa_secp256r1_sha256	No	307.4

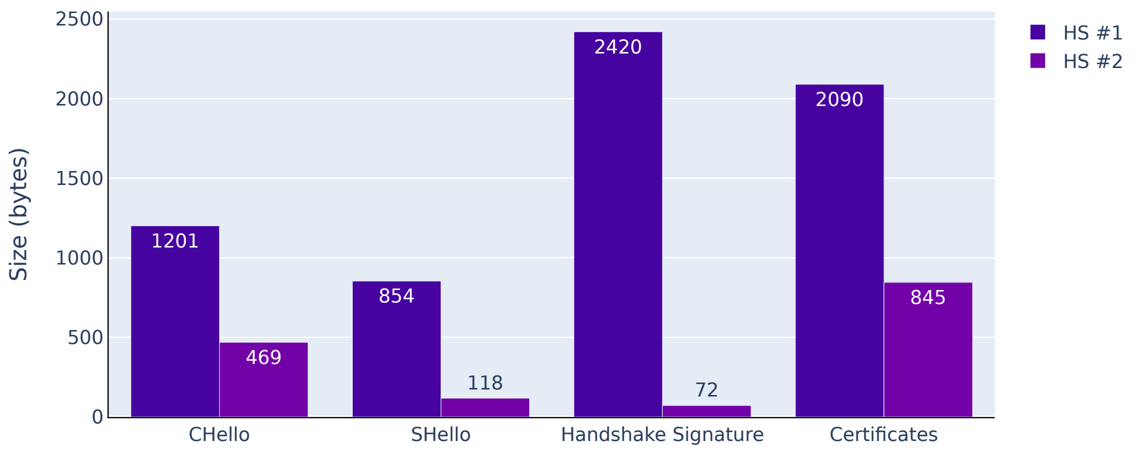

Figure 2. Per-message sizes of TLS sample capture

The tool indicates to the user whether the handshake used post-quantum algorithms. In this demonstration, HS #1 is quantum-safe but HS #2 is not. Additionally, both handshakes did not show support for the ECH extension. Nonetheless, the ciphersuites in use are secure. According to CIPHERSUITE.INFO, the ciphersuite `TLS_AES256_GCM_SHA384` is recommended for use.

Regarding performance information, the tool shows the handshake time (in milliseconds) and sizes. Note that both handshakes have similar performance, but in this test server only the HS #1 signature and KEX are PQC. The certificate chain of both handshakes remain with the same classical algorithm. No average or standard deviation is computed in this case because they are two separate capture files with only one handshake each. Nevertheless, the tool does support capture files with more than one handshake (and then providing additional statistics). Those and other sample capture files are available at our repository [Giron et al. 2022], along with a video demonstration of the tool.

5. Conclusions

Due to the transparency of TLS usage in web applications, users are unaware whether their connections have quantum-safe and privacy-friendly mechanisms. Aided by our TLS 1.3 Handshake Analyzer tool, users can now visualize detailed security and performance information about their connections, given that they provide the capture files.

Future work aims to improve the tool's usability: since `pyshark` provides a live-capture mode, we plan to integrate this feature into our analyzer. Still, there will be an additional requirement regarding the TLS keylog file configuration for this new feature. The live mode would allow users to check the security and performance of their connections in real-time.

6. Acknowledgments

Thanks go to the Federal Institute of Education, Science and Technology of Rio Grande do Sul (Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul, IFRS) and to the Federal University of Technology - Parana (UTFPR), which allowed doctoral studies for Frederico Schardong and Alexandre Augusto Giron, respectively.

References

Bindel, N., Brendel, J., Fischlin, M., Goncalves, B., and Stebila, D. (2019). Hybrid key encapsulation mechanisms and authenticated key exchange. In Ding, J. and Steinwandt, R., editors, *Post-Quantum Cryptography*, pages 206–226, Cham. Springer.

Braithwaite, M. (2016). Experimenting with post-quantum cryptography. Available at: `https://security.googleblog.com/2016/07/experimenting-with-post-quantum.html`. Accessed on 2021-02-25.

Castryck, W. and Decru, T. (2022). An efficient key recovery attack on sidh (preliminary version). Cryptology ePrint Archive, Paper 2022/975. `https://eprint.iacr.org/2022/975`.

Chan, C.-l., Fontugne, R., Cho, K., and Goto, S. (2018). Monitoring tls adoption using backbone and edge traffic. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 208–213. IEEE.

Geekflare (2022). 10 online tools to test ssl. Available at: https://geekflare.com/ssl-test-certificate/. Accessed on 2022-08-27.

Giron, A. A., Schardong, F., and Custódio, R. (2022). Tls handshake analyzer github repository. Available at: https://github.com/AAGiron/tls-handshake-analyzer. Accessed on 2022-08-28.

Green, D. (2022). Pyshark. Available at: https://kiminewt.github.io/pyshark/. Accessed on 2022-07-13.

Grover, L. K. (1996). A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, Philadelphia Pennsylvania USA. ACM.

Mosca, M. (2018). Cybersecurity in an era with quantum computers: Will we be ready? *IEEE Security Privacy*, 16(5):38–41.

Naylor, D., Finamore, A., Leontiadis, I., Grunenberger, Y., Mellia, M., Munafò, M., Papagiannaki, K., and Steenkiste, P. (2014). The cost of the" s" in https. In *10th ACM International on Conference on emerging Networking Experiments and Technologies*, pages 133–140.

NIST (2016). Post-quantum cryptography. Available at: https://csrc.nist.gov/Projects/Post-Quantum-Cryptography. Accessed on 2020-06-26.

Paracha, M. T., Dubois, D. J., Vallina-Rodriguez, N., and Choffnes, D. (2021). Iotls: Understanding tls usage in consumer iot devices. In *21st ACM Internet Measurement Conference*, IMC '21, New York, NY, USA. Association for Computing Machinery.

Patton, C. (2022). Good-bye esni, hello ech! Available at: https://blog.cloudflare.com/encrypted-client-hello/. Accessed on 2022-07-09.

Plotly (2022). Dash documentation and user guide. Available at: https://dash.plotly.com/. Accessed on 2022-07-13.

Rescorla, E. (2018). The transport layer security (TLS) protocol version 1.3. RFC 8446, RFC Editor. Available at: http://www.rfc-editor.org/rfc/rfc8446.txt. Accessed on 2021-05-02.

Rescorla, E., Oku, K., Sullivan, N., and Wood, C. A. (2022). Tls encrypted client hello. Internet-Draft draft-ietf-tls-esni-14, IETF Secretariat. https://www.ietf.org/archive/id/draft-ietf-tls-esni-14.txt.

Rudolph, H. C. and Grundmann, N. (2022). Ciphersuite info. Available at: https://ciphersuite.info/. Accessed on 2022-07-12.

Shor, P. W. (1994). Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134, Santa Fe, NM, USA. IEEE, IEEE.

Stebila, D. and Mosca, M. (2016). Post-quantum key exchange for the internet and the open quantum safe project. In *International Conference on Selected Areas in Cryptography*, pages 14–37. Springer.

Wormly, I. (2022). Free ssl web server tester. Available at: https://www.wormly.com/test_ssl. Accessed on 2022-08-27.