ENG25519: Faster TLS 1.3 handshake using optimized X25519 and Ed25519

Jipeng Zhang, *CCST, Nanjing University of Aeronautics and Astronautics;* Junhao Huang, *Guangdong Provincial Key Laboratory IRADS, BNU-HKBU United International College; Hong Kong Baptist University;* Lirui Zhao, *CCST, Nanjing University of Aeronautics and Astronautics;* Donglong Chen, *Guangdong Provincial Key Laboratory IRADS, BNU-HKBU United International College;* Çetin Kaya Koç, *CCST, Nanjing University of Aeronautics and Astronautics; Iğdır University; University of California Santa Barbara*

https://www.usenix.org/conference/usenixsecurity24/presentation/zhang-jipeng

August 14–16, 2024 • Philadelphia, PA, USA

978-1-939133-44-1

USENIX Security '24 Artifact Appendix: ENG25519: Faster TLS 1.3 handshake using optimized X25519 and Ed25519

Jipeng Zhang[1]
[1]CCST, Nanjing University of Aeronautics and Astronautics
jp-zhang@outlook.com

A Artifact Appendix

A.1 Abstract

The artifact comprises three parts of source code, each corresponding to a separate folder. The first part (i.e. `Crypto/` folder) focuses solely on optimized implementations of cryptographic algorithms. The second part (i.e. `ENG25519/` folder) involves an OpenSSL ENGINE called ENG25519 and encompasses end-to-end experiments, including TLS 1.3 handshake and DNS over TLS (DoT). The third part (i.e. `Verify/` folder) involves the utilization of the CryptoLine toolchain for formal verification of our finite field implementations, ensuring correctness and robustness of the implementations. We provide README documents in each sub-project, primarily referencing these README documents to describe the replication steps.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

All implementations will not cause harm to your computer.

A.2.2 How to access

The code is available at `https://github.com/Ji-Peng/eng25519_artifact/tree/usenix_security2024`.

A.2.3 Hardware dependencies

The essential requirement for executing our experiments is support for the AVX-512IFMA instruction set. See CPUs with AVX-512 for more details.

The machines for our benchmarks are obtained from Alibaba Cloud. The details of the machine are as follows.

- Family: Compute Optimized Type c7.
- Instance Type: ecs.c7.2xlarge.
- vCPUs: 8 vCPUs.
- Memory: 16 GiB.
- Internal Network Bandwidth: Up to 10 Gbps.
- Physical Processor: Intel Xeon (Ice Lake) Platinum 8369B.

If you just want to reproduce the experimental results of the cryptographic performance part of the paper, one machine is enough. If you want to reproduce the experimental results of the DoT (DNS over TLS) part of the paper as accurately as possible, you should rent two machines in an internal network. When you purchase two machines on Alibaba Cloud at the same time, they are often in an internal network.

Potential impacts due to hardware variations Running experiments on different CPU models may yield varying performance results; for example, you may observe different CPU cycle consumption of a cryptographic primitive. For similar results, it's advisable to choose CPUs that are the same or similar. For our end-to-end experiments, communication between the client and server occurs over an internal network. If in your setup, the communication path between the client and server is longer, this may also lead to different outcomes.

A.2.4 Software dependencies

- Linux (Ubuntu 20.04 in our case)
- OpenSSL 1.1.1q

We provide installation steps for OpenSSL in each sub-project. Please note that OpenSSL does not need to be reinstalled; it only needs to be executed once.

A.2.5 Benchmarks

Our experiments do not rely on any data. The raw data obtained from our end-to-end experiments is available in the folder ENG25519/eng25519-results.

A.3 Set-up

A.3.1 Installation

For reproducing the performance results of cryptographic optimizations (i.e. `Crypto/` folder), the main dependency is OpenSSL. To replicate the results of end-to-end experiments (i.e. `ENG25519/` folder), dependencies include OpenSSL, unbound, tls_timer, and dot_timer. For the formal verification of our finite field implementations (i.e. `Verify/` folder), the primary dependency is the CryptoLine toolchain. Detailed installation instructions for these dependencies are provided in the corresponding README files in their respective folders.

A.3.2 Basic Test

Specific instructions for running the tests are also provided in detail in the README files in the corresponding folders.

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): Table 2 compares our X25519 implementation with others. The throughput (executions per second) of our implementation is 12.01 times that of OpenSSL X25519-KeyGen. Table 3 provides a comparison of various Ed25519 implementations. Ed25519-Sign and Ed25519-Verify achieve 3.79 and 3.33 times higher throughput than OpenSSL, respectively. We also give the corresponding fixed-point scalar multiplication comparison in Table 4, showing that the throughput of our optimized fixed-point multiplication is 8.06 times and 1.67 times that of OpenSSL and Faz-Hernández et al., respectively. This is proven by the experiment (E1).

(C2): As shown in Figure 3a, the proposed ENG25519 setting (1,707 #connections/second) enables 25% and 35% more handshakes per second than X25519 (1,366) and P256 (1,260), respectively. For the end-to-end experiment targeting DoT queries, our ENG25519 outperforms all other configurations, as illustrated in Figure 3b. Regarding peak throughput for DoT servers, our ENG25519 configuration achieved a significant improvement, achieving 290,315 #queries/min, which represents a 41% and 24% increase over P256 (206,275) and X25519 (234,875), respectively. This is proven by the experiment (E2).

(C3): We utilize a semi-automatic formal verification tool, CryptoLine, to provide mathematical guarantee of the correctness of our finite field implementations. This is proven by the experiment (E3).

A.4.2 Experiments

(E1): *The performance of our optimized cryptographic implementations (i.e. Crypto/ folder) (0.5 human-hour + 0.2 compute-hour)*

How to: Table 2 and Table 3 in the paper provide comparisons between our work and OpenSSL, Faz-H et al., Hisil et al., Nath et al., and Cheng et al. The project under the `Crypto/Curve25519\_AVX512IFMA` path can reproduce the performance of OpenSSL and our work; please refer directly to the README file in that directory. The project under the `Crypto/Hisil\_AVX512` path can reproduce the performance of Hisil et al.; please refer directly to the README file in that directory. For the other three works, we provide links to their projects in the README file at the top-level directory; please refer directly to their projects to reproduce their performance.

(E2): *End-to-end experiments of TLS handshakes and DoT queries (i.e. ENG25519/ folder) (1 human-hour + 28 compute-hour)*

How to: This experiment primarily refers to the README file under the `ENG25519/eng25519` path. In this README, we also reference the `ENG25519/DoT_timer` project and the `unbound` project. The main reason for the time-consuming nature of this experiment is that end-to-end experiments require running for extended periods to collect experimental data. The most complex and time-consuming aspect is the reproduction of DoT query peak throughput. We provide some scripts to automate this time-consuming process; however, it's essential to note that adjustments according to your configuration are necessary before using them. For more detailed information, please refer to the README file under the `ENG25519/eng25519` path.

(E3): *Formal verification of our finite field implementations (i.e. Verify/ folder) (0.5 human-hour + 0.5 compute-hour)*

How to: This experiment primarily refers to the README file under the `Verify/` path. The main step is to install the CryptoLine toolchain.

A.5 Version

Based on the LaTeX template for Artifact Evaluation V20231005. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenixsec2024/.