A high-speed key management method for quantum key distribution network

Ririka Takahashi, Yoshimichi Tanizawa, Alexander Dixon
Corporate Research and Development Center, Toshiba Corporation, Kawasaki, Japan
Email: {ririka.takahashi, yoshimichi.tanizawa, alexander.dixon}@toshiba.co.jp

***Abstract*—Quantum Key Distribution (QKD) is a technique for sharing encryption keys between two adjacent nodes. It provides unconditional secure communication based on the laws of physics. From the viewpoint of network research, QKD is considered to be a component for providing secure communication in network systems. A QKD network enables each node to exchange encryption keys with arbitrary nodes. However previous research did not focus on the processing speed of the key management method essential for a QKD network. This paper focuses on the key management method assuming a high-speed QKD system for which we clarify the design, propose a high-speed method, and evaluate the throughput. The proposed method consists of four modules: (1) local key manager handling the keys generated by QKD, (2) one-time pad tunnel manager establishing the transparent encryption link, (3) global key manager generating the keys for application communication, and (4) web API providing keys to the application. The proposed method was implemented in software and evaluated by emulating QKD key generation and application key consumption. The evaluation result reveals that it is capable of handling the encryption keys at a speed of 414 Mb/s, 185 Mb/s, 85 Mb/s and 971 Mb/s, for local key manager, one-time pad tunnel manager, global key manager and web API, respectively. These are sufficient for integration with a high-speed QKD system. Furthermore, the method allows the high-speed QKD system consisting of two nodes to expand corresponding to the size of the QKD network without losing the speed advantage.**

Keywords—QKD network, quantum key distribution, key management, secure communication, one-time pad

I. INTRODUCTION

In view of the explosive growth in the volume of communication data in recent years, the security of communication is becoming increasingly important. Users require communication data to be safe from interception and eavesdropping.

Quantum Key Distribution (QKD) is a technique for sharing random numbers between two adjacent nodes. Since QKD can detect any eavesdropping based on quantum theory [1], users can communicate in an unconditionally secure manner while using the shared random numbers as encryption keys. As QKD provides security against any eavesdropping attacks even with infinite computing resources such as quantum computing, QKD is considered a highly secure communication technology.

A basic QKD system consists of a set of a transmitter and a receiver, connected via an optical fiber. The encryption key sharing process consists of preparing random numbers, a series of single photons transmitted/detected based on the random numbers, sifting for the transmitted/detected photon data by exchanging control data, and key distillation processing [1]. As a result of the process, random numbers are shared between the transmitter and the receiver, which can be used as shared encryption keys. When the shared encryption keys are used with a one-time pad (OTP) algorithm [2], which requires the use of a one-time encryption key the same size as the communication message, the encrypted data communications are unconditionally secure [3]. Through several field trials [4-8], QKD technology has moved from the research phase to the engineering phase and is considered to be a future secure communication solution for fields such as defense, finance and healthcare

However, QKD still has two limitations that need to be overcome. The first is the limited spread of key sharing. Since the keys are shared between the transmitter and the adjacent receiver, only these two directly connected nodes can communicate the encrypted data. This in turn means that non-adjacent distant nodes cannot communicate the encrypted data, and this limits the data communication scalability. To overcome the distance limitation, several QKD field trials have been conducted for networking QKD technologies, namely, "QKD networks". The world's first field trial is the DARPA Quantum Network [4]. The DARPA Quantum Network consists of five sets of QKD systems and relays keys between the nodes to achieve longer-distance key sharing. Following the DARPA Quantum Network, the best-known field trial is SECOQC [5]. The SECOQC field trial uses a dynamic routing protocol for determining the next hop node for key relaying on the QKD network. The Tokyo QKD network [6] is another notable field trial. This field trial defines the key management layer and demonstrates encrypted video broadcasting as an application using the encryption key shared by QKD. Other QKD network field trials have also been conducted in Geneva [7], and in China [8]. As regards the research on QKD networks, a wide range of topics have been addressed, including the efficient QKD routing method [9], QKD network simulation using ns-3 [10] and network traffic analysis in QKD networks [11]. In regard to the research on the key management for QKD networks, there are some implementations and evaluations [12-14]. Another approach for networking QKD is using quantum repeater networks [15]. The quantum repeater network transfers photon state using quantum teleportation and entanglement swapping instead of relaying key data on the node. However, quantum repeater technology is still ongoing research, while QKD networks are in the engineering phase.

The second limitation is key sharing speed [16, 17]. Due to the difficulty of high-speed and high-accuracy photon transmission and detection, and fluctuations depending on the environmental changes such as the optical fiber vibration and the temperature change, the record for key sharing speed in a real fiber environment was about 1.9 Mb/s [18]. In addition, although in a laboratory environment, the world's highest speed of about 13.72 Mb/s [19] was recorded by developing high-speed key distillation processing. In regards to data transmission, 200 G/s [18] classical data transmission while riding the same single fiber with photon transmission for QKD was reported, though the data traffic was encrypted using the AES algorithm. To exploit the QKD advantage of unconditional secure communication, the data traffic needs to be encrypted by an OTP algorithm. The key sharing speed required for OTP is identical to the data transmission speed since OTP encrypts the data one bit at a time with the use of a one-time encryption key. For the case of AES, the key sharing speed depends on the key refresh frequency since AES can use the encryption key repeatedly. Thus, the OTP processing speed is important for the design of a high-speed QKD system and secure communications.

However, the previous research did not focus on the speed of key sharing in a QKD network and did not detail the design, implementation and evaluation for the key management method essential for a QKD network. This paper proposes a high-speed key management method capable of handling high-speed QKD systems in a QKD network. The result leads to straightforward experiments and commercialization of the high-speed QKD network. The proposed method consists of the four modules: (1) local key manager, (2) OTP tunnel manager, (3) global key manager, and (4) key providing web Application Programming Interface (API). The local key manager ensures efficient use of the encryption keys generated by the QKD. Keys are simply sorted to storage divided for encryption and for decryption. The OTP tunnel manager provides the virtual interface that realizes the OTP encryption tunnel. An unconditional secure communication channel is generated in an efficient manner by changing the data transfer configuration and changing a key removing method. The global key manager prepares the random number to be provided to an application as an encryption key. The key providing web API is an interface applied by a lightweight architecture that provides the encryption key to an application. The proposed key management method can manage, forward, and provide the encryption key safely and at high speed even if there are several encryption keys shared with each node in the QKD network.

The proposed method was implemented and evaluated with a network emulating QKD and an application. The results will show the throughputs of 414 Mb/s, 185 Mb/s, 85 Mb/s and 971 Mb/s for the local key manager, OTP tunnel manager, global key manager and key providing web API, respectively. Furthermore, the proposed method can extend the high-speed QKD system corresponding to the size of the QKD network.

The contributions of this paper are presenting the design, the implementation, and the evaluation results of the high-speed key management method and clarifying the speed capability of the proposed key management method and showing it is sufficient for a high-speed QKD system.

The rest of the paper is organized as follows. Section II describes the basic and standard QKD network architecture. Section III describes the proposed key management method. Section IV explains the evaluation of the result. Section V concludes the paper.

II. QKD Network

This section describes a basic QKD network architecture. The QKD network described is based on the SECOQC project [5]. QKD only ensures the communication security between nodes. Further security should be ensured by operational-level or physical-level security technology. The details and the security assumptions are also described in [20]. Fig. 1 shows the structure of the QKD network. The QKD network consists of three types of computer, namely, QKD nodes, key management nodes and applications. In addition, there are three types of network, namely, QKD links, key sharing network and data communication network. Communications between the application and the key management node and between the key management node and the QKD node are assumed to be secured by conventional security such as public key cryptography. They should authenticate one another by means such as password, pre-shared key or certification using SSH. As an alternative approach, all three types of computers can be placed physically at the same secure area, such as inside a securely locked single server rack or server room with rigorous access control.

A. QKD Node

A QKD node is a QKD transmitter/receiver sharing the random numbers as encryption keys by QKD. We call the encryption keys shared between the directly connected QKD nodes by QKD "local keys." All QKD nodes are assumed to be trusted and not to maliciously use the local keys. Note that the security of the transmitted data encrypted/decrypted by local keys is valid only between directly connected QKD nodes.

B. QKD Link

A QKD link is an optical fiber link connecting QKD transmitters and QKD receivers for QKD. Since the security of the QKD relies on the basic principle of quantum mechanics, the photons cannot be amplified and split on the optical fiber link. Furthermore, the one-to-one QKD node can be connected by only one QKD link.

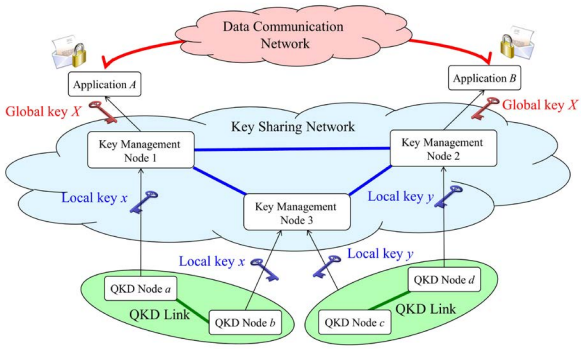


Fig. 1. QKD network scheme.

C. Key Management Node

A key management node is a computer equipped with one or more QKD transmitters/receivers, and manages the local keys shared between all adjacent nodes. The key management node receives the local key generated by each QKD node and generates the tunnel link encrypted by the OTP algorithm based on the local keys shared with adjacent QKD nodes. The key management node also has a function for generating other random numbers that are different from the local keys. We call the random numbers used for secure communication between applications "global keys." All key management nodes including the relaying nodes are assumed to be trusted and not to compromise the global keys. Moreover, the key management node should be placed in a secure location, such as a room having the entrance and exit controlled by biometric authentication. Secret sharing schemes for relaying global keys could partially relax the requirements, allowing security even with some untrusted key management nodes in a QKD network [21].

D. Key Sharing Network

A key sharing network consists of multiple connected key management nodes as well as the OTP tunnel link generated by using the local key. The global keys are forwarded to and shared with the other key management nodes via the OTP tunnel link. A key management node encrypts the global keys using the local key shared with the connected key management nodes, and transmits the encrypted global key to the connected key management nodes. The key management nodes receiving an encrypted global key decrypt it by using the local key shared with the originating key management node, determine the next hop, re-encrypt the global key using the local key shared with the next hop key management node, and transmit the encrypted global key to the determined next hop key management node. This key relaying sequence allows key management nodes to share encryption keys with an arbitrary key management node on the key sharing network. Since global keys are encrypted by the OTP algorithm using unconditionally secure local keys, global keys are also unconditionally secure.

E. Application

The application is formed by computers securely communicating with each other. Each application obtains the global keys from the connecting key management node, and performs secure communications with the other applications using the global keys.

F. Data communication Network

Secure communication between applications is performed via the data communication network. The applications can use any encryption algorithm using global keys. When the applications use the OTP algorithm for encryption and for decryption, data communication is also unconditionally secure.

III. Key Management Method

This section describes the functions and the design of the four modules that make up the key management method as shown in Fig. 2.

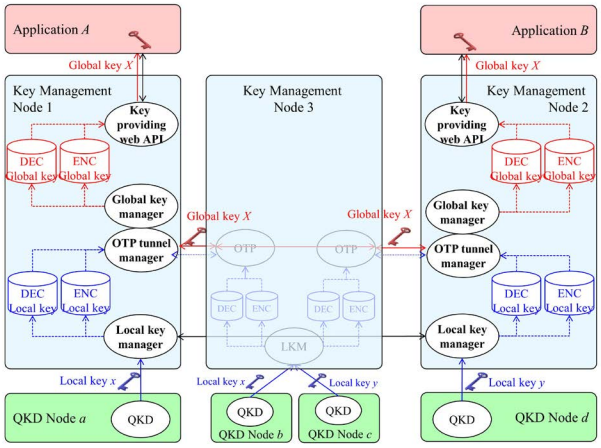


Fig. 2. Key management design. QKD nodes generate local keys and transfer them to key management nodes. Local key managers in key management nodes sort the keys into storage for encryption (ENC) and for decryption (DEC). OTP tunnel managers generate the OTP encryption tunnel by using the local keys. Global key managers generate global keys, share them via OTP tunnel and save them into database for encryption (ENC) and for decryption (DEC). Key providing web API provides the global keys to applications.

A. Local Key Manager

The local key manager reads the local keys by SSH protocol from QKD node to secure access and sorts them for encryption and for decryption based on a predetermined ratio while monitoring the upper limit of the number of stored keys. The functions required for the local key manager are as follows:

1) Key sorting function: This function is to sort the local keys separately into for encryption and for decryption based on a predetermined ratio. The OTP tunnel manager selects between encrypting and decrypting according to the data transmission direction. In the case of bidirectional communication, it is necessary to separate the storage for encryption and for decryption. The local key saving ratio between encryption and decryption should be configured unsymmetrically, so that the number of local keys can be used efficiently if the traffic in one direction is heavier than in the other direction.

2) Upper limit monitoring function: This function is to monitor the predetermined upper limit of local key storage capacity. The local key in processing is discarded if the number of stored keys approaches the maximum capacity.

B. One-Time Pad Tunnel Manager

The OTP tunnel manager provides a virtual interface and an encryption tunnel. By using this interface, the data communication is encrypted by the OTP algorithm transparently. The OTP tunnel manager encrypts/decrypts the data by consuming the local key according to the data transmission direction. The global keys are transmitted to the arbitrary node via the key sharing network consisting of the OTP tunnel in order to prevent leakage of global key information to eavesdroppers. The functions required for OTP tunnel manager are as follows:

1) Virtual interface generating function: This function provides a communication interface based on the virtual device. A TUN device [22] that can operate IP packets and simulate the network layer in the OSI model is used. The MTU (Maximum Transmission Unit) size, which is one of the parameters, is configured to the default 1500 bytes to support the Ethernet link. However the virtual interface can limit the throughput, since the OTP module needs to handle all of the small packets arising from IP fragmentation. To overcome this limitation, the MTU size is extended from the 1500 bytes as a TUN device can operate IP packets. The impact of the MTU size extension is shown in Fig. 3. If the MTU size is extended from 1500 bytes to 65535 bytes, the number of OTP module calls is reduced from 700 to 17, when the transmitted data are assumed to be 1 Mbytes. Thus, MTU extension leads to the overhead reduction for the packet operation of the OTP tunnel.

2) Encrypting/decrypting function: The function is used to OTP encrypt/decrypt by using the local key according to the data transmission direction. This function can prevent fast implementation since it imposes overheads such as OTP encryption, decryption, local key read and used key removal. To avoid the overhead, the used key removing is improved as shown in Fig. 4. In the conventional method, the used keys are removed by every packet encryption/decryption as shown in (a). However, in the proposed method, they are removed in a larger block unit, when the number of used keys exceeds the predetermined large unit as shown in (b). This leads to throughput improvement by decreasing the overheads attributable to the used key removing.

3) Key routing function: The function is to forward the data to the final destination key management node by OTP encrypting and decrypting. At the relay node, the data is forwarded by using an IP routing function provided by the OS kernel, when the source node and the destination node are not connected directly by QKD link.

The design flow of the OTP tunnel manager is summarized as follows: (1) receive the data from the virtual interface/socket, (2) read the encryption/decryption keys and remove used keys, (3) OTP encryption/decryption using the local key, and (4) send the encrypted/decrypted data to the socket/virtual interface. The network architecture forwarding the global key is constituted by extending the one-to-one directly connected OTP tunnel.

C. Global Key Manager

The global key manager shares the global keys that are stored separately for encryption and for decryption used for the communication between arbitrary nodes. The global key manager generates a random number to be the global key, saves it to its own encryption key database (DB) and sends it to the key management node connecting to the destination application. The destination key management node saves the received global key into the decryption key DB. The security of the global key sharing can be ensured by forwarding the global key via the key sharing network consisting of an OTP tunnel. The functions required for the global key manager are as follows:

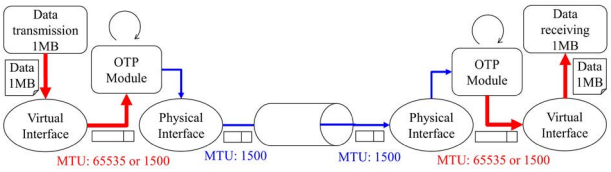
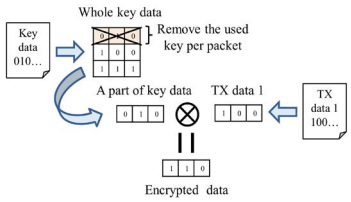
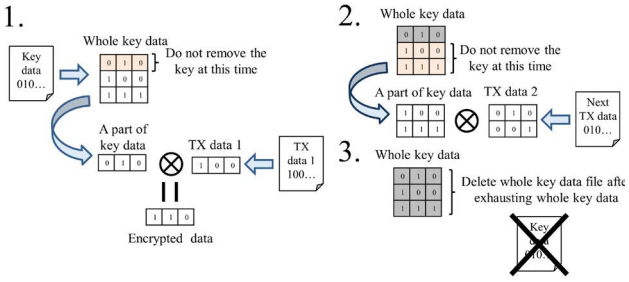


Fig. 3. Impact of the MTU size extension on the number of OTP module calls. If the MTU size is 65535 bytes as the proposed method, 1 Mbytes transmitted data are divided into 1048576/65535=17 packets. OTP module is called 17 times. Whereas if the MTU size is 1500 bytes as the conventional method, the data are divided into 1048576/1500=700 packets. OTP module is called 700 times for the 1 Mbytes data transmission.



(a) Remove every packets



(b) Remove in a larger unit

Fig. 4. Key removing functions. (a) Used key is removed every packets as the conventional method (left). (b) Used key is removed in a larger unit as the proposed method. The method removes used keys when the number of used keys exceeds to the predetermined units (right).

1) Random number generating function: The function is to generate a random number to be a global key by using pseudo device /dev/urandom etc.

2) Key sending function: The function is to send the global key to an arbitrary node.

3) Key saving to DB function: The function is to save the shared global key into a DB such as MongoDB.

4) Key restocking function: The function is to restock the global keys if the number of stored global keys in the DB is less than the predetermined threshold.

In order to provide the global key faster, the global key manager generates and shares the global keys before the application request.

D. Key Providing Web API

The key providing web API delivers the global key taken from the encryption DB and decryption DB according to the application request. The predetermined length of global keys is provided. The key providing web API employs the REST API

style [23] for simple and fast implementation. The API uses the HTTPS protocol for secure access between the application and the key management node. The API functions required for the key providing web API are as follows:

1) Key status providing API: The API provides the global key information based on the result of the global key manager.

2) Encryption key providing API: The API provides encryption global keys and their identifiers, which are shared with the specified key management nodes, by taking them from the encryption DB.

3) Decryption key providing API: The API provides decryption global keys and their identifiers, which are shared with the specified key management nodes, by taking them from the decryption DB. The application specifies the key identifier, which is obtained at the time when the application has requested the encryption global key. The API returns the decryption global key according to the requested key identifier.

IV. EVALUATION

The evaluated results as well as considerations about the proposed key management method are presented in this section.

A. Evaluation Environment

The proposed key management method was evaluated with a QKD node emulator that can generate the local keys at a specified speed to emulate the QKD nodes, and with an application emulator that can request the global keys to the API to emulate a real application. Three PCs, namely, PC 1, PC 2 and PC 3 are connected with each other directly by a 1 Gb Ethernet cable. The specification of each PC is HP Compaq Elite 8300 (CPU: Core i5-347 3.2 GHz, RAM: 4GB, HDD: 500GB, OS: CentOS 7 64bit Kernel 3.10.0). Three software elements – the QKD node emulator, the proposed key management modules and the application emulator – are all run on each PC.

B. OTP Tunnel Throughput

The throughput of the virtual interface of the OTP tunnel between PC 1 and PC 2 is evaluated. The OTP tunnel manager uses the local keys that have been stored in advance by the local key manager. The conventional method uses an MTU with a default size of 1500 bytes and removes the used key for every packet. The proposed method uses an MTU with a size of 65535 bytes and removes used keys by larger block unit.

The OTP throughputs as a function of virtual interface MTU size are shown in Fig. 5. The conventional method obtains 3.92 Mb/s for the throughput. However this throughput does not satisfy the requirement for a high-speed QKD system. The proposed method obtains an OTP throughput of 185 Mb/s. This is sufficiently fast for a high-speed QKD system. It also can be seen that the OTP throughputs increase as the MTU size increases. This means the OTP overhead decreases as the MTU size increases. The average processing times for each step in the OTP tunnel manager for the proposed method and for the conventional method are shown in Table I. A step for local key reading and removing in the proposed method improves from 208 ms to 7.3 ms. It is reduced to one-twenty-eighth by modifying the key removing unit and MTU size from the conventional method. Moreover the processing times for all steps are reduced when the MTU size is increased. This means that the OTP overhead is reduced by changing the MTU size.

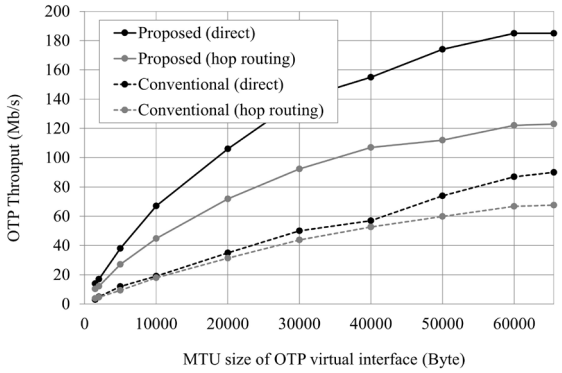


Fig. 5. OTP throughput as a function of MTU size. The solid line presents the proposed method and the dashed line presents the conventional method. The black line presents the throughput for the direct link traffic and the gray line presents the throughput for the hop routing traffic, respectively.

TABLE I. AVERAGE PROCESSING TIMES FOR EACH STEP IN OTP

Step in OTP	Conventional	Proposed
1. Receive data	7.339 ms	4.869 ms
2. Read and remove used key	208.431 ms	7.307 ms
3. OTP encryption/decryption	0.842 ms	0.661 ms
4. Send data	1.308 ms	0.158 ms

For the network throughput evaluation, we measure the OTP throughput of the hop routing. For example, when the link between PC 1 and PC 2 is disconnected, the routing path from PC 1 to PC 2 is established as PC 1, PC 3 and PC 2. The hop routing throughputs as a function of virtual interface MTU size for the proposed method and the conventional method are shown in Fig. 5. The throughput of the proposed method achieves 123 Mb/s, which is sufficient, even if the traffic hops via a relaying node, whereas the conventional method is only 3.81 Mb/s.

C. Local Key Manager Processing Speed

For the evaluation, a QKD emulator is configured to generate the local keys at a rate of 1 Gb/s, which is sufficient for the communication over the 1 Gb Ethernet cable. The generated local keys are fed to the local key manager. We calculated the processing speed by measuring the timestamp and the size of the generated local key files. The processing speed is 414 Mb/s, which is sufficient for a high-speed QKD system.

D. Global Key Processing Speed

The processing speed of the global key manager is measured using the physical network interface. We calculated the processing speed by measuring the time until the global key storage is filled to the predetermined maximum size. As a result, the processing speeds vary depending on the size of the global key. When the global key size is configured to be 1 Mbytes, the processing speed becomes 85 Mb/s. This is sufficient for a high-speed QKD system. However, when the global key size is

configured to be smaller, the processing speed becomes unsuitable for a high-speed QKD system. The reason for this is the increased overhead of the global key sharing. To prevent this from limiting the QKD throughput, a large global key size should be used.

E. Key Providing Web API Throughput

For the evaluation, an application emulator pulls a number of global keys as often as possible at a given time. We calculated the throughput by measuring the time until all stored global keys were completely pulled from the DB with respect to each size of the global key. As a result, the throughputs vary depending on the size of the global keys. When the global key size is configured to be 1 Mbytes, the throughput becomes 971 Mb/s. This is sufficient for a high-speed QKD system. However, when the global key size is configured to be smaller, the throughput becomes lower. To prevent the QKD throughput from being limited, a large global key size should be used.

V. CONCLUSION

We proposed a key management method for the high-speed QKD network. This paper describes the design, implementation and evaluation of a new key management method. In recent years, study of high-speed key management methods for QKD networks has been insufficient, although the throughput for QKD has increased. We proposed a simple high-speed key management method and described the four modules. We improved the performance of the OTP tunnel manager, the most throughput-limited module. As a result, the throughput of the local key manager, the OTP tunnel manager, the global key manager and key providing web API reached 414 Mb/s, 185 Mb/s, 85 Mb/s and 971 Mb/s respectively, thus confirming that the proposed key management method can handle the throughput of a high-speed QKD system.

REFERENCES

[1] C. H. Bennet and G. Brassard, “Quantum cryptography: public key distribution and coin tossing”, Proceedings of IEEE International Conference on Computers Systems and Signal Processing, Bangalore India, pp. 175-179, December 1984.

[2] G. S. Vernam, "Cipher printing telegraph systems for secret wire and radio telegraphic communications," in Transactions of the American Institute of Electrical Engineers, vol. XLV, pp. 295-301, Jan. 1926.

[3] C. E. Shannon, "Communication theory of secrecy systems," in The Bell System Technical Journal, vol. 28, no. 4, pp. 656-715, Oct. 1949.

[4] C. Elliott, A. Colvin, D. Pearson, O. Pikalo, J. Schlafer, and H. Yeh, “Current status of the DARPA quantum network,” arXiv:quant-ph/0503058, March 2005.

[5] M. Peev, C. Pacher, R. Alleaume, C. Barreiro, J. Bouda, W. Boxleitner, T. Debuisschert, E. Diamanti, M. Dianati, J. F. Dynes, S. Fasel, S. Fossier, M. Furst, J.-D. Gautier, O. Gay, N. Gisin, P. Grangier, A. Happe, Y. Hasani, M. Hentschel, H. Hubel, G. Humer, T. Langer,M. Legre, R. Lieger, J. Lodewyck, T. Lorunser, N. Lutkenhaus, A. Marhold, T. Matyus, O. Maurhart, L. Monat, S. Nauerth, J.-B. Page, A. Poppe, E. Querasser, G. Ribordy, S. Robyr, L. Salvail, A.W. Sharpe, A. J. Shields, D. Stucki, M. Suda, C. Tamas, T. Themel, R. T. Thew, Y. Thoma, A. Treiber, P. Trinkler, R. Tualle-Brouri, F. Vannel, N. Walenta, H. Weier, H. Weinfurter, I. Wimberger, Z. L. Yuan, H. Zbinden, and A. Zeilinger, “The SECOQC quantum key distribution network in Vienna,” New Journal of Physics, vol. 11, no. 7, p. 075001, 2009.

[6] M. Sasaki, M. Fujiwara, H. Ishizuka, W. Klaus, K. Wakui, M. Takeoka, S. Miki, T. Yamashita, Z. Wang, A. Tanaka, K. Yoshino, Y. Nambu, S. Takahashi, A. Tajima, A. Tomita, T. Domeki, T. Hasegawa, Y. Sakai, H. Kobayashi, T. Asai, K. Shimizu, T. Tokura, T. Tsurumaru, M. Matsui, T. Honjo, K. Tamaki, H. Takesue, Y. Tokura, J. F. Dynes, A. R. Dixon, A. W. Sharpe, Z. L. Yuan, A. J. Shields, S. Uchikoga, M. Legre, S. Robyr, P. Trinkler, L. Monat, J.-B. Page, G. Ribordy, A. Poppe, A. Allacher, O. Maurhart, T. Langer, M. Peev, and A. Zeilinger, “Field test of quantum key distribution in the tokyo qkd network,” Optics Express, vol. 19, no. 11, pp. 10387-10409, May 2011.

[7] D. Stucki, M. Legré, F. Buntschu, B. Clausen, N. Felber, N. Gisin, L. Henzen, P. Junod, G. Litzistorf, P. Monbaron, L. Monat, J-B. Page, D. Perrou, G. Ribordy, A. Rochas, S. Robyr, J. Tavares, R. Thew, P. Trinkler, S. Ventura, R. Voirol, N. Walenta, and H. Zbinden, “Longterm performance of the SwissQuantum quantum key distribution network in a field environment,” New Journal of Physics, vol. 13, no. 12, p. 123001, 2011.

[8] S. Wang, W. Chen, Z. Q. Yin, H. W. Li, D. Y. He, Y. H. Li, Z. Zhou, X. T. Song, F. Y. Li, D. Wang, H. Chen, Y. G. Han, J. Z. Huang, J. F. Guo, P. L. Hao, M. Li, C. M. Zhang, D. Liu, W. Y. Liang, C. H. Miao, P. Wu, G. C. Guo, and Z. F. Han, “Field and long-term demonstration of a wide area quantum key distribution network,” Optics Express, vol. 22, no. 18, pp. 21739-21756, 2014.

[9] Y. Tanizawa, R. Takahashi, and A. R. Dixon, “A routing method designed for a Quantum Key Distribution Network,” 8th International Conference on Ubiquitous and Future Networks (ICUFN), pp.208-214, July 2016.

[10] M. Mehic, P. Fazio, M. Voznak and E. Chromy, “Toward designing a quantum key distribution network simulation model,” Advances in Electrical and Electronic Engineering, vol. 14, no. 4. pp.413-420, November 2016.

[11] M. Mehic, O. Maurhart, S. Rass, D. Komosny, F. Rezac and M. Voznak, “Analysis of the public channel of quantum key distribution link,” IEEE Journal of Quantum Electronics, vol. 53, no. 5, pp.1-8, Octorber 2017.

[12] C. Kollmitzer and M. Pivk, Applied quantum cryptography, Lecture Notes in Physics, vol. 797, Springer, pp 27-47, 2010.

[13] AIT Austrian Institute of Technology, AIT QKD Software, https://sqt.ait.ac.at/software/projects/.

[14] M. Pattaranantakul, A. Janthong, K. Sanguannam, P. Sangwongngam, and K. Sripimanwat, “Secure and efficient key management technique in quantum cryptography network,” 2012 Fourth International Conference on Ubiquitous and Future Networks (ICUFN), pp280-285, July 2012.

[15] H.-J. Briegel, W. Dür, J. I. Cirac, and P. Zoller, “Quantum repeaters: The role of imperfect local operations in quantum communication,” Physical Review Letters, vol. 81, pp. 5932–5935, Dec 1998.

[16] V. Scarani, H. BechmannPasquinucci, N. J. Cerf, M. Dusek, N. Lutkenhaus, and M. Peev, “The security of practical quantum key distribution,” Review of Modern Physics, vol. 81, no.3, p.1301, July-September 2009.

[17] E. Diamanti, H. K. Lo, B. Qi, Z. Yuan, “Practical challenges in quantum key distribution,” arXiv preprint arXiv:1606.05853, 2016.

[18] J. F. Dynes, S. W.-B. Tam, A. Plews, B. Fröhlich, A. W. Sharpe, M. Lucamarini, Z. L. Yuan, C. Radig, A. Klar, A. Straw, T. Edwards, and A. J.Shields, “Ultra-high bandwidth quantum secured data transmission,” Scientific Reports, vol. 6, p. 35149, 2016.

[19] Z. L. Yuan, A. Plews, R. Takahashi, K. Doi, W. Tam, A. W. Sharpe, A. R. Dixon, E. Lavelle, J. F. Dynes, A. Murakami, M. Lucamarini, Y. Tanizawa, H. Sato, and A. J. Shields, “10 Mb/s quantum key distribution,” Journal of Lightwave Technology, vol. 36, no. 16, pp. 3427-3433, Aug 2018.

[20] Y. Tanizawa, R. Takahashi, H. Sato, A. R. Dixon, and S. Kawamura, “A secure communication network infrastructure based on quantum key distribution technology,” IEICE Transaction on Communications, vol. E99-B, no. 5, May 2016.

[21] L. Salvail, M. Peev, E. Diamanti, R. Alleaume, N. Lutkenhous, and t. Langer, “Security of trusted repeater quantum key distribution networks,”, Journal of Computer Security, vol. 18, no,1, pp. 61-87, 2010.

[22] Universal TUN/TAP device driver, https://www.kernel.org/doc/Documentation/networking/tuntap.txt.

[23] REST API, http://www.ics.uci.edu/%7Efielding/pubs/dissertation/rest_arch_style.htm.