

Article

The QQUIC Transport Protocol: Quantum-Assisted UDP Internet Connections

Peng Yan  and Nengkun Yu * 

Centre for Quantum Software and Information, Faculty of Engineering and Information Technology, University of Technology Sydney, Sydney 2007, Australia

* Correspondence: nengkunyu@gmail.com

Abstract: Quantum key distribution, initialized in 1984, is a commercialized secure communication method that enables two parties to produce a shared random secret key using quantum mechanics. We propose a QQUIC (Quantum-assisted Quick UDP Internet Connections) transport protocol, which modifies the well-known QUIC transport protocol by employing quantum key distribution instead of the original classical algorithms in the key exchange stage. Due to the provable security of quantum key distribution, the security of the QQUIC key does not depend on computational assumptions. It is possible that, surprisingly, QQUIC can reduce network latency in some circumstances even compared with QUIC. To achieve this, the attached quantum connections are used as the dedicated lines for key generation.

Keywords: transport protocol; QUIC; QKD; handshake; key exchange



Citation: Yan, P.; Yu, N. The QQUIC Transport Protocol: Quantum-Assisted UDP Internet Connections. *Entropy* **2022**, *24*, 1488. <https://doi.org/10.3390/e24101488>

Academic Editor: Leong Chuan Kwek

Received: 8 September 2022

Accepted: 14 October 2022

Published: 18 October 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The problem of generating a secret key between two remote parties affects anyone wishing to use encryption in modern communication. Traditionally, symmetric encryption suffers one enormous shortcoming, it is necessary for either the sender or the recipient to create a key and then send it to the other party. While the key is in transit, it could be stolen or copied by a third party, who can then destroy the encryption. In the 1970s, the Diffie–Hellman key agreement method [1], a method of distributing keys without actually sending the keys themselves, was developed. It provides the basis for a variety of authenticated protocols. For example, Transport Layer Security (TLS) protocol uses the ephemeral Diffie–Hellman to enable forward secrecy.

In modern Internet engineering, significant efforts have been focused on establishing more secure connections by employing and updating TLS protocols. Deployed in 2012 by Google [2], QUIC (pronounced “quick”) is a new general-purpose transport layer network protocol based on UDP to improve the performance of connection-oriented web applications based on the Transmission Control Protocol (TCP) with greatly reduced latency. Unlike TLS over TCP, QUIC sends data as stream frames carried in the QUIC packets by its reconstructed transport layer rather than through the TLS record layer directly. The TLS and QUIC protocols work cooperatively, where TLS mainly provides the TLS handshake and state change messages for the authentication and negotiation of parameters. Nowadays, QUIC is widely used by more than half of all connections within the Chrome web browsers, especially for gaming, streaming media, and VoIP services.

However, both the RSA and Elliptic Curve Diffie–Hellman asymmetric algorithms that set up the TLS exchange are vulnerable to quantum computers big enough to run Shor’s algorithm. For instance, the Diffie–Hellman key agreement itself is a non-authenticated key-agreement protocol. In other words, the security of the Diffie–Hellman key agreement method, as well as RSA, is threatened by Shor’s polynomial-time algorithm for the discrete logarithm problem [3], which can be performed on emergent quantum computers [4–7]. Shor’s algorithm can cause modern asymmetric cryptography to collapse since it is based

on large prime integer factorization or the discrete logarithm problem. Therefore, they are not considered to be quantum-safe or post-quantum cryptographic algorithms.

Using quantum mechanics, Bennett and Brassard introduced a famous quantum key distribution (QKD) method named BB84 [8,9] in 1984, which enables two remote end nodes to establish provably secure random keys. To implement this quantum key distribution, requires quantum encoders and decoders to be capable of preparing and measuring weak coherent quantum states. In [10], the world's first network, the DARPA Quantum Network, was built by BBN, Harvard, and Boston University, which delivered end-to-end network security via high-speed QKD. Another QKD network [11] designed and implemented by the European project SEcure COmmunication based on Quantum Cryptography (SECOQC) was put into operation in Vienna in 2008. Building the 1200-mile quantum communication landline between Beijing and Shanghai in 2016 and the quantum communication satellite (known as Micius) in 2017, China has the world's first space-ground quantum network [12]. Recently, many research teams have succeeded in building and operating more practical quantum cryptographic devices [13–16].

To the best of our knowledge, QKD has not been employed as a component in the design and implementation of modern practical Internet protocols. This is very surprising regarding the fundamental role played by key distribution in communication security. In this paper, we propose the QQUIC transport protocol by carefully adding QKD into the TLS handshake in the well-known QUIC protocol to enhance the generation of shared secrets. We develop an encrypted transport protocol to improve the security and transport performance for HTTPS traffic based on the new TLS 1.3 and quantum key distribution BB84. The basic motivation is that quantum key distribution (QKD) can generate a completely random secret known only to the communication sides, which can be used to improve security significantly and optimize the procedure of key exchange in the conventional transport protocol. The QKD is mainly used to replace the key exchange to create the shared secret in the initial handshake, and the security of the application data after the connection establishment is still undertaken by the TLS. Interestingly, QQUIC can reduce network latency in some circumstances even compared with QUIC.

2. Preliminaries

2.1. Quick UDP Internet Connection

Transport Layer Security (TLS) is designed to achieve the goal of a safe connection between a client and a server by providing privacy, integrity, and authentication for the data transmitted over the network. The security of data can be guaranteed via symmetric cryptography with a known shared key, and public-key cryptography can be used to provide the digital signature and authenticate the identity of the communicating parties, and the message authentication code (MAC) generated for every message can be used to detect message tampering and forgery.

Designed in 2012 by Google, QUIC [2] has undergone great improvements in several aspects, such as stream multiplexing, flow/congestion control, loss recovery, and connection migration. The most significant advantage of QUIC is that it has much lower connection establishment latency compared with the combination of TCP and TLS. QUIC employs the TLS handshake to derive the keys used for data encryption and also relies on TLS for authentication, parameter negotiation, and state change information. Here, we mainly focus on the connection establishment process and briefly restate the full handshake of TLS used in the current QUIC, as shown in Figure 1.

The client sends a ClientHello(CHLO) package to initiate a new handshake. The CHLO package contains information on (a) a random client number R_c to generate the master secret, (b) supported TLS versions and cipher suits, and (c) extensions for session resumption, signature, and compression algorithms. The server responds to the client with a ServerHello (SHLO) package at once, which also contains a random server number R_s , the choice of TLS version and cipher suits, and response for extensions. Then a signed certificate is sent by the server to prove its identity to the client, which contains the server's

long-term public key. Furthermore, a Server Key Exchange message is sent if the public key contained in the server's certificate is not sufficient for the client to finish the key exchange. Finally, a Server Hello Done message is sent to indicate that the server is done and waiting for the client's response.

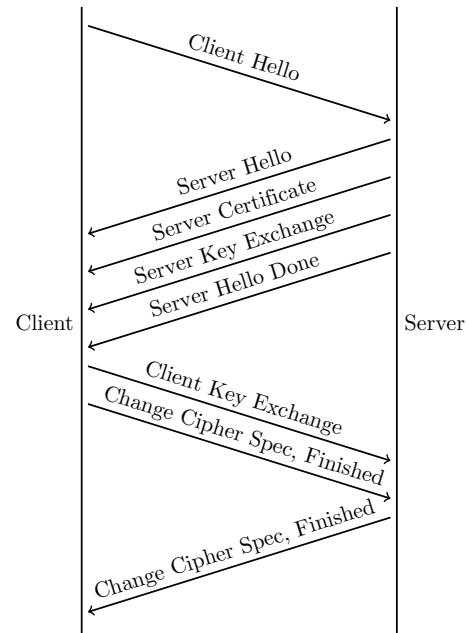


Figure 1. TLS 1.2 full handshake.

Once the client receives the ServerHello message, it first verifies the server's certificate and caches the information needed for a reconnection. Optionally, the client may need to send its certificate if required by the server. Similarly, a Client Key Exchange message encrypted by the server's public key is also sent by the client. Now the client can infer the master secret based on the Key Exchange messages and the random number R_c and R_s . A Change Cipher Spec message will be sent to notify the server that any data sent by the client from now on will be encrypted by the symmetric key derived from the master secret, followed by a Finished message, which contains a hash of all the handshake messages sent previously. Once the server uses its private key to decrypt the Client Key Exchange message, the server can also calculate the same master secrets similar to the client. Now the server can use the symmetric key also derived from the master secret to decrypt the client handshake Finished message and check the validity of all previous handshake messages. After this, the server will also send the Change Cipher Spec and Finished message similar to the client to achieve the same goal. Once the client successfully verifies the server's Finished message, a full TLS Handshake is completed, and application data can be transmitted securely. Therefore, TLS 1.2 takes 2-RTT to create a new secure connection.

2.2. Quantum Key Distribution

BB84 [8] is a quantum key distribution scheme developed by Charles Bennett and Gilles Brassard in 1984. The protocol is provably secure [17], relying on the fundamental aspect of quantum mechanics that information gain is only possible at the expense of disturbing the signal [18,19]. The workflow of BB84 is shown in Figure 2.

1. Alice randomly generates two strings, $a = a_0a_1 \dots a_n$ and $b = b_1b_2 \dots b_n$. a is called the basis string. If $a_i = 0$, she chooses a qubit in $\{|0\rangle, |1\rangle\}$ according to b_i ; otherwise, she chooses a qubit in $\{|+\rangle, |-\rangle\}$ ($|+\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and $|-\rangle = \frac{|0\rangle-|1\rangle}{\sqrt{2}}$) according to b_i . These qubits are sent by Alice to Bob in strict sequence.

2. After receiving these quantum states, Bob randomly measures each qubit in basis $\{|0\rangle, |1\rangle\}$ or in basis $\{|+\rangle, |-\rangle\}$, and records his choice of basis string as a' . After he receives the measurement outcome b' , a' is sent to Alice.
3. Alice discards the bits of b in the locations where a and a' do not match. Alice announces a first; then she randomly takes half of the rest of the bits of b to be the checking bits b_c and announces the selection.
4. Bob discards the bits of b' in the locations where a and a' do not match after knowing a . Then he checks if the states of qubits in checking bits b_c before and after the measurement are the same. If the disagreement is less than an acceptable threshold, the quantum channel is considered to be reliable, and the remaining bits can be used to obtain the shared key bits.

However, the standard BB84 protocol [8] is not practical since it is not feasible to generate strings of truly perfect single photons efficiently [20]. The decoy-state method [14,21–23] was proposed to use a few different photon intensities to perform almost as well as a single photon. Different configurations [24,25] of decoy states and the length of keys [26] would also affect the secret key rates. An optimal configuration would reduce the time for gathering sufficient statistics to generate a secure key, which is important to make classical and quantum channels collaborate well on the same time scale. Moreover, the high-speed state of polarization generation [27,28] and other degrees of freedom of polarization, such as time-bin [15], can also be implemented to improve the efficiency of QKD.

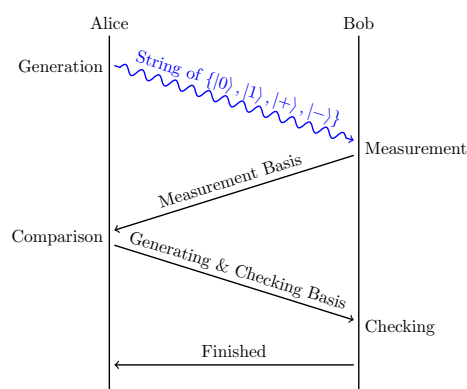


Figure 2. BB84 workflow. A string of polarized photons randomly in $\{|0\rangle, |1\rangle, |+\rangle, |-\rangle\}$ states are sent by Alice to let Bob give random measurements in sequence. After the measurements on the photons and basis communication, both sides can generate a shared secret.

3. QQUIC Transport Protocol

In the previous section, we briefly explained how BB84 works to generate a shared secret between two sides. If we want to combine these properties in traditional transport protocols, it requires a quantum channel to generate, transport, and measure the single polarized photon in a predefined way, and it also needs two peers to communicate their basis choice and measurement results through the classical channel. This may appear to be much more complicated compared with the traditional key exchange process, but the quantum channel and classical channel can work in parallel to finish the handshake, and QKD can also continuously work to provide keys for key updates after the connection has been established.

We introduce a new secure transport protocol called QQUIC, where QKD replaces the key exchange stage in QUIC. The encryption and transmission of the application data after connection establishment works the same way as QUIC, except for the initial full handshake for a new connection and key updating. There are two channels during the connection; one is the classical channel used in traditional transport protocols, and the other quantum channel is used for quantum operations on the photons. The network structure of the protocol works as shown in Figure 3.

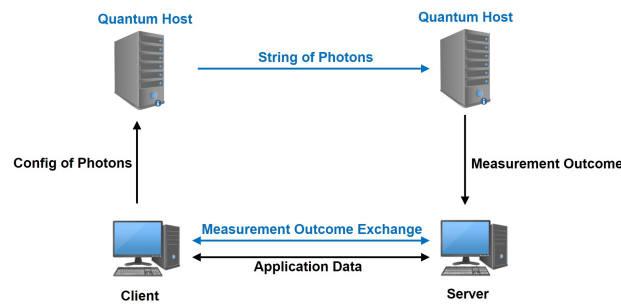


Figure 3. Network Structure of the QQUIC.

Each side has a quantum host and a classical host. The quantum hosts and their authenticated quantum channel are responsible for generating the shared secret to replace the classical key exchange. For the initial full handshake and the periodical key updating, the client needs to activate its quantum host to generate and send strings of special polarized photons, where the quantum host undertakes the generation of true randoms a and b , which determines the photon states. Meanwhile, the client informs the server of the configuration of the photons through the classical channel, including the state basis for photon generation and measurement, the number and sending speed of photons, etc. After the server knows the configuration of these photons, its quantum host generates a random basis string a' and uses it to get the measurement outcome b' of the received photons. After the measurement, the two sides communicate their basis choices a and a' and discard the useless bits in b and b' . After verifying the validity of the quantum channel, the secure, shared secret can be created and used for key derivation. Notice that the client cannot announce its basis string a' until it knows the server has completed the measurements of the photons, which means the QKD needs at least 2-RTT to complete the shared secret negotiation. Here we give a detailed description of the workflow of the full handshake, shown in Figure 4.

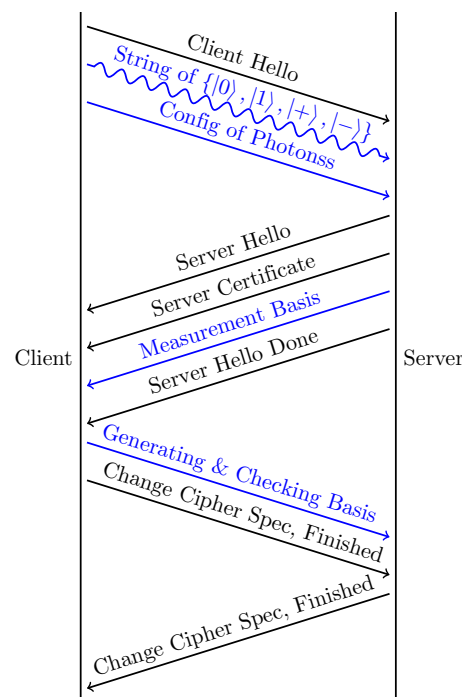


Figure 4. QQUIC full handshake.

Initially, the client has no information about the server, so it needs to initiate the connection via a ClientHello (CHLO) message. The CHLO message contains the lists of supported TLS versions, cipher suites, signature algorithms, and extensions for session resumption, which are similar to the CHLO message in TLS 1.3. However, the CHLO

package provides no information related to key exchange, such as a random client number or key exchange extension. Instead, the client requires its quantum host to generate two private binary randoms, a and b , to determine how to generate a string of polarized photons and send them through the quantum channel at the same time, as explained in BB84. Thus, the CHLO message usually needs to contain some extra information about the photons, such as the set of state basis used for photon generation and measurement and the number and sending speed of photons. The transmission of photons in the quantum channel works in parallel with the CHLO message in the classical channel.

The server gives a ServerHello (SHLO) message as a response to the CHLO package. The SHLO message contains: (a) the server configuration information, (b) the choice of cipher suites and signature algorithms, and (c) a certificate and certificate verification. The server's quantum host needs to receive these polarized photons in the same order and measure them in sequence by a basis string a' , which denotes the random selections of two possible bases provided by the client. After this, the server keeps the measurement results b' private and sends the string a' to the client through the classical channel.

Once the client receives the SHLO message from the server, it can verify the identity of the server via the certificate and signature. After this, the client has to wait for the handshake message about the server's choice of measurement basis a' and discard the bits in b where the two sides choose differently. Then the client needs to randomly take half of the remaining bits in b to create a random b_c , and the rest of the bits in b can be used to generate the shared secret. Now the client needs to send a (generating basis) to inform the server as to how these photons are created, and b_c (checking basis) to let the server decide which photons are used for checking. Finally, the client should use the derived key to encrypt the Change Cipher Spec message and the Finished message, which contains a hash of all the handshake information up to this point to avoid tampering.

Once receiving the handshake messages from the client, the server should first check whether or not the quantum measurement on the photons corresponding to the bits in b_c have changed their states, where the states before measurement can be inferred from a and b_c . If the difference rate is above a safe threshold, this means there is too much noise or there is an eavesdropper in the quantum channel. Thus, the handshake has failed, and it is necessary to try again. Otherwise, the server can safely derive the same shared secret based on a and b_c from the client and its measurement results b' . Now the server can decode the Finished message from the client to check the validity of all the previous messages. In the end, the server also sends the Change Cipher Spec and Finished message to the client. After the client checks this Finished message, a safe connection has been established.

As a comparison, the establishment of a safe connection via QKD requires 2-RTT, which is 1-RTT slower than TLS 1.3. However, if the success rate of the validity checking for the quantum channel is high enough, the client can also send some unimportant encrypted application data after the client's Finished message in the first round trip, and the server can also directly give its response to the request after its Finished message. This "quick start" strategy can be used in most HTTP requests, where it usually begins with an unimportant request, such as permission for log-in. As for the session assumption, the PSK mode and 0-RTT reconnection used in QUIC can also be deployed here. Again, because the quantum channel needs 2-RTT to generate the shared secret safely, the application data in the first two round trips during the handshake are still encrypted by the previous old key.

It has been confirmed that a connection encrypted by one session key for a long time is not safe enough, even using authenticated encryption with associated data. To guarantee security, TLS 1.3 proposes to use the KeyUpdate message to update session keys in an existing connection, which takes 1-RTT to notify each side and derive the next shared traffic secret based on the current traffic secret and KeyUpdate message, as shown in Figure 5a.

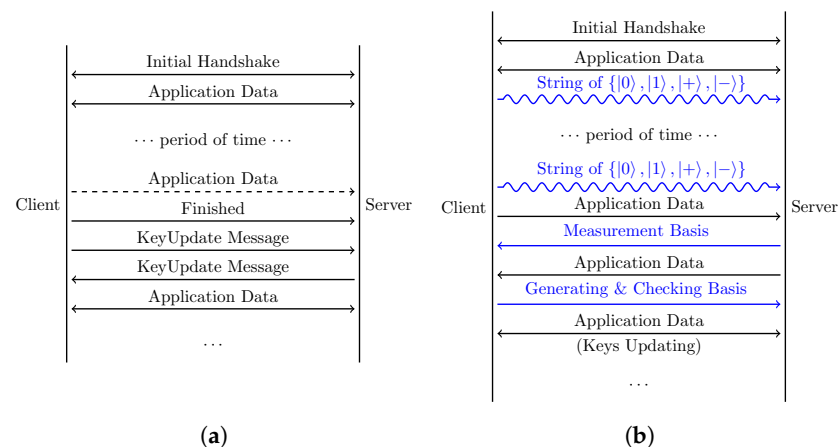


Figure 5. KeyUpdate in a long-term connection. (a) KeyUpdate in TLS 1.3; (b) KeyUpdate in QQIC.

In addition to the higher security level of the master secret derived from QKD, the most obvious advantage of deploying QKD is that the key generation and application data transmission can work in parallel. As shown in Figure 5b, after the connection has been established, the client can keep sending strings of photons continually, and each string can generate a master secret. The server measures a group of photon strings once and negotiates the generation and measurement bases in only one round trip, which can effectively save travel time. Since every photon string is independent of each other, we can guarantee the independence between these shared secrets. This ensures that each side always has several shared secrets stored in advance for a future key update. If both sides have made an agreement about the number of ordered packages for the lifetime of one session key, then the two sides can automatically switch to a new session key in the sequence without notification when the right time comes. The package number and the stream offset in QUIC are used to reorder the received packages and choose the right session key for different package chunks. This protocol will have huge advantages for long-term connections such as video conferences and big data transmission, which require frequent key updates. Furthermore, due to the higher security of QKD, the results from QKD can be used as a master secret directly, and there is no need to add more random materials to generate the final session keys compared with the HKDF used in TLS 1.3. The same goes for the frequent key updates in the long-term connection, where the session's key derivations of different chunks are totally independent.

4. Discussion and Conclusions

In this paper, we propose a new protocol named QQIC, which combines the advantages of QUIC and quantum key distribution to achieve more secure and effective key generation for the initial handshake and key updating. This protocol uses QKD to replace the classical key exchange procedure and performs a 2-RTT full handshake for a new connection, and with a 1-RTT time cost with the “quick start” technique. The concurrent work of the quantum and classical channels makes the key generation much more efficient and resource-saving, which has huge advantages in long-term connection applications, such as video meetings and database sharing. For implementation of QKD with a relatively high key rate, the portion of asymmetric cryptography in the network protocol may be significantly reduced. This would provide a considerable computational speedup due to the well-known fact that public-key cryptography, or asymmetric cryptography, is much slower than symmetric cryptography. Today QKD is a relatively mature technology, and some companies such as ID Quantique, QuantumCTek, and ThinkQuantum can provide feasible turn-key solutions for implementing and testing our proposal.

Author Contributions: Conceptualization, N.Y.; Formal analysis, P.Y. and N.Y.; Funding acquisition, N.Y.; Investigation, P.Y. and N.Y.; Supervision, N.Y.; Validation, P.Y. and N.Y.; Visualization, P.Y.;

Writing—original draft, P.Y.; Writing—review and editing, N.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the ARC Discovery Program (#DP210102449) and ARC DECRA (#DE180100156).

Institutional Review Board Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: N.Y. would like to thank Gushu Li for the helpful discussions at the beginning of this project. N.Y. is grateful to Zhengfeng Ji for sharing his insight on this topic.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Diffie, W.; Hellman, M. New Directions in Cryptography. *IEEE Trans. Inf. Theor.* **2006**, *22*, 644–654. [[CrossRef](#)]
2. Langley, A.; Riddoch, A.; Wilk, A.; Vicente, A.; Krasic, C.; Zhang, D.; Yang, F.; Kouranov, F.; Swett, I.; Iyengar, J.; et al. The QUIC Transport Protocol: Design and Internet-Scale Deployment. In Proceedings of the Conference of the ACM Special Interest Group on Data Communication, Association for Computing Machinery, New York, NY, USA, 21–25 August 2017; pp. 183–196. [[CrossRef](#)]
3. Shor, P.W. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM J. Comput.* **1997**, *26*, 1484–1509. [[CrossRef](#)]
4. Ladd, T.; Jelezko, F.; Laflamme, R.; Nakamura, Y.; Monroe, C.; O’Brien, J. Quantum computers. *Nature* **2010**, *464*, 45–53. [[CrossRef](#)] [[PubMed](#)]
5. Flamini, F.; Spagnolo, N.; Sciarrino, F. Photonic quantum information processing: A review. *Rep. Prog. Phys.* **2018**, *82*, 016001. [[CrossRef](#)]
6. Vandersypen, L.M.K.; Steffen, M.; Breyta, G.; Yannoni, C.S.; Sherwood, M.H.; Chuang, I.L. Experimental realization of Shor’s quantum factoring algorithm using nuclear magnetic resonance. *Nature* **2001**, *414*, 883–887. [[CrossRef](#)]
7. Politi, A.; Matthews, J.C.F.; O’Brien, J.L. Shor’s Quantum Factoring Algorithm on a Photonic Chip. *Science* **2009**, *325*, 1221. [[CrossRef](#)]
8. Bennett, C.; Brassard, G. Quantum cryptography: Public key distribution and coin tossing. In Proceedings of the IEEE International Conference on Computers, Systems, and Signal Processing, Bangalore, India, 9–12 December 1984; p. 175.
9. Bennett, C.H.; Bessette, F.; Brassard, G.; Salvail, L.; Smolin, J. Experimental Quantum Cryptography. *J. Cryptol.* **1992**, *5*, 3–28. [[CrossRef](#)]
10. Elliott, C.; Pearson, D.; Troxel, G. Quantum Cryptography in Practice. In Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, New York, NY, USA, 25–29 August 2003; pp. 227–238.
11. Peev, M.; Pacher, C.; Alléaume, R.; Barreiro, C.; Bouda, J.; Boxleitner, W.; Debuisschert, T.; Diamanti, E.; Dianati, M.; Dynes, J.F.; et al. The SECOQC quantum key distribution network in Vienna. *New J. Phys.* **2009**, *11*, 075001. [[CrossRef](#)]
12. Yin, J.; Cao, Y.; Li, Y.H.; Liao, S.K.; Zhang, L.; Ren, J.G.; Cai, W.Q.; Liu, W.Y.; Li, B.; Dai, H.; et al. Satellite-based Entanglement Distribution over 1200 Kilometers. *Science* **2017**, *356*, 1140–1144. [[CrossRef](#)]
13. Pirandola, S.; Andersen, U.L.; Banchi, L.; Berta, M.; Bunandar, D.; Colbeck, R.; Englund, D.; Gehring, T.; Lupo, C.; Ottaviani, C.; et al. Advances in quantum cryptography. *Adv. Opt. Photon.* **2020**, *12*, 1012. [[CrossRef](#)]
14. Boaron, A.; Boso, G.; Rusca, D.; Vulliez, C.; Autebert, C.; Caloz, M.; Perrenoud, M.; Gras, G.; Bussièeres, F.; Li, M.J.; et al. Secure Quantum Key Distribution over 421 km of Optical Fiber. *Phys. Rev. Lett.* **2018**, *121*, 190502. [[CrossRef](#)]
15. Islam, N.T.; Lim, C.C.W.; Cahall, C.; Kim, J.; Gauthier, D.J. Provably secure and high-rate quantum key distribution with time-bin qudits. *Sci. Adv.* **2017**, *3*, e1701491. [[CrossRef](#)]
16. Agnesi, C.; Avesani, M.; Calderaro, L.; Stanco, A.; Foletto, G.; Zahidy, M.; Scriminich, A.; Vedovato, F.; Vallone, G.; Villoresi, P. Simple quantum key distribution with qubit-based synchronization and a self-compensating polarization encoder. *Optica* **2020**, *7*, 284. [[CrossRef](#)]
17. Shor, P.W.; Preskill, J. Simple Proof of Security of the BB84 Quantum Key Distribution Protocol. *Phys. Rev. Lett.* **2000**, *85*, 441–444. [[CrossRef](#)]
18. Dieks, D. Communication by EPR devices. *Phys. Lett.* **1982**, *92*, 271–272. [[CrossRef](#)]
19. Wootters, W.K.; Zurek, W.H. A Single Quantum Cannot be Cloned. *Nature* **1982**, *299*, 802–803. [[CrossRef](#)]
20. Scarani, V.; Bechmann-Pasquinucci, H.; Cerf, N.J.; Dušek, M.; Lütkenhaus, N.; Peev, M. The security of practical quantum key distribution. *Rev. Mod. Phys.* **2009**, *81*, 1301–1350. [[CrossRef](#)]
21. Hwang, W.Y. Quantum Key Distribution with High Loss: Toward Global Secure Communication. *Phys. Rev. Lett.* **2003**, *91*, 057901. [[CrossRef](#)]
22. Rosenberg, D.; Harrington, J.W.; Rice, P.R.; Hiskett, P.A.; Peterson, C.G.; Hughes, R.J.; Lita, A.E.; Nam, S.W.; Nordholt, J.E. Long-Distance Decoy-State Quantum Key Distribution in Optical Fiber. *Phys. Rev. Lett.* **2007**, *98*, 010503. [[CrossRef](#)]

23. Foletto, G.; Picciariello, F.; Agnesi, C.; Villaresi, P.; Vallone, G. Security bounds for decoy-state quantum key distribution with arbitrary photon-number statistics. *Phys. Rev. A* **2022**, *105*, 012603. [[CrossRef](#)]
24. Wang, X.B. Beating the Photon-Number-Splitting Attack in Practical Quantum Cryptography. *Phys. Rev. Lett.* **2005**, *94*, 230503. [[CrossRef](#)] [[PubMed](#)]
25. Ma, X.; Qi, B.; Zhao, Y.; Lo, H.K. Practical decoy state for quantum key distribution. *Phys. Rev. A* **2005**, *72*, 012326. [[CrossRef](#)]
26. Rusca, D.; Boaron, A.; Grunenfelder, F.; Martin, A.; Zbinden, H. Finite-key analysis for the 1-decoy state QKD protocol. *Appl. Phys. Lett.* **2018**, *112*, 171104. [[CrossRef](#)]
27. Grunenfelder, F.; Boaron, A.; Rusca, D.; Martin, A.; Zbinden, H. Performance and security of 5 GHz repetition rate polarization-based quantum key distribution. *Appl. Phys. Lett.* **2020**, *117*, 144003. [[CrossRef](#)]
28. Ramos, M.; Pinto, A.; Silva, N. Polarization based discrete variables quantum key distribution via conjugated homodyne detection. *Sci. Rep.* **2022**, *12*, 6135. [[CrossRef](#)] [[PubMed](#)]