Quantum Threats to the TLS 1.3 Protocol

Luiz Filipi Anderson de Sousa Moura
Informatics and Statistics Institute
Federal University of Santa Catarina
Florianópolis-SC, Brazil
e-mail: luiz.f.s.m@posgrad.ufsc.br

Alexandre Augusto Giron
Computer Engineering Department
Federal University of Technology-Paraná
Toledo-PR, Brazil
e-mail: alexandregiron@utfpr.edu.br

Ricardo Felipe Custódio
Informatics and Statistics Institute
Federal University of Santa Catarina
Florianópolis-SC, Brazil
e-mail: ricardo.custodio@ufsc.br

***Abstract*—Transport Layer Security 1.3 is the latest version available. This protocol is widely used in Internet security, present in more than 60% of all Internet connections based on HTTPS. Quantum computers are a new computational paradigm that threatens information security as we know it, solving mathematical problems used in current cryptography in polynomial time or providing quadratic acceleration for brute force attacks. This paper highlights the quantum threat to Transport Layer Security, focusing on public key cryptography, exposes threat scenarios, propose a detailed attack model to the protocol, shows expected storage requirements to store-now-decrypt-later attacks, and explores ways to mitigate these quantum threats.**

***Index Terms*—Quantum Computing; Transport Layer Security (TLS) 1.3; Store-Now-Decrypt-Later.**

I. Introduction

Transport Layer Security (TLS) 1.3 is a notorious Internet security protocol, present in more than 60% of all Internet connections based on HTTPS [1][2]. TLS is the de-facto standard for securing applications like web servers, browsers, e-mails, messaging, and Voice over Internet Protocol (VoIP), providing end-to-end secure channels with confidentiality, integrity protection, peer authentication, Forward Secrecy (FS), and other [3] guarantees. TLS, like many security protocols, uses Public Key Cryptography (PKC) in its design, for example, for peer authentication and Key Exchange (KEX).

Quantum Computers (QCs) are a threat to PKC since 1994, when Shor's algorithm [4] presented ways to solve the integer factorization problem and Discrete Logarithm Problem (DLP) with an exponential speedup [5]. For symmetric cryptography, the threat is also present with Grover's 1996 [6] search algorithm, capable to perform a brute force attack on a size n list with only $n^{1/2}$ steps, while a classical computer needs about $n/2$ steps for the same task [7].

It is difficult to say when QCs will be ready to break the cryptographic systems we know today. A good guess is about 15 years or less [8]. However, even if the QCs are not yet available, we cannot ignore their threat, as the encrypted data can be stored now and decrypted when the QCs are ready. This process is called "Store-Now-Decrypt-Later" (SNDL) [9].

It is, then, strikingly important to understand the threat to TLS before the arrival of QCs. For this reason, this paper:

- exposes the quantum threat specifically on TLS 1.3, modeling the steps necessary to perform an attack on the PKC;
- predicts the storage requirements for a SNDL attack;
- explores possible threat actors, their capabilities, and post-quantum stages; and
- presents mitigation techniques.

This paper is organized as follows: Section II presents the TLS 1.3 protocol; Section III explores quantum computers and algorithms relevant to perform an attack on PKC; Section IV models the quantum threat on TLS; Section V presents approximate resources for a SNDL attack; Section VI discuss ways to mitigate the threat; and Section VII is a conclusion.

II. TLS 1.3 Protocol

TLS is a secure communication protocol, developed throughout the years, with the current version (1.3) defined in RFC 8446 [3]. The protocol is divided in three parts: the handshake protocol, where most of the PKC operations are employed; the record protocol, securing the application data with symmetric encryption; and the alert protocol, responsible for triggering error messages and counter actions. We focus on the TLS 1.3 handshake messages in this section, since this part is vulnerable to attacks on PKC, except for the Post-Handshake Authentication mode. They are detailed below.

- `ClientHello`: the first message, sent by the client, to start the TLS handshake. It includes a random nonce, protocol versions, list of supported cryptographic algorithms, and extensions. Example extensions are: the `keyshare`, used to carry an Eliptic-Curve Diffie-Hellman Exchange (ECDHE) public key, and the `pre_shared_key`, carrying Pre-Shared Key (PSK) labels.
- `ServerHello`: This message contains a random nonce, the negotiated algorithms, and extensions. Depending on the extensions that the client has sent in the `ClientHello`, the server replies accordingly. Again, examples include a `keyshare` containing an ECDHE public key, and the selected PSK label, depending on what the client demanded. After the `ServerHello` is received, the client and server can derive symmetric keys for encrypting their application data.
- Authentication messages: The most common use case is server authentication (see Figure 1), but client authentication, although optional, can be used (see Figure 2). The messages used for certificate-based authentication are:
  - `Certificate`: this message comprises a set of certificate(s) that identifies one TLS peer. Servers

send this message if not authenticating by PSK. Clients can authenticate as well and in the mutual authentication scenarios they also send their certificate with this message type;
  - `CertificateVerify`: this message comprises a digital signature on the TLS handshake transcript. The transcript is a hash of the handshake messages. `CertificateVerify` is sent only if not authenticating by PSK; and
  - `Finished`: concludes the handshake. Both peers send this message for two purposes: integrity check and key confirmation. A `Finished` is an HMAC of the handshake transcript, using keys derived after the KEX (`ClientHello`, `ServerHello`) so that each peer can verify that they have established symmetric keys correctly.
- `EncryptedExtensions`: this message carries additional server parameters as extensions that are not appended to the `ServerHello` because they are sent encrypted to the client.
- `NewSessionTicket`: this optional message is sent by the server for the Session Resumption feature. After establishing a handshake, the server can optionally send this message, which contains the information required to derive a new PSK. The new PSK can be used for resuming the TLS connection (called Session Resumption), avoiding a complete handshake.

It is worthy to note that all handshake messages are encrypted using keys derived from the ECDHE (or PSK) process, except `ClientHello` and `ServerHello`. Depending on the scenario, some messages are used in contrast to others. The scenarios of Authentication in TLS 1.3 handshakes are [3]: Certificate-based, in which either Server-only or Mutual Authentication types are offered; and Pre-shared-key (PSK), either by Session Resumption or Out-Of-Band (OOB) PSK which authenticates the parties for the session.

Figure 1 shows two types of handshake authentication: Certificate-based and PSK-mode (in a session resumption). The first type is more commonly used, such as in a first-time interaction between the peers. In the PSK-mode, no certificates are sent. Normally, the PSK-mode saves communication bandwidth, but it requires a previous handshake to establish the PSK (or an OOB method). Additionally, a PSK allows sending Zero Round Trip Time Resumption (0-RTT) data, which means that application data is sent together with the `ClientHello` but encrypted using a PSK established in a previous handshake (or by OOB). Note that PSK can be used in conjunction to ECDHE, allowing Forward Secrecy (RFC 8446, Section E.1 [3]).

Figure 2 shows the scenarios for client authentication. The server can request the client certificate and corresponding signature within the handshake (Mutual Authentication) or after the handshake, depending on the desired policy. For example, the server can establish the handshake, at its discretion, without client authentication, or abort the connection.

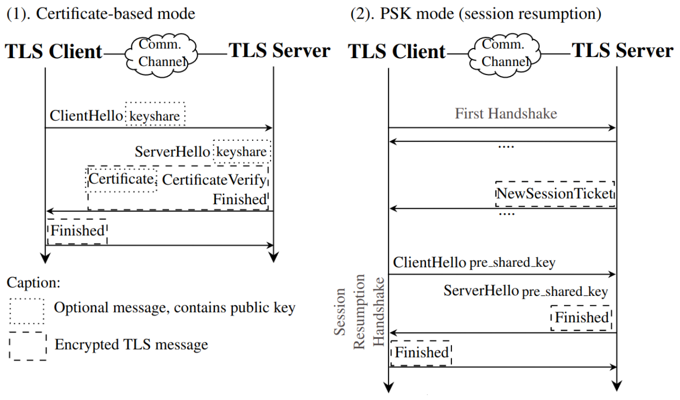
(1). Certificate-based mode
(2). PSK mode (session resumption)
TLS Client
Comm. Channel
TLS Server
ClientHello keyshare
ServerHello keyshare
Certificate, CertificateVerify, Finished
Finished
First Handshake
NewSessionTicket
Session Resumption Handshake
ClientHello pre_shared_key
ServerHello pre_shared_key
Finished
Caption:
Optional message, contains public key
Encrypted TLS message

Fig. 1. TLS 1.3 Handshake Authentication types.

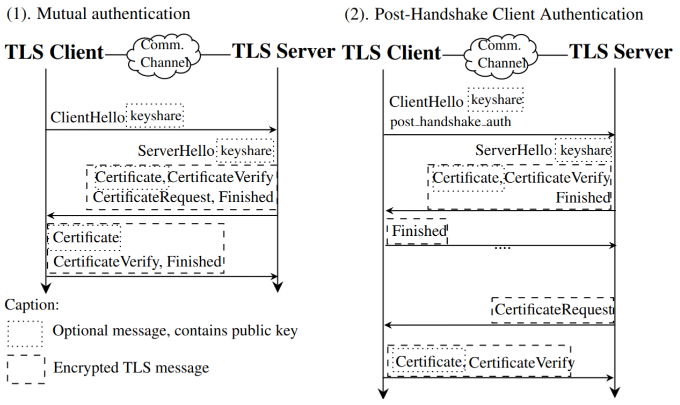
(1). Mutual authentication
(2). Post-Handshake Client Authentication
TLS Client
Comm. Channel
TLS Server
ClientHello keyshare
ServerHello keyshare
Certificate, CertificateVerify
CertificateRequest, Finished
Certificate
CertificateVerify, Finished
ClientHello keyshare post_handshake_auth
Certificate, CertificateVerify, Finished
Finished
CertificateRequest
Certificate, CertificateVerify
Caption:
Optional message, contains public key
Encrypted TLS message

Fig. 2. TLS 1.3 Client Authentication types.

TLS 1.3 implements a key-derivation method called Key Schedule, responsible for deriving and updating encryption keys, and based on the OPTLS protocol [10]. Basically, it takes as input a shared secret after the KEX process and uses a key derivation method to derive new secrets that will then be derived into keying material for encryption. Each type is used for a particular purpose and has different input labels. For example, the keys for encrypting client-to-server are different from the keys used to server-to-client application messages.

III. Quantum Computer and Algorithms

In general, a QC is a computer that takes advantage of quantum mechanics for its computation. A quantum bit (qubit) is a linear combination of two states — ground and excited, hence represented by $|\psi\rangle = a|0\rangle + b|1\rangle$, being on state 0, on state 1, or in a superposition of both states in its quantum state. After measurement, the qubit's wave function collapses to one of the two states [7][11]. When measuring a qubit, the probabilities of it to collapse to state 0 or state 1 must sum 1, so we say it is normalized, $|a|^2 + |b|^2 = 1$. A quantum register is a physical system containing a set of qubits in sequence. The state of a quantum register is a tensor product of the states of each qubit within: $|q\rangle = |q_1\rangle \otimes |q_0\rangle = c_{00}|00\rangle + c_{01}|01\rangle + c_{10}|10\rangle + c_{11}|11\rangle$ [11].

A circuit model QC uses the initial state of a quantum register as the input (usually a sequence of qubits in the ground state). Then, a number of gates (G) are applied to the quantum register to perform a computational step. After a sequence of computational steps is finished, the result is measured as the output. When the last gate is applied, the quantum register goes to state $|q_n\rangle = G^n |q_{n-1}\rangle$ [11].

Another model of quantum computation is quantum annealing or adiabatic quantum computing. This type of computation is very suitable for optimization (minimization) problems of a cost function. It functions by preparing an initial quantum register in equal superposition of all configurations. Then an adiabatically slow time evolution of the state is applied, changing the system to a final Hamiltonian, from which the solution can be extracted [12][13].

In quantum cryptanalysis, the most discussed algorithm is Shor's algorithm. A period finding algorithm that provides an exponential speedup for solving factorization and DLP based problems [7][14] with some newer implementations extending its usability to ECDLP [15][16]. However, there are some challenges in implementing it. The most economic implementation of Shor's algorithm requires $2n + 1$ qubits and roughly $n^3 \log n$ gates to factorize an n-bits long number [17]. It means the most economic implementation of Shor's algorithm in terms of the number of qubits will require 4097 stable qubits and billions of gates to break a 2048-bits Rivest-Shamir-Adleman (RSA) key, still far beyond the current IBM's 433-qubits universal QC Osprey. And even the 4158-qubits QC IBM has plans to release in 2025 [18] will not be enough due to errors, what can be mitigated executing the calculations multiple times or combining circa 1568 noisy qubits into each perfect logical qubit [19]. Another thing to consider is the gate time, which depends on the technology used. On average, a superconductor QC has a gate time of 25 ns, one built on neutral atoms has a gate time of 19 μs, and one built on trapped ions has a gate time of 32 μs [20]. It is clear that the key length to decrypt does change the final execution time by raising the number of gates. If the execution time is longer than the coherence time of the qubits, it will not be able to complete the algorithm.

There are also implementations of factoring algorithms using adiabatic QC that presented good results [21][22][23], and implementations to solve the discrete logarithm problem using adiabatic QC as well [24]. Table I, adapted from [14][17][25], shows some of the best achievements in factoring RSA public keys for quantum computers.

TABLE I. Achievements in factoring RSA public keys.

Year	Key Length	Algorithm
2001	4 bits	Shor
2012	5 bits	Shor
2012	16 bits	Adiabatic
2016	18 bits	Adiabatic
2018	19 bits	Adiabatic
2019	20 bits	Adiabatic
2020	41 bits	Adiabatic

IV. Threat Model and Attack Scenarios

We hereby use the terms "pre-quantum era" for the era we are now, when quantum computers are still not powerful enough for an effective break on cryptography, and "post-quantum era" for the time after t, when the quantum computers will be already an effective threat. Following the idea of [25], it is possible to name a few quantum threat actors. For a better modeling, it is necessary to subdivide the post-quantum era in three:

- **Initial post-quantum era**: QCs are slow, gate times are high, and coherence times are low. Only a few qubits are available. The cost to perform an attack is high, and the skill level necessary is also high.
- **Intermediate post-quantum era**: The quantum hardware, price, and skill level to perform an attack are at an intermediate stage.
- **Advanced post-quantum era**: The QC is fully established and available. The cost to perform and attack is low and the wide range of algorithms, frameworks, and libraries available make the skill level necessary for an attack much lower.

The threat actors can also be further divided according to the total resources available for an attack, in terms of money and number of personnel. Governments and large organizations play the biggest threat here, followed by hacker groups and small organizations, and an individual playing the smallest threat. Each one of the threat actors also have a certain skill level (3, 2, 1, or no threat) on applying quantum algorithms on cryptographic scheme and other skills relevant for a security attack. Table II correlates threat actors and skill levels to the respective post-quantum era at which they become a threat. From the table, it is clear to see that threat actors with more available resources and skill level become a threat earlier.

TABLE II. Correlating threat actors and its skills to post-quantum era.

Available Resources	Skill Level	Becomes a Threat at Which Post-Quantum Era?
Governments and Large Organizations	3	Initial
	2	Intermediate
	1	Advanced
Hacker Groups and Small Organizations	3	Intermediate
	2	Advanced
	1	∞
Individuals	3	Advanced
	2	∞
	1	∞

For a prediction of the time t, which is the threshold from the pre-quantum to the post-quantum era, a big number of the respondents in [8] came up with 15 years or less as the time we have left within the pre-quantum era. So, it is plausible to think that the post-quantum era will begin before 2038.

Two main capabilities of a quantum attacker on TLS are impersonation and breaking confidentiality. Impersonation is the capability a threat actor has to authenticate as another client or as another server; and breaking confidentiality means the threat actor is able to read confidential messages. The attacker can also perform the attack in two different ways: passive, listening to the network channel; and active, modifying the communication. With this in mind, impersonation will always be an active attack, whilst breaking confidentiality can be both.

A quantum attack on TLS 1.3 should follow these steps to break confidentiality, on each one of the handshake types (Figures 1 and 2):

- **Certificate-based (server)**:
  1) collect `Client` and `ServerHello` (CH and SH, respectively), extracting the public keys epk_{CH} and epk_{SH} present in `keyshare` messages;
  2) use Shor's algorithm for ECDLP to break the KEX: it computes the private key from epk_{ch} or epk_{sh} in order to recover the ephemeral private key; and
  3) use the recovered ephemeral key to derive the symmetrical keys, using the TLS Key Schedule [3] (Section 7), allowing to decrypt the whole communication.
- **Mutual Authentication**: same as previous.
- **Post-Handshake Auth.**: same as previous.
- **PSK-based resumption**:
  1) use the steps 1-3 above on the First Handshake (right part of Figure 2);
  2) use the recovered ephemeral key to derive the symmetrical keys used throughout the communication;
  3) decrypt the `NewSessionTicket` message, recovering the ticket information (such as nonces and labels);
  4) use the recovered information to derive the resumption PSK; and
  5) use the PSK to derive the second handshake's (Session Resumption) symmetrical keys, allowing to violate confidentiality of the resumed connection.

To achieve impersonation, a quantum attacker has to follow these steps:

- **Certificate-Based (server)**:
  1) collect `Client` and `ServerHello`, extracting the public keys epk_{CH} and epk_{SH} present in `keyshare` messages;
  2) use Shor's algorithm for ECDLP to break the KEX: it computes the private key from epk_{ch} or epk_{sh} in order to recover the ephemeral private key;
  3) use one of the recovered private keys to derive the symmetrical keys, using the TLS Key Schedule [3] (Section 7), and then decrypt the authentication messages (which contains `Certificate`, `CertificateVerify`, and `Finished`); and
  4) use one of the alternatives to attack the `Certificate` message and return the certificate private key:
     - use Shor's algorithm or adiabatic QC to solve the factorization problem on the RSA public key; or
     - use Shor for ECDLP on the public key based on elliptic curves.
- **Mutual Authentication**: same as for server authentication mode, but the attacker can choose to impersonate server or client. The main difference is the target `Certificate` message (from the server or client).
- **Post-Handshake Auth.**: impersonate the server is similar to the previous modes, but to impersonate client:
  1) check the presence of `post_handshake_auth` extension;
  2) use the steps 1-2 of the Certificate-based authentication (server);
  3) decrypt the communication using the recovered symmetric keys, searching for the `CertificateRequest` message; and
  4) use one of the alternatives to attack the client's `Certificate` message and return the private key:
     - solve the factorization problem with Shor's algorithm or adiabatic QC; or
     - use Shor for ECDLP instead.
- **PSK-based resumption**: similar steps as used for server authentication mode, but the steps should be applied to the First Handshake. Having the PSK information, the attacker can impersonate both peers. However, PSKs duration time can be limited up to 7 days [3], so, the attack window is limited.

Applying these methods to impersonate a client, the threat actor can pretend to be another person to the server, getting access to confidential information. Impersonating the server makes a client or a set of clients think they are sending data to a trustworthy server, who is receiving confidential data from the users. When the threat actor applies these methods to break confidentiality, it is possible that the attacker only listens to the communication channel or actively communicate to the other side, in both ways, causing harm to the victim.

V. RESOURCES FOR A SNDL ATTACK

If the threat actor decides to store packets now to decrypt in the post-quantum era, the attacker will face the problem of storing days, weeks, months, or years of information. Table III brings expected storage cost for SNDL attacks against some of the most accesses websites worldwide. The packets were collected with Wireshark using the URL's IP address as a filter, filtering all the connection between a logged user and the server, simulating a man in the middle attack.

TABLE III. Estimating the SNDL storage requirements.

Site	1 h of captured packets (MB)	Expected Storage Cost for 24 h (GB)	Expected Storage Cost for 1 y (TB)
instagram.com	835.4	19.6	7
youtube.com	723.7	17	6
amazon.com	272.6	6.4	2.3
gmail.com	124.8	2.9	1

As it is possible to infer, SNDL attacks require a large amount of storage, depending on the nature of the content to be captured, the timespan of the capture, and the number of victims. Table III considers the attack against a single victim, for more victims the expected value is multiplied by the number of victims. E.g., to store 1 y of packets from 200 GMail users the necessary storage would be 200 $\times$ 1 TB =

200 TB; and from Amazon users the storage would be 200 × 2.3 TB = 460 TB. Of course this number cannot be exact, due to different navigation profiles, but it is approximate.

VI. Mitigation and Discussion

Quantum cryptography is the use of quantum physics to create a different class of cryptography. The simplest example is the use of quantum superposition in order to provide a perfectly random number, but the most common example in this class is the Quantum Key Distribution (QKD) [25][26]. The most known QKD protocol is BB84 [27], but there are many others, as can be found on [28].

- **Pros of implementing QKD**:
  - the mathematics of quantum mechanics guarantees the key exchange is perfectly secure; and
  - the no-copy property of quantum mechanics ensures there will be no man-in-the-middle attack, because a measurement of the system would modify it.
- **Cons of implementing QKD**:
  - the no-copy property makes it impossible to re-rout or broadcast a qubit, making it necessary to develop special network channels and hardware for QKD;
  - it is affected by decoherence and longer travels might be impossible. Most of the current QKD systems do not allow travels further than 200 km [28]; and
  - implementation cost immensely for large networks. Making it a viable solution only for limited use cases.

Post-Quantum Cryptography (PQC) rely on mathematics running on classical devices that are not easily solvable by a QC. NIST announced in 2022 four PQC algorithms promised to be quantum-safe: CRYSTALS-Kyber [29], a key-encapsulation mechanism that can be used to establish symmetric keys for TLS or other protocols; CRYSTALS-Dilithium [30], a digital signature algorithm; Falcon [31], another method for digital signatures; and SPHINCS+ [32], a hash-based digital signature algorithm. There is also the Open Quantum Safe Project, an open-source project created to evaluate PQC candidates and to prototype their use in protocols like TLS 1.3 [33].

- **Pros of implementing PQC**:
  - a more viable solution for KEX than QKD; and
  - there are also implementations for digital signatures.
- **Cons of implementing PQC**:
  - PQC algorithms have been tested for years, but it is still impossible to tell for how long they will remain unbreakable [28]; and
  - most of the existent PQC algorithms are slower or require larger keys than the most common classical algorithms for KEX or digital signature, impacting in slower page loads and a risk of packet loss.

There are also hybrid implementations that combine a pre-quantum cryptography with a post-quantum one. To exemplify, a hybrid KEX scheme can be achieved by combining the output of a pre-quantum algorithm and a post-quantum one with an XOR operation. An example of how it could be applied in TLS 1.3 can be found at [34]. For the case of a hybrid digital signature, it is possible to create two signatures, one with a pre-quantum algorithm and another with a post-quantum one [35].

Some newer implementations also tackle the problem by using Post-Quantum Key-Encapsulation Mechanisms (PQKEM) [36] or lattice-based cryptography [37].

Other than the previous presented alternatives, there are approaches that are not a definitive solution, but can make the attacker give up on the attack by diminishing the return or raising the investment. From Section 3, it is possible to infer that the key length influences both the number of gates and the number of qubits needed for decryption, also, each gate consumes a certain amount of processing time. The number of qubits and the processing time can raise the attack cost, and further, a very long key may make the attack time longer than the coherence time, turning the attack impractical. E.g., using a QC to break RSA would be impractical if the key length is 8 KB [38].

The Extended Triple Diffie-Hellman (X3DH) key agreement protocol [39], present in the Signal protocol, provide multiple key exchanges in parallel, what can drive up the attack requirements, since the QC ought to be used for each encryption layer. Other algorithms like this, that add new encryption layers, might be a good idea, or tunneling a TLS connection to another encrypted Virtual Private Network (VPN) [25].

Diminishing the attack return can be achieved by diminishing the amount of data recovered on each attack, making it necessary to perform the same attack multiple times or conforming to have limited information extracted. Security controls such as Perfect Forward Secrecy (PFS) and Post Compromise Security (PCS) can limit the attacker's access to information [25]. Other examples are the Double Ratchet key management algorithm [40] also present in the Signal protocol, and short-term certificates like the one present in the ACME protocol [41]. Short-term certificates are a good mitigation due to the short attack window available to both break the certificate's algorithm and, consequently, impersonate.

SNDL attacks are more urgent, since malicious individuals might be storing information now to decrypt later. A way for an organization to decide a good moment to migrate to quantum safe cryptography is given by [8]. Also, the company has to be aware of social engineering attacks, because, as Table III portrayed, the amount of storage necessary for SNDL attacks is huge, and the threat actor may use social engineering to better filter which packets on which specific date to collect.

VII. Conclusion and Future Work

This paper exposed the threat of QC on TLS 1.3, presenting: existing quantum algorithms for an attack against PKC, and achievements on practical implementations of these algorithms; detailed steps to perform a quantum attack against the PKC in different TLS 1.3 handshake modes; approximate storage requirements for a practical SNDL attack; existing and new mitigation methods to avoid such attacks.

Section III brought different implementations of Shor's algorithm and adiabatic alternatives, but no adiabatic implementation for the ECDLP were found. As showed in Section III, the adiabatic implementations are outperforming on the factorization, and not having an adiabatic alternative for the ECDLP can delay the threat to TLS in some years.

Section IV exposed threat actors and the steps necessary to attack PKC in TLS 1.3. From Section VI, KEX can be done securely with QKD, at the cost of a large investment and research, making PQC and hybrid solutions a more viable way to achieve KEX and digital signatures, however, at the cost of losing performance. Section VI also presented more immediate forms of mitigation that can be implemented before changing drastically the TLS 1.3 infrastructure.

SNDL attacks are a more immediate threat, but, as Section V showed, they require a huge amount of storage. As a sidenote: it is important to notice SNDL is a threat not limited to the advent of QC, since technology naturally evolves.

Future work to complement this one can be done by studying different attacks against TLS 1.3.

References

[1] C.-l. Chan, R. Fontugne, K. Cho, and S. Goto, "Monitoring tls adoption using backbone and edge traffic," in: IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), IEEE, 2018, pp. 208–213.

[2] D. Sikeridis, P. Kampanakis, and M. Devetsikiotis, "Post-quantum authentication in tls 1.3: a performance study," Proceedings of the Network and Distributed Systems Security (NDSS) Symposium, 2020.

[3] E. Rescorla, "The transport layer security (tls) protocol version 1.3," RFC 8446, August 2018.

[4] P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in: Proceedings 35th annual symposium on foundations of computer science, IEEE, pp. 124–134, 1994.

[5] J.-P. Aumasson, "The impact of quantum computing on cryptography," Computer Fraud & Security, 2017.

[6] L. K. Grover, "A fast quantum mechanical algorithm for database search," in: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing, pp. 212–219, 1996.

[7] V. Mavroeidis, K. Vishi, M. D. Zych, and A. Jøsang, "The impact of quantum computing on present cryptography," unpublished.

[8] M. Mosca and M. Piani, "Quantum threat timeline report 2022," Global Risk Institute, Toronto, ON.

[9] G. Mone, "The quantum threat," Communications of the ACM, 2020.

[10] H. Krawczyk and H. Wee, "The optls protocol and tls 1.3," in: 2016 IEEE European Symposium on Security and Privacy (EuroS&P), IEEE, pp. 81–96, 2016.

[11] P. I. Hagouel and I. G. Karafyllidis, "Quantum computers: Registers, gates and algorithms," in: 2012 28th International Conference on Microelectronics Proceedings, IEEE, pp. 15–21, 2012.

[12] C. R. Laumann, R. Moessner, A. Scardicchio, and S. L. Sondhi, "Quantum annealing: The fastest route to quantum computation?," The European Physical Journal Special Topics, 2015.

[13] S. Yarkoni, E. Raponi, T. Bäck, and S. Schmitt, "Quantum annealing for industry applications: Introduction and review," Reports on Progress in Physics, 2022.

[14] A. Petrenko, Applied Quantum Cryptanalysis, CRC Press, 2023.

[15] J. Proos and C. Zalka, "Shor's discrete logarithm quantum algorithm for elliptic curves," unpublished.

[16] D. Maslov, J. Mathew, D. Cheung, and D. K. Pradhan, "An O(m^2)-depth quantum algorithm for the elliptic curve discrete logarithm problem over gf $(2^m)^a$," Quantum Information & Computation, 2009.

[17] J. Suo, L. Wang, S. Yang, W. Zheng, and J. Zhang, "Quantum algorithms for typical hard problems: a perspective of cryptanalysis," Quantum Information Processing, 2020.

[18] C. Q. Choi, "Ibm's quantum leap: The company will take quantum tech past the 1,000-qubit mark in 2023," IEEE Spectrum, 2023.

[19] C. Gidney and M. Ekerå, "How to factor 2048 bit rsa integers in 8 hours using 20 million noisy qubits," Quantum, 2021.

[20] M. Suchara, A. Faruque, C.-Y. Lai, G. Paz, F. T. Chong, and J. Kubiatowicz, "Comparing the overhead of topological and concatenated quantum error correction," unpublished.

[21] Z. Li, N. S. Dattani, X. Chen, X. Liu, H. Wang, R. Tanburn, H. Chen, X. Peng, and J. Du, "High-fidelity adiabatic quantum computation using the intrinsic hamiltonian of a spin system: Application to the experimental factorization of 291311," unpublished.

[22] S. Jiang, K. A. Britt, A. J. McCaskey, T. S. Humble, and S. Kais, "Quantum annealing for prime factorization," Scientific reports, 2018.

[23] W. Peng, B. Wang, F. Hu, Y. Wang, X. Fang, X. Chen, and C. Wang, "Factoring larger integers with fewer qubits via quantum annealing with optimized parameters," SCIENCE CHINA Physics, Mechanics & Astronomy, 2019.

[24] M. Wroński, "Practical solving of discrete logarithm problem over prime fields using quantum annealing," in: Computational Science–ICCS 2022: 22nd International Conference, London, UK, June 21–23, 2022, Proceedings, Part IV, Springer, pp. 93–106, 2022.

[25] T. Runge, "Dismantling the quantum threat," Ph.D. thesis, Technische Hochschule Brandenburg, 2023.

[26] V. Padamvathi, B. V. Vardhan, and A. Krishna, "Quantum cryptography and quantum key distribution protocols: a survey," in: 2016 IEEE 6th International Conference on Advanced Computing (IACC), IEEE, pp. 556–56, 20162.

[27] G. Brassard and C. H. Bennett, "Quantum cryptography: Public key distribution and coin tossing," in: International conference on computers, systems and signal processing, pp. 175–179, 1984.

[28] G. Xu, J. Mao, E. Sakk, and S. P. Wang, "An overview of quantum-safe approaches: Quantum key distribution and post-quantum cryptography," in: 2023 57th Annual Conference on Information Sciences and Systems (CISS), IEEE, pp. 1–6, 2023.

[29] J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehlé, "Crystals-kyber: a cca-secure module-lattice-based kem," in: 2018 IEEE European Symposium on Security and Privacy (EuroS&P), IEEE, pp. 353–367, 2018.

[30] L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé, "Crystals-dilithium: A lattice-based digital signature scheme, IACR Transactions on Cryptographic Hardware and Embedded Systems," 2018.

[31] P.-A. Fouque, J. Hoffstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Prest, T. Ricosset, G. Seiler, W. Whyte, Z. Zhang, et al., "Falcon: Fast-fourier lattice-based compact signatures over ntru," Submission to the NIST's post-quantum cryptography standardization process, 2019.

[32] D. J. Bernstein, A. Hülsing, S. Kölbl, R. Niederhagen, J. Rijneveld, and P. Schwabe, "The sphincs+ signature framework," in: Proceedings of the 2019 ACM SIGSAC conference on computer and communications security, pp. 2129–2146, 2019.

[33] D. Stebila and M. Mosca, "Post-quantum key exchange for the internet and the open quantum safe project," in: International Conference on Selected Areas in Cryptography, Springer, pp. 14–37, 2016.

[34] D. Stebila, S. Fluhrer, and S. Gueron, "Hybrid key exchange in TLS 1.3," Internet-Draft draft-ietf-tls-hybrid-design-06, Internet Engineering Task Force, work in progress, February 2023.

[35] W. Beullens, J.-P. D'Anvers, A. T. Hülsing, T. Lange, L. Panny, C. de Saint Guilhem, and N. P. Smart, "Post-quantum cryptography: Current state and quantum mitigation," Tech. rep., Eindhoven University of Technology, 2021.

[36] P. Schwabe, D. Stebila, and T. Wiggers, "Post-quantum tls without handshake signatures," in: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, p. 1461–1480, 2020.

[37] J. W. Bos, C. Costello, M. Naehrig, and D. Stebila, "Post-quantum key exchange for the tls protocol from the ring learning with errors problem," in: 2015 IEEE Symposium on Security and Privacy, pp. 553–570, 2015.

[38] K. Li and Q.-y. Cai, "Practical security of rsa against ntc-architecture quantum computing attacks," International Journal of Theoretical Physics, 2021.

[39] M. Marlinspike and T. Perrin, "The x3dh key agreement protocol," Open Whisper Systems, 2016.

[40] T. Perrin and M. Marlinspike, "The double ratchet algorithm," GitHub wiki, 2016.

[41] Y. Sheffer, D. Lopez, O. G. de Dios, A. Pastor, and T. Fossati, ”Support for short-term, automatically renewed (star) certificates in the automated certificate management environment (acme),” RFC 8739.