# The Quantum Menace: An Analysis of Postquantum TLS

**Collin Berman**
cmb5nh@virginia.edu

**Reid Bixler**
rmb3yz@virginia.edu

## Abstract

With the introduction of quantum computing, the inherent security of the TLS protocol becomes endangered. Risks of losing the privacy and integrity of communications over the internet are too great to not consider preventative measures against quantum attacks. We look at the current key exchange protocol for TLS to determine the flaws caused by quantum computers. Our research focuses on analysing current alternatives that are quantum secure as well as other cryptographic methods that have not yet been considered for a post-quantum TLS. We offer final recommendations on where the TLS community should go from now in order to mitigate the risk of the quantum menace.

## 1 Introduction

The primary goal of the TLS protocol is to provide privacy and data integrity between two communicating applications (Dierks and Rescorla, 2008). With TLS, clients and servers are able to securely communicate with one another with little to no risk of having a third party read or alter their messages. TLS is used widely across the internet and is essentially the defacto standard for secure communications, often even being implemented into applications where an alternative protocol may be more useful. In order of priority, the makers of TLS desire cryptographic security, interoperability, extensibility, and relative efficiency. The security behind TLS is based off of a number of secure algorithms based off of hashing, symmetric cryptography, prime-factorization, elliptic curves, and other cryptographic methods.

Notably, there are essentially two portions of the TLS protocol that use these algorithms for either key exchange or for authentication. The key exchange allows for the two users to initiate a channel of communication and securely talk with one another. The authentication allows for proving that users are who they say that they are. While TLS is considered secure by todays standards, there are actually some risks that most either do not realize or do not care about. That is, the Quantum Menace. With the introduction of quantum computing, the inherent security of many widespread algorithms is at risk of being broken. Specifically for TLS, things such as ECDH, RSA, and DSA can be brute forced with much better efficiency with a quantum algorithm known as Shors Algorithm. Luckily enough, quantum computers are still in their early stages and some suggest that quantum computers will never be powerful enough to break our current level cryptography. However, because we as security researchers would rather focus on proactive measures rather than reactive ones, we want to find alternatives before there is capable quantum computing.

Specifically, we are going to focus on finding solutions for key exchange rather than authentication, because a broken key exchange allows for attackers to read past communications. In terms of security of the communication over the internet, focusing on privacy is more essential than data integrity. In this paper, we go about analyzing a number of post-quantum secure TLS implementations as well as potential alternative cryptographic methods that have not yet been implemented into TLS. We finalize this analysis with a comparison between post-quantum TLS methods in order to determine where the security community should be heading towards.

## 2 Post Quantum Computing

Quantum computing is slowly gaining traction since it was theorized in 1982 by Richard Feynman. It relies on quantum principles and super positioning of particles to perform computational operations significantly faster than classical computers (Feynman, 1982). There already exist quantum algorithms, which use quantum logic gates to solve problems that were once thought computationally unsolvable. One of these algorithms, Shor's Algorithm, is able to easily determine the prime factors of a large number (i.e. integer factorization) within a polynomial time (Shor, 1999). Before this algorithm, integer factorization was considered non-polynomial, also known as NP, such that given a NP problem, one could verify its solutions quickly but it could not be solved within a reasonable time, often taking longer than the expected lifespan of the universe to solve (Ladner, 1975). This is a major problem for the security of many of today's technologies which rely on the hardness of prime factorization.

While quantum computers are only in their infantile stage, the potential security risks behind broken encryption schemes from quantum algorithms would be unfathomable in today's world. If anybody were able to obtain a quantum computer capable of performing significant computations, breaking into many secure systems would be completely feasible. As mentioned earlier, TLS uses ECDH, RSA, and DSA for the key exchange protocol in TLS, to name a few. ECDH is reliant on elliptic curve cryptography, which requires low computing power for implementation but massive amounts to brute force an attempt to break it. However, a modified Shor's algorithm could notably decrease the computational requirement for a brute force attempt (Sullivan, 2013). With a working quantum computer capable of significant computation, any current and previous TLS communications could be viewed by attacking the key exchange protocol.

However, there still exist a number of cryptographic systems that are actually secure against quantum attacks. These include: hash-based cryptography, code-based cryptography, lattice-based cryptography, multivariate-quadratic-equations cryptography, and secret-key cryptography.

### 2.1 Hash-Based Cryptography

With hashing, when given an input x there will be an output y that should not be able to easily go back to find x. Before the interest in postquantum security, many researchers didnt focus much on hash-based cryptography because of the limit on the number of signatures that can exist for a generated key, even with the existence of logarithmically scaling Merkle trees. Also, with hashing, we can turn one-time hashing into multiple by adding new public keys in each message, which is also known as chaining. By doing this, the signature of the nth message includes all n-1 previous signed messages. With Merkle Trees, it is possible to have provable reductions that say that the security of the single-signing function will be the same security as the tree. One of the potential drawbacks of hashing is the fact that over time, researchers will eventually find ways to produce collisions, which can map two different inputs to the same output. This is why the security community often is always looking for a new and better hash function.

### 2.2 Code-Based Cryptography

The methods often considered with code-based cryptography is McEliece and Niederreiter. With these, the public key is made up of a dt x n matrix K and a message m is encrypted by multiplying this matrix K by m. The receiver will receive a Hidden Goppa code to decrypt this message. These cryptographic systems have extremely efficient key generation, encryption, and decryption. However, often the problem is the very long public keys required.

### 2.3 Lattice-Based Cryptography

Lattices have gotten a lot of attention from the postquantum community and a majority of current quantum-secure systems are done with lattice cryptosystems. Lattices are reliant on the Shortest Vector Problem and the Closest Vector Problem, which have not yet been reduced to a polynomial time to break. There are a number of different methods including NTRU, LWE, and BLISS.

### 2.4 Multivariate-Quadratic-Equations Cryptography

These rely on a sequence of polynomials and variables with coefficients. Each polynomial is required to have a degree of at most 2, with no squared terms. Its possible to verify these signatures with a standard hash function, which will result in shorter public keys. There are quite a number of current methods including: Rainbow, Hidden Field Equations (HFE), and UOV Cryptosystems. One of the major problems with these systems is that, while efficient, the security is not fully understood and new attacks are found on a regular basis.

### 2.5 Secret-Key Cryptography

Otherwise known as symmetric cryptography, this uses either Stream Ciphers or Block Ciphers. These cryptosystems are widely used and implemented including: Twofish, Serpent, AES (Rijndael), Blowfish, CAST5, Kuznyechik, RC4, 3DES, Skipjack, Safer+/++ (Bluetooth), and IDEA. There already exist some symmetric key management systems like Kerberos and 3GPP, which offers the benefit of already being widespread.

## 3 PQC in TLS

### 3.1 TLS Specifics

Although many post-quantum cryptosystems have been proposed by researchers, not all are ready for widespread deployment. Cryptographic software needs to be carefully written to avoid so called side-channel attacks, which can introduce vulnerabilities into an otherwise secure protocol. Furthermore, in order for a cryptographic library to be useful to the larger programming community, it should expose a simple interface to allow easy integration into software projects.

### 3.2 Criteria

Although many post-quantum cryptosystems have been proposed by researchers, not all are ready for widespread deployment in TLS. When developers need to include a secure communication channel in their software, they use libraries that expose a high-level interface, rather than directly calling cryptographic primitives themselves. Therefore, a post-quantum key-exchange protocol is likely to be integrated into software only if is packaged into such a library, like OpenSSL. Furthermore, cryptography code needs to be written to be resistant to timing attacks, which allow an adversary to recover secret data from a cryptosystem even if the protocol itself is theoretically secure. Attackers targeting a software implementation have access to additional information than security proofs model. Such vulnerabilities often arise from altering program control-flow based on secret data, for example performing a computation only when a certain bit of the secret key is set.

TLS implementations have been successfully exploited by timing-attacks. For example, Canvel, Hiltgen, Vaudenay, and Vaugnoux (Canvel et al., 2003) were able to intercept passwords in a TLS channel by measuring differences in timing arising from whether or not a forged message authentication code (MAC) is valid. This attack was possible even though Krawczyk (Krawczyk, 2001) had proven that the way MACs are validated is secure.

Due to the effectiveness of these attacks, we only consider a protocol to be ready for TLS deployment if it has been implementing to be resistant to timing attacks. Programmers typically achieve protection against timing attacks by writing constant-time programs, that is, programs whose runtime do not depend on secret day. This involves avoiding branching on secret data as described above, but also requires avoiding memory accesses based on secret data, since recently-accessed data stored in the cache can be read faster than data from main memory (Bernstein, 2005).

## 4 Current Implementations

In this section we investigate post-quantum cryptosystems which have implementations targeting TLS.

In 2015, Bos, Costello, Naehrig, and Stebila (Bos et al., 2015) constructed TLS ciphersuites with post-quantum key-exchange. Their key-exchange protocol is based on the ring learning with errors (R-LWE) problem, which is related to the shortest vector problem on lattices. The authors integrate four ciphersuites targeting the 128-bit security level into OpenSSL. The first two, RLWE-ECDSA-AES128-GCM-SHA256 and RLWE-RSA-AES128-GCM-SHA256, simply replace a classical key-exchange protocol with their post-quantum protocol. The other two are hybrid ciphersuites, which combine their R-LWE protocol with elliptic curve DiffieHellman key exchange. Their performance results, in HTTPS connections per second, are shown in Figure 1.

Later in 2015, Alkim, Ducas, Pppelmann, and Schwabe identified a number of performance and security problems with the BCNS protocol. By performing a more detailed security analysis, the authors were able to optimize the parameters of the protocol, resulting in higher efficiency and security. Further, the authors suggest replacing a fixed public parameter with a randomly chosen one for each key exchange, avoiding Logjam-style precomputation attacks (Adrian et al., 2015). The authors provide a C implementation of their

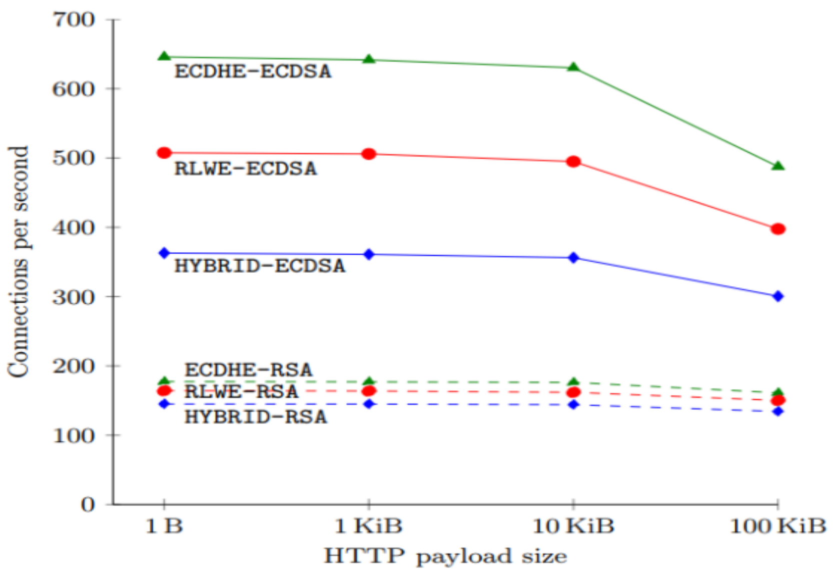

Figure 1: BCNS HTTPS connections per second supported by a web server (Bos et al., 2015)

new protocol, NewHope, but do not integrate their work into OpenSSL.

The absence of a NewHope-based TLS ciphersuite implementation did not stop Google from integrating it into their Chrome web browser (Langley, 2016). To retain classical security, the Chrome team combined NewHope with the X25519 elliptic curve DiffieHellman key-exchange protocol, creating a hybrid scheme they called CECPQ1. Because the project was only an experiment, CECPQ1 has since been removed from Chrome, but the expriment was successful. Adding an additional key-exchange protocol to TLS resulted in increased packet sizes and latencies, but this was expected. No unexpected problems arose, showing that deploying NewHope in TLS is feasible.

Because NewHope is such a new protocol, the Chrome engineers did not want their CECPQ1 protocol to become a standard (Langley, 2016). In fact, recent research has made improvements to the NewHope key-exchange protocol. Bos et al. (Bos et al., 2016) suggest basing cryptosystems on the learning with errors (LWE) problem, rather than R-LWE, developing a protocol called Frodo, which they integrate into OpenSSL. Although R-LWE allows for more efficient protocols, the additional ring structure may introduce new vulnerabilities. On the other hand, Longa and Naehrig (Longa and Naehrig, 2016) develop new algorithms which speed up NewHope by a factor of up to 1.4. A library implementing these, LatticeCrypto techniques is integrated into OpenSSL (Microsoft, 2016).

An alternative lattice-based cryptosystem, NTRU, was described by Hoffstein, Pipher, and Silverman in 1998. NTRU was patented for most of its lifetime, but was recently made patent-free (**?**), enabling wider adoption of the public-key encryption system. Recent research by Bernstein, Chuengsatiansup, Lange, and van Vredendaal (Bernstein et al., 2016) has improved the security and performance of the cryptosystem. NTRU has been integrated into the wolfSSL embedded TLS library (wolfSSL, 2015).

Moving beyond lattice-based cryptography brings us to the supersingular isogeny DiffieHellman (SIDH) key-exchange protocol. Microsoft researchers Costello, Longa, and Naehrig (Costello et al., 2016a) give efficient algorithms for SIDH, which have been integrated into a patch for OpenSSL (Microsoft, 2016a). This library also integrates the SIDH public-key compression algorithms developed by Costello et al. (Costello et al., 2016b).

## 5 Possible Alternatives

The protocols described in the previous section have implementations specifically built for TLS. Many have been built for OpenSSL, providing software developers an easy way to integrate these post-quantum key-exchange protocols into their applications. In this section, we look at

| | TLS-Ready | Security (bits) | Communication (bits) | Keygen (ms) | Constant Time |
|---|---|---|---|---|---|
| NewHope | X | 199 | 3,872 | 0.31 | X |
| FRODO | X | 130 | 22,584 | 2.6 | X |
| LatticeCrypto | X | 128 | 3,872 | 0.21 | X |
| SIDH | X | 128 | 660 | 900 | X |
| wolfSSL NTRU | | 128 | 1,128 | 2.249 | |
| NTRU Prime | | 195 | 9,802 | N/A | X |
| McBits | | 128 | 1,046,738 | N/A | X |
| Algebraic Eraser | | 120 | 1,554 | N/A | |
| Ore DiffieHellman | | 111 | 1,027,000 | N/A | |

Table 1: Postquantum TLS Analysis

post-quantum key-exchange protocols that, while promising, are not ready for deployment in TLS.

Code-based cryptography provides an alternative to lattice-based schemes such as NTRU and the R-LWE protocols described in the previous section. The best example of code-based cryptography is McElieces (McEliece, 1978) hidden-Goppa-code public-key encryption system. Bernstein presented a constant-time implementation of the cryptosystem in 2013. This cryptosystem supports extremely fast encryption, but the public keys are far too large for usage in TLS.

Another alternative to lattice-based cryptography is group-theoretic cryptography. Because the computational problems used in these cryptosystems have not been well-studied, interest in non-commutative cryptography remains for the most part theoretical. Nonetheless, a key-exchange protocol, Algebraic Eraser, has been designed for use in systems with limited computational resources (Anshel et al., 2006). The parameters given in the original article were broken by Ben-Zvi, Blackburn, and Tsaban in 2016, but the protocol is defined in great generality, and Anshel, Atkins, Goldfeld, and Gunnells (Anshel et al., 2016) were able to defeat the attack by giving a new instantiation of the system. Further research is needed to decide whether this protocol is secure.

As a final key-exchange alternative to lattice-based cryptography, we discuss multivariate-quadratic-equations cryptography. This class of cryptosystems comprises for the most part public-key signature schemes, but Burger and Heinle presented a key-exchange protocol based on multivariate Ore polynomials in 2014. However, this cryptosystem has received little scrutiny by the broader cryptography community, and uses public keys nearly as large as those in McElieces code-based system.

## 6 PQTLS Analysis

In this section, we evaluate the cryptosystems surveyed above. The most important factor for a protocol is whether it is ready to be deployed for TLS based on the existence of a constant-time SSL library. We also analyze the performance of each scheme based on the communication and key generation costs for the recommended parameters.

We collect our data in Table 1. All of the protocols discussed in the current implementations section 4 except for NTRU are ready for TLS deployment. Even though NTRU is implemented in wolfSSL, the lack of constant-time protections in their code disqualifies them from being TLS-ready.

Of the cryptosystems ready for use in TLS, SIDH requires the least data transfer, while LatticeCrypto provides the fastest key generation. However, while the communication cost of LatticeCrypto is less than 6 times that of SIDH, key generation in SIDH takes over 4,000 times as long as in LatticeCrypto. Therefore, we believe LatticeCrypto is currently the best choice for integration into TLS.

Most of the schemes analyzed here target the 128-bit security level. However, due to the difficulty of estimating the post-quantum security of a protocol, some authors choose conservative parameters, giving a comfortable margin above the 128-bit level. We also note that Algebraic Eraser is intended for low-cost, resource-constrained devices. As such, the focus their attention on the 80-bit level, but they do provide parameters for 120 bits of security.

## 7 Conclusion

### 7.1 Future Research

This analysis is meant to be a conglomeration of current knowledge on post-quantum enabled TLS. However, we have distinctly focused on looking at key-exchange rather than authentication for all of the research. There are a large number of interesting applications of quantum secure algorithms in signature schemes that would likely have much different criteria than what we showed here.

In terms of potential future research, additional research on current implementations could be done by running them on a single system for a better runtime analysis. As time goes on, there will likely be more implementations that we did not mention, and at that time it would make sense to compare those new ones using the same criteria and methods. We would like to see more research done on non-lattice-based cryptosystems, simply because the security community right now is focusing very heavily on lattices as a means to quantum security.

### 7.2 Final Remarks

Integrating a post-quantum key-exchange protocol into TLS is necessary for preserving the privacy of todays internet communications. We surveyed the post-quantum cryptography literature for implementations that are ready to be deployed today, and found that Microsofts LatticeCrypto library is the current best choice.

However, future research is likely to produce new attacks against ring learning with errors key-exchange. It is important for the cryptographic research community to continue to investigate alternative protocols. In particular, we need to develop further protocols based on problems besides lattices, in case the latter are shown to be easy solved. Supersingular isogeny DiffieHellman is the most promising non-lattice protocol, but many other classes of cryptographic systems exist, and others likely remain to be discovered.

Quantum computers may not be very capable today, but by planning ahead and integrating post-quantum security as soon as we can, we will be ensuring the privacy and integrity of TLS.

## References

D. Adrian, K. Bhargavan, Z. Durumeric, P. Gaudry, M. Green, J. A. Halderman, and P. Zimmermann. 2015. Imperfect forward secrecy: How diffie-hellman fails in practice. *In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security.*

I. Anshel, M. Anshe, D. Goldfeld, and S. Lemieux. 2006. Key agreement, the algebraic erasertm, and lightweight cryptography.

I. Anshel, D. Atkins, D. Goldfeld, and P. E. Gunnells. 2016. Defeating the ben-zvi, blackburn, and tsaban attack on the algebraic eraser. *arXiv Preprint.*

D. J. Bernstein, C. Chuengsatiansup, T. Lange, and C. van Vredendaal. 2016. Ntru prime. *Cryptology ePrint Archive, Report 2016/461.*

D. J. Bernstein. 2005. Cache-timing attacks on aes. *Cr.Yp.To Antiforgery.*

J. W. Bos, C. Costello M. Naehrig, and D. Stebila. 2015. Post-quantum key exchange for the tls protocol from the ring learning with errors problem. *In 2015 IEEE Symposium on Security and Privacy.*

J. Bos, C. Costello, L. Ducas, I. Mironov, M. Naehrig, V. Nikolaenko, A. Raghunathan, and D. Stebila. 2016. Frodo: Take off the ring! practical, quantum-secure key exchange from lwe. *Cryptology ePrint Archive, Report 2016/659.*

B. Canvel, A. P. Hiltgen, S. Vaudenay, and M. Vuagnoux. 2003. Password interception in a ssl/tls channel. *In D. Boneh (Ed.), Advances in Cryptology – CRYPTO 2003.*

C. Costello, D. Jao, P. Longa, M. Naehrig, J. Renes, and D. Urbanik. 2016a. Efficient compression of sidh public keys. *Cryptology ePrint Archive, Report 2016/963.*

C. Costello, P. Longa, and M. Naehrig. 2016b. Efficient algorithms for supersingular isogeny diffie-hellman. *Cryptology ePrint Archive, Report 2016/413.*

T. Dierks and E. Rescorla. 2008. The transport layer security (tls) protocol version 1.2. *IETF Standards Track.*

R P. Feynman. 1982. Simulating physics with computers. *International journal of theoretical physics.*

H. Krawczyk. 2001. The order of encryption and authentication for protecting communications (or: how secure is ssl?). *In J. Kilian (Ed.), Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology.*

R E. Ladner. 1975. On the structure of polynomial time reducibility. *Journal of the ACM.*

A. Langley. 2016. Cecpq1 results [blog post]. *Imperial Violet.*

P. Longa and M. Naehrig. 2016. Speeding up the number theoretic transform for faster ideal lattice-based cryptography. *Cryptology ePrint Archive, Report 2016/504*.

R. J. McEliece. 1978. A public-key cryptosystem based on algebraic coding theory. *DSN Progress Report*.

Microsoft. 2016. Sidh for open ssl [computer software].

P W. Shor. 1999. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*.

N Sullivan. 2013. A (relatively easy to understand) primer on elliptic curve cryptography. *Ars Technica*.

wolfSSL. 2015. Quantum-safe wolfssl [blog post]. *WolfSSL Blog*.