Quantum-Proofing Data: The Power Of Post-Quantum Cryptography

Krystian Matusiewicz
Intel, Poland
krystian.matusiewicz@intel.com

Milena Olech
Intel, Poland
milena.olech@intel.com

Natalia Wochtman
Intel, Poland
natalia.wochtman@intel.com

Abstract

Quantum algorithms can efficiently solve some problems that are difficult for classical computers. Notably, Shor's algorithm can be used for efficient integer factorization and discrete logarithm finding. While there are still no sufficiently large quantum computers to threaten practically used RSA and Elliptic Curve Diffie-Hellman instances, deploying quantum-resistant alternatives to RSA and EC-based cryptography is becoming an urgent need.

Quantum attacks pose a threat to cryptography utilized in the TLS protocol in two critical areas: key exchange and client/server authentication. As a result, the security of the TLS handshake process can be compromised, necessitating the adoption of post-quantum cryptography (PQC) algorithms.

This year, the National Institute of Standards and Technology selected a suite of asymmetric algorithms resistant to quantum attacks. One of them is CRYSTALS-Kyber, a Key Encapsulation Mechanism (KEM) that can be used to establish a session key securely.

This paper studies its application in the TLS handshake. In order to assess the efficacy of post-quantum cryptography (PQC), a comprehensive comparison between traditional cryptographic methods and post-quantum alternatives has been conducted. The study analyzes the outcomes and provides a condensed summary of the findings.

Introduction

Quantum computers leverage principles of quantum mechanics to perform computations by controlling the evolution of a state of a quantum system. By carefully setting up a quantum system and making it undergo state transformations (called quantum gates) one can arrive at a destination state that after measuring yields the result of the computation.

It turns out that that there exist computations that can be done more efficiently on a quantum computer. One of them is a Quantum Fourier Transform (Cop94). This algorithm is the crucial ingredient of the algorithm developed by Peter Shor (Sho95) that, by finding periods of appropriately defined functions, is able to efficiently factor composite integers and find discrete logarithms in finite fields.

There are no quantum computers with sufficiently large state to be able to factor practically used RSA moduli yet, but the race to scale up quantum computers is ongoing and we may expect significant progress in that area over the next couple of years. U.S. administration treats the risk as real enough for the President to issue a dedicated National Security Memorandum (Bid).

National Institute of Standards and Technology has been running a public competition (Nat) to address this concern and select a new set of quantum-resistant algorithms that will be standardized to augment legacy schemes from FIPS-186.

The deployment of quantum-resistant algorithms is meant to mitigate the threat of "store now, decrypt later" attacks where the communication is being recorded now and will be decrypted later when the advanced threat actors will have access to powerful enough quantum computers.

On the other hand, the adoption of post-quantum cryptography involves completely new algorithms and carries cryptanalytic, implementation and operational risks. One of the solutions to mitigate those risks is to use hybrid modes combining both classical and post-quantum algorithms. That direction is recommended by French and German cybersecurity agencies (Age; Bun) and has been explored e.g. in (CPS19), (SFG23).

Taking into account that the key exchange and client/server authentication utilized in TLS protocol are significantly threaten, there is a need to consider defense methods that can be applied to current cryptographic system.

This paper provides an extended background for all related areas, like Transport Layer Security, Key Encapsulation Mechanism, Shor's algorithm and Learning with Error Problem. Subsequently, performed research has been described, including methodology, used tools, and the set of results. The purpose of the research was to determine whether using hybrid keys - traditional one combined with the Post Quantum - significantly increases CPU load compared to traditional methods.

Related work

Transport Layer Security

Transport Layer Security Protocol (Res18) allows client/server applications to communicate over Internet securely.

Providing a secure channel between the client and the server is the primary goal of the TLS. Secure channel shall ensure that the following properties are supported:

- Authentication

  The concept of TLS assumes that the server side is always authenticated. In the case of the client, authentication is optional. Authentication may happen using asymmetric cryptography, the Elliptic Curve Digital Signature Algorithm (ECDSA), The Edwards-Curve Digital Signature Algorithm (EdDSA), or a symmetric pre-shared key (PSK).

- Confidentiality

  Data sent over the channel shall be visible exclusively to the endpoints. The length of the data is not hidden by TLS itself, but endpoints may apply padding to the TLS records to obfuscate the length and improve protection level.

- Integrity

  Attackers cannot change data sent over the channel.

These requirements should be full-filled even if the attacker has complete control of the network.

TLS includes two main components:

- Handshake protocol

  Handshake protocol is responsible for negotiating cryptographic parameters and modes and establishing shared keys. The protocol shall resist changes made by the potential attacker, which means that the attacker cannot influence the negotiation between server and client by forcing parameter changes.

- Record protocol

  Record protocol uses parameters negotiated during the handshake to protect traffic between server and client. The primary responsibility of this protocol is to divide the traffic into a series of records and protect them independently using appropriate traffic keys.

TLS does not influence the application layer. Higher-level protocols can operate on the top of the TLS transparently. The standard does not provide requirements for adding security with TLS, initiating TLS handshake, and interpreting the authentication certificates exchanged between the server and the client.

Key Encapsulation Mechanism (KEM)

Ken Encapsulation Mechanisms (KEMs) are a set of encryption methods designed to provide secure symmetric key material for transmission using asymmetric (public-key) algorithms (YHM23).

The standard method for sending a symmetric key with the public key system is following:

1. Generate a random symmetric key
2. Encrypt using the chosen public key algorithm
3. Recipient decrypts the public key message to retrieve symmetric key

Padding is required to provide better security if the symmetric key is relatively short.

KEMs are designed to simplify the process by generating a random element in the finite group underlying the public key system and deriving a symmetric key by hashing this element and eliminating the need for padding.

Two peers can agree on the secret value if the first peer can send the secret in an encrypted form and the second one can decrypt and use it. KEM enables that flow by three algorithms:

- Key generation algorithm: KeyGen() -> (pk, sk)
- Encapsulatiton algorithm: Encaps(pk) -> (ct, ss)
- Decapsultion algorithm: Decaps(sk, ct) -> ss

where pk stands for the private key, sk represents the secret key, ct is a ciphertext, and ss stands for shared secret.

The first algorithm generates a public key and a private key, called a keypair. The second algorithm, encapsulation, takes a public key as an input and outputs a shared secret value and a ciphertext of this secret value. The last, decapsulation, takes the ciphertext and the private key as input and outputs the shared secret value.

KEM can be considered similar to Public Key Encryption (PKE) because both use keypairs - public and private keys. However, in the case of PKE, one peer encrypts a message using the public key and decrypts the message using the private key. When KEM is used, one peer uses the public key to create an encapsulation, and the second one decrypts this encapsulation with the private key.

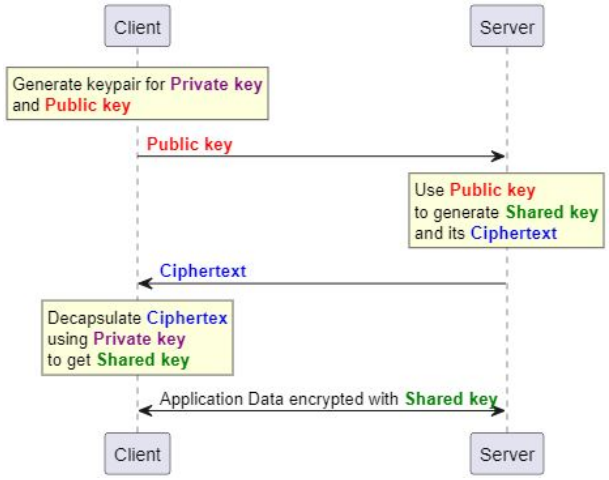


Figure 1: TLS handshake in Key Encapsulation Mechanisms

Shor's algorithm

In 1994, Peter Shor - a scientist working for Bell Labs, created a polynomial time algorithm for factoring large numbers of a quantum computer. The algorithm is considered crucial from the view of cryptography because most cryptographic systems rely on the difficulty of factoring large numbers. If an efficient method for factoring large numbers was found, most current encryption systems would be seriously compromised.

It has yet to be proven that factoring of large numbers can be achieved on a classical computer in polynomial time. The fastest, widely known algorithm for factoring a large number n runs in $O(e^{c(logn)^{\frac{1}{3}}*(loglogn)^{\frac{2}{3}}})$, or expotential time. Contrary to quantum computers, where the Shor's algorithm runs in $O((logn)^2 * loglogn)$ and must continue on a classical computer with $O(logn)$ steps of post-processing. It means that the quantum computer is capable of finding the prime factors of an integer in a polynomial time (Sho95; Hay15).

Shor's algorithm has the potential to break algorithms that are pivotal for key exchange in Transport Layer Security (TLS):

- The RSA scheme
- The Finite Field Diffie-Hellman key exchange
- The Elliptic Curve Diffie-Hellman key exchange

Post Quantum Cryptography

Quantum computers will be capable of solving mathematical problems that are difficult or impossible to solve for traditional computers. If quantum computers are built, they will bring a possibility to break many of the public-key cryptosystems that are currently used in security (BL17). That is why the first steps in post-quantum cryptography (also called quantum-resistant cryptography) have been issued. The primary purpose of post-quantum cryptography is to prepare cryptographic methods that are secure against quantum and classical computers attacks and can be easily incorporated into existing communication protocols (CPS19).

The National Institute of Standards and Technology (NIST) initiated a process to standardize quantum-resistant public-key cryptographic algorithms in 2017.

In July 2022 NIST selected Crystals Kyber as a Public-key Encryption and Key-establishment algorithm and CRYSTALS-DILITHIUM, SPHINCS+ and FALCON for Digital Signature Algorithms.

Learning with Error Problem

Learning with Errors is a mathematical problem used in cryptography to create secure encryption algorithms.

The main idea is to have secret information presented as a set of equations with errors, so the shared secret is hidden behind additional noise that cannot be easily solved from the mathematical point of view.

At first, a secret key value s and additional value e are created. Subsequently, a number of values $A[]$ are selected and following equation shall be solved:

$$B[] = A[] \times s + e \tag{1}$$

Values of $A[]$ and $B[]$ indicates the public key. If s is a single value, then both A and B are one-dimensional matrices. However, if s is a one-dimensional matrix, then A is a two-dimensional matrix, and B is one-dimensional matrix. The difficulty is to solve equation 1, having only A and B - the public key.

Steal now, decrypt later

Security experts warn there is a risk of storing encrypted data for the future - when quantum computers will achieve a maturity level capable of cracking currently used algorithms. Advanced threat actors may already steal the data and try to decrypt it once possible.

The potential scale of information disclosure will be enormous, posing a significant threat to everyone. The technique to "steal now, decrypt later" brings a significant risk of information leaks both in businesses and governments.

In July 2022, The US Cybersecurity and Infrastructure Security Agency shared a roadmap that the organization should follow in cooperation with NIST to prepare for a transition.

Most post-quantum algorithms are relatively new, so the level of understanding of their usage has yet to be studied as profoundly as the traditional approach like RSA or elliptic curve Diffie-Hellman. That is why the security community needs more confidence in their security.

Additionally, even if the post-quantum algorithms are officially defined, some users will not be eager to use them immediately. In general, updating cryptographic algorithms is a time-consuming process for most organizations that need to modify a significant amount of hardware and software. Transition in network protocols consists of several steps. Firstly, the protocol must be examined for any constraints that may influence adding new algorithms with entirely new characteristics. Next, there is a need to prepare an efficient method for integration into a protocol. Finally, the design must provide backward compatibility with not upgraded endpoints (CPS19).

On the other hand, some users may want to accelerate the acquisition because of the 'steal now, decrypt later' danger. The availability of quantum computers may be a cryptographic breakthrough and pose the risk of data security where handshakes and encrypted communications were previously recorded.

Hybrid keys are a response to the needs of both groups. Combining existing encryption methods with quantum-resistant algorithms protects current traffic from future attacks and provides a stable, well-known security level. The challenge is hybrid mode includes an additional factor that was not considered in previous cryptographic transitions: the use of two cryptographic algorithms simultaneously.

Kyber overview

Kyber is an IND-CCA2-secure KEM (BKS19). The security of the Kyber is based on the difficulty of solving the Learning with Errors (LWE) problem over module lattices. Kyber-512 is considered to provide a security level roughly equivalent to AES-128, Kyber-768 is considered to provide a security level roughly equivalent to AES-192, and Kyber-1024 is considered to provide a security level roughly equivalent to AES-256.

Version	PR size	PK size	CT size
Kyber-512	1632	800	768
Kyber-768	2400	1184	1088
Kyber-1024	3168	1568	1568

where PR size stands for Private Key size, PK size stands for Private Key size, and CT size stands for Ciphertext size.
Kyber aims to establish a shared secret between two peers with solid resistance to future quantum computer attacks.

Hybrid keys

This chapter discusses the increasing necessity for hybrid key exchange mechanisms within traditional internet security protocols like TLS, primarily due to the looming threat posed by quantum computing. While secure against current classical computers, traditional cryptographic methods are potentially at risk from quantum technologies. To address this potential threat, a hybrid approach combining standard algorithms, like Elliptic Curve Diffie-Hellman (ECDH), with those resistant to quantum attacks (e.g., Kyber, Bite Flipping Key Exchange BIKE, Supersingular Isogeny Key Exchange SIKE).

However, despite their theoretical security, these quantum-resistant algorithms are relatively new and have yet to be tested as thoroughly as traditional algorithms. The hybrid model's strength lies in its redundancy. It combines traditional and PQ exchange results to provide better security if one gets compromised.

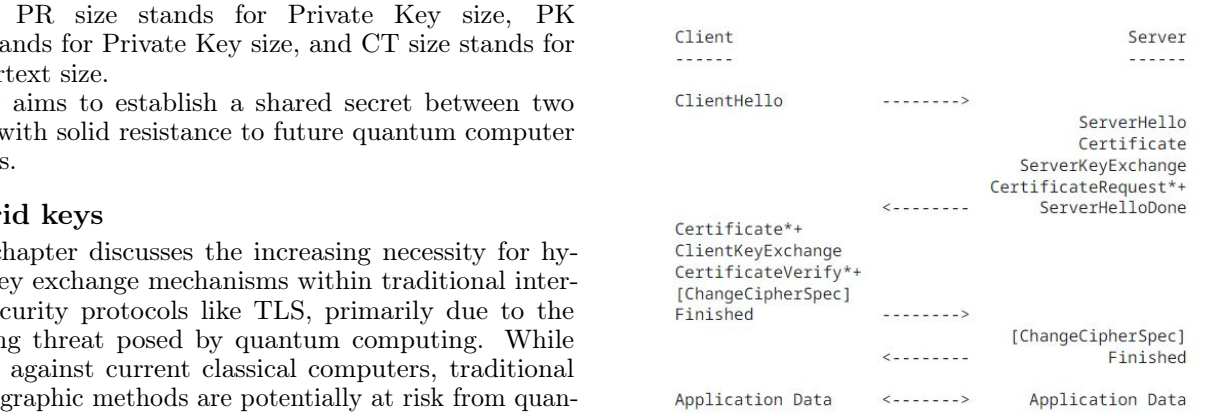
The hybrid key exchange in TLS 1.2 modifies ClientHello, ServerHello, ServerKeyExchange, and ClientKeyExchange messages. Client Hello and ServerHello messages are modified to indicate support for PQ key exchange extensions. Once a hybrid key exchange is negotiated, the server sends its ephemeral ECDH public key, generated using the corresponding curve, and ephemeral Kyber public key generated with Kyber parameters. Both keys with corresponding params are sent in the ServerKeyExchange message.

In response, the client generates its own ECDHE key pair on the same curve as the server's ephemeral ECDH key, computes a ciphertext value based on the Kyber public key received in the ServerKeyExchange, and sends them back in the ClientKeyExchange message.

Both sides perform an ECDH operation to generate the resultant ECDHE shared secret Z as part of the premaster secret. The server side performs the Kyber decapsulation routine to compute the Kyber encapsulated key K. Premaster secret is formed by concatenating both keys as premaster_secret = Z || K.

The TLS 1.3 standardized two separate Hybrid Key Exchange Methods. The first one uses ECDHE with Kyber768 (KK23) and the second one combining X25519 with Kyber768 (BW23). Both Key Exchange Methods were also added to the supported_groups extensions. For hybrid key exchange in TLS 1.3 the key_exchange field in the KeyShareEntry is the concatenation of key_exchange fields for each hybrid algorithm.

Figure 2: Message flow in a hybrid TLS handshake (CPS19)



s2n-tls

Amazon's s2n-tls is an open-source implementation of the TLS (Transport Layer Security) protocol available under the terms of the Apache License 2.0. It focuses on simplicity and security and aims to be a more secure alternative by reducing the potential attack surface and making the code easier to review and maintain compared to traditional TLS implementations. Compared to OpenSSL, which contains around 70 000 lines of code to support TLS, it implements it in significantly less (around 6 000) and undergoes external security evaluations and penetration tests.

The key exchange scheme of s2n is based on combining exactly one traditional algorithm and one PQ. It implements the hybrid specification from TLS 1.2 and TLS 1.3 hybrid specification. It defines new cipher suites where the key exchange mechanism is a hybrid of ECDHE and post-quantum KEM. For example, ECDHE-KYBER-RSA-AERS256-GCM-SHA384 is a hybrid cipher suite with ECDHE for the classical component and Kyber for the PQ component of the key exchange.

Results

The primary purpose of this paper was to measure and compare the CPU load when traditional and hybrid key exchanges in a TLS handshake are performed. The data has been collected using s2n-tls, Amazon's TLS implementation described in the previous section, and all operations were executed on Intel© Core™ Xeon(R) Platinum 8280 CPU @ 2.70GHz.

Methodology

In general, hybrid key exchange method has significant impact on the *ClientHello*, *the ServerHello*, the *ServerKeyExchange*, and the *ClientKeyExchange* messages presented on Figure 2.

An extension for the post-quantum key encapsulation method enables the negotiation of specific PQ KEM parameters during a handshake. The TLS client that proposes PQ KEM cipher suite in its *ClientHello* message shall include PQ extension. The TLS servers that implement PQ KEM cipher suite shall support this extension. As a result, when the client uses PQ extension, the server should only negotiate using a PQ KEM parameters once the handshake is completed. This prevents unexpected negotiation aborting when the client cannot deal with the server's PQ KEM key (CPS19).

The value examined in this research is the number of CPU cycles spent on establishing the connection between the client and the server. All results were collected using a perf tool capable of measuring the performance counters for Linux subsystems.

The s2n-tls provides two applications - ***s2nd*** and ***s2nc*** - that show the usage of numerous s2n-tls APIs for the server and the client sides accordingly.

The framework s2n-tls enables choosing the method of key exchange. For the research described in this paper, a hybrid key that consists of ECDHE and Kyber has been compared to the traditional ECDHE method.

Furthermore, a flame graph is generated to visualize the hierarchical data. The flame graph shows a trace of distributed requests and depicts each service invocation during execution with colored horizontal bars.

The tests for both method1 and method2 were repeated ten times to increase the predictability and stability.

Results show that the CPU usage difference is relatively small - 554059 cycles of the CPU. The CPU used in this research is running at 2.70GHz frequency, so one cycle takes around 0.37 nanoseconds. Considering the time, the difference between post-quantum and traditional key exchange methods equals approximately 200 microseconds.

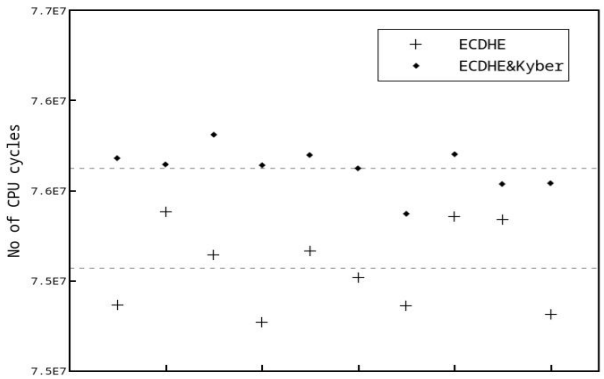
Compared to the traditional method, generated flamegraphs visualize the complexity that hybrid key exchange introduces to the source code.

For TLS handshake in s2n reference application, negotiation consumes 10.345% of CPU usage for the traditional ECDHE method and 16.129% of CPU usage for Kyber and ECDHE. The record command was sampled at 100Hz, and the number of samples equals 3 and 5, respectively. Results are presented on Figure4 and Figure5.

Table 1: CPU load for hybrid and classical key exchange

	ECDHE	ECDHE & Kyber
No of CPU cycles	74863824	75678482
	75382136	75645037
	75139931	75810349
	74766096	75639798
	75163583	75695950
	75017512	75622811
	74860248	75371292
	75352972	75698335
	75335142	75532862
	74812731	75539853
Average	**75069418**	**75623477**
	Difference	**554059 cycles**

Figure 3: CPU load for hybrid and classical key exchange



Conclusions

Sensitive information, such as bank transfers and personal data, is currently protected by public key encryption techniques based on math problems. Sufficiently powerful quantum computers can defeat these techniques.

The presented study investigated the implications of substituting traditional key exchange methodologies with a hybrid key exchange system on CPU workload. The results of our study suggest a negligible increase in CPU workload following this modification.

The comparison proves that the CRYSTALS-Kyber algorithm, designed for general encryption purposes and selected by NIST as a candidate for standardization strongly relies on the device used as a client or server and may not affect the CPU usage notably.

These findings comply with the conclusions of other researchers regarding the potential impacts of incorporating Post-Quantum Cryptography (PQC) into modern systems. It has also been proven that in some use cases, the energy consumption for PQ implementation in TLS 1.3 can be equal to the traditional TLS (TDF+23).

This insight underscores the viability of post-quantum cryptographic methods in contemporary communication frameworks.

References

[Age] Agence nationale de la sécurité des systèmes d'information. ANSSI views on the post-quantum cryptography transition. https://www.ssi.gouv.fr/en/publication/anssi-views-on-the-post-quantum-cryptography-transition/.

[Bid] Joseph H. Biden. National security memorandum on promoting united states leadership in quantum computing while mitigating risks to vulnerable cryptographic systems. The White House, 2022.

[BKS19] Leon Botros, Matthias J. Kannwischer, and Peter Schwabe. Memory-efficient high-speed implementation of Kyber on Cortex-M4. In *Progress in Cryptology – AFRICACRYPT 2019*, pages 209–228. Springer, 2019.

[BL17] Daniel J. Bernstein and Tanja Lange. Post-quantum cryptography. *Nature*, 549(7671):188–194, Sep 2017.

[Bun] Bundesamt für Sicherheit in der Informationstechnik. Migration zu Post-Quanten-Kryptografie. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Krypto/Post-Quanten-Kryptografie.pdf.

[BW23] Douglas Stebila Bas Westerbaan. X25519kyber768draft00 hybrid post-quantum key agreement draft-tls-westerbaan-xyber768d00-03. University of Waterloo, 2023.

[Cop94] Don Coppersmith. An approximate Fourier transform useful in quantum factoring. Technical Report RC 19642, IBM Research, 1994. https://arxiv.org/abs/quant-ph/0201067.

[CPS19] Eric Crockett, Christian Paquin, and Douglas Stebila. Prototyping post-quantum and hybrid key exchange and authentication in TLS and SSH. 2nd NIST PQC Standardization Conference, 2019. https://csrc.nist.gov/CSRC/media/Events/Second-PQC-Standardization-Conference/documents/accepted-papers/stebila-prototyping-post-quantum.pdf.

[Hay15] Matthew Hayward. Quantum Computing and Shor's Algorithm. https://github.com/digitalmacgyver/quant/tree/master/299/, 2015.

[KK23] Panos Kampanakis Kris Kwiatkowski. Post-quantum hybrid ecdhe-kyber key agreement for tlsv1.3. PQShield, LTD, AWS, 2023.

[Nat] National Institute of Standards and Technology. Post Quantum Cryptography Standardization. https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022. Accessed: 2023-10-03.

[Res18] Eric Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446, 2018.

[SFG23] Douglas Stebila, Scott Fluhrer, and Shay Gueron. Hybrid key exchange in TLS 1.3. IETF draft, https://datatracker.ietf.org/doc/draft-ietf-tls-hybrid-design/, 2023.

[Sho95] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Rev.*, 41:303–332, 1995.

[TDF+23] George Tasopoulos, Charis Dimopoulos, Apostolos P. Fournaris, Raymond K. Zhao, Amin Sakzad, and Ron Steinfeld. Energy consumption evaluation of post-quantum tls 1.3 for resource-constrained embedded devices. In *Proceedings of the 20th ACM International Conference on Computing Frontiers*, CF '23, page 366–374, New York, NY, USA, 2023. Association for Computing Machinery.

[YHM23] Jiewen Yao, Anas Hlayhel, and Krystian Matusiewicz. Post quantum KEM authentication in SPDM for secure session establishment. *IEEE Design & Test*, (early access), 2023. https://ieeexplore.ieee.org/document/10175549.

Figure 4: Flamegraph for the s2n negotiation in a hybrid TLS handshake

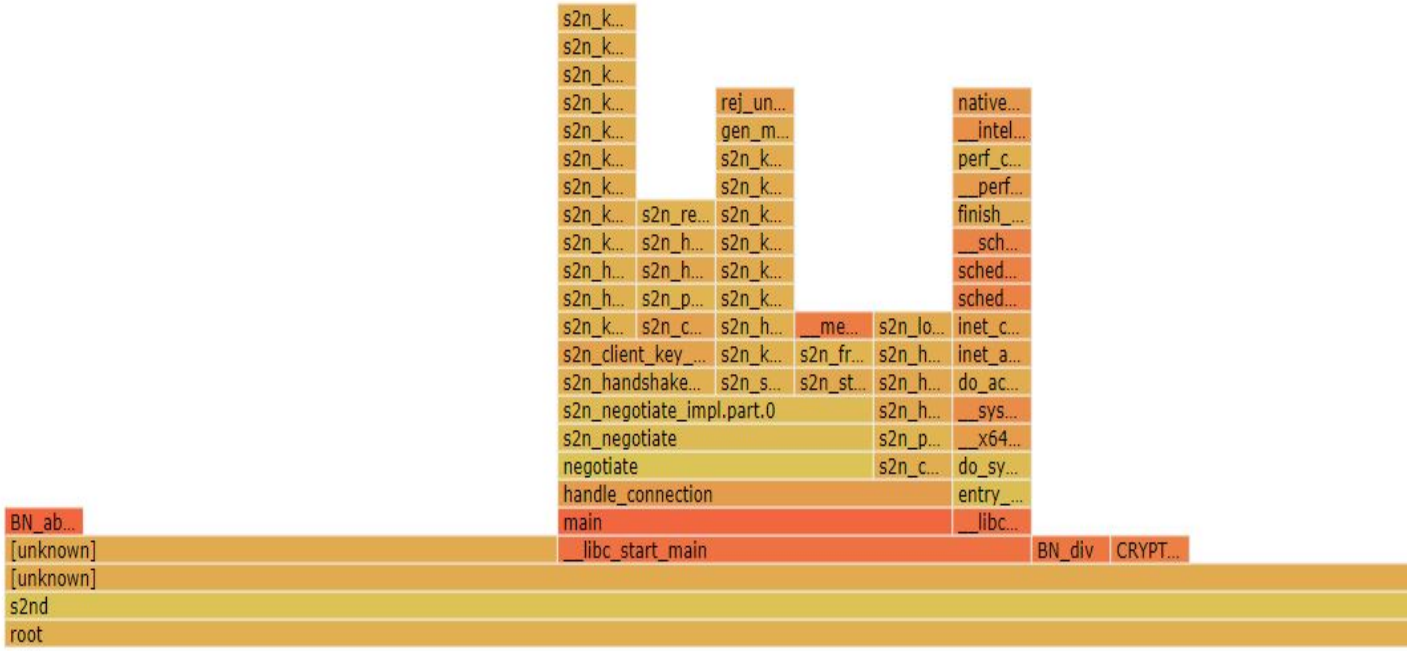


Figure 5: Flamegraph for the s2n negotiation in a traditional TLS handshake