Software Defined Quantum Key Distribution Network

Wanrong Yu, Baokang Zhao, Zhe Yan
College of Computer
National University of Defense Technology
Changsha, China
e-mail:{wlyu, bkzhao, zhyan}@nudt.edu.cn

***Abstract*—Based on the analysis of quantum key distribution (QKD) scenarios, existing point-to-point QKD systems need to be networked for satisfied actual applications. However, the architecture and network layer protocols of existing QKD network have lots of limitations, such as excessive key consumption and inability to achieve random routing. In this paper, we propose the software defined quantum key distribution network (SQN). We introduce the design of the architecture and critical modules for SQN. Analysis results show that SQN can efficiently reduce the secure key consumption and improve the availability and performance of QKD network.**

Keywords-software defined quantum key distribution network; random routing; key consumption

I. INTRODUCTION

Quantum Key Distribution (QKD) technology is one of the fastest growing technologies in the field of quantum information, which is expected to achieve the first large-scale and widely used application. Large-scale QKD applications require extending existing local QKD networks to wider area to better satisfy the requirements of the multiple-access secure communication scenario.

In 2004, the America Defense Advanced Research Projects Agency (DARPA), United BBN Corp., Boston University and Harvard University jointly completed the construction of the world's first quantum cryptography network [1]. In 2008, the SECOQC quantum network [2] which has six nodes and eight links was established in Vienna. In 2010, TOSHIBA, Switzerland IDQ and several Austria groups set up in collaboration the Tokyo QKD Network [3]. University of Science & Technology China (USTC) also established the quantum-government network [4], Beijing to Shanghai Quantum Backbone network [5] and so on. The first satellite-to-ground QKD experiment has been established in 2016 [6].

According to the network construction, existing QKD networks can be divided into three categories: passive optical network, quantum repeater based network, and trusted relay based network [7]. To make the works more technologically feasible, DARPA, SECOQC and Tokyo QKD networks are trusted relay based, while USTC constructing both passive optical network and trusted relay network.

The architecture of trusted relay based QKD network is shown in Fig. 1. Through the trusted relay nodes, multi-hop QKD network link can be divided into several one-hop links, which can generate local secure keys through point-to-point QKD technique, such as $K_{1\text{-}2}$ is the key between Node1 and Node2. Alice and Bob want to share the session key $K_{Alice\text{-}Bob}$. $K_{Alice\text{-}Bob}$ is first transferred from Alice to Node1 encrypted by the local key $K_{Alice\text{-}1}$ between Alice and Node1. Node1 decryption and re-encrypt $K_{Alice\text{-}Bob}$ using the local key $K_{1\text{-}2}$ between Node1 and Node2. And so on, $K_{Alice\text{-}Bob}$ is transferred from Alice to Bob finally.

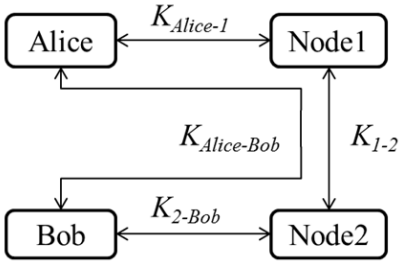

Figure 1. Trusted relay based QKD network.

Similar to the classical hierarchical TCP/IP protocol suite, SECOQC network also is divided into five layers [8]: physical layer, link layer, network layer, transport layer and application layer. Wherein, the network layer is responsible for discovering optimal path in the QKD transmission network for exchanging the classical information of the post-processing procedures and session keys. In the SECOQC network, the network layer protocol OSPF faces the high resource consumption problem, and is difficult to achieve random routing and complex network management. To solve the above problems, we apply the software-defined technology to the QKD network and issue the Software-defined Quantum key distribution Network (SQN).

The rest of the paper is organized as follows. In Section II we introduce the routing protocol of typical QKD networks. The architecture and critical modules of the SQN are detailed in Section III, followed by the implementation of prototype and the simulation of SQN in Section IV. Finally, in Section V, we draw our main conclusions with the future work.

II. RELATED WORKS

A. Existing QKD Network

Existing quantum key distribution networks all conduct the similar network layer protocol as the classic network,

wherein the SECOQC is the most typical QKD network. The network layer of SECOQC is shown in Fig. 2 [9].

Routing module: Deep customization OSPFv2 protocol is conducted in this module. First, each network node maintains a local interface database, that collects the key consumption of links connected to the node; then, each node calculates the network topology through the broadcasted link state information and stores the topology information in the link-state database; Finally, the route calculation sub-module calculates the shortest path from the current node to each other node in the network according to the network topology and stores the entries in the routing table.

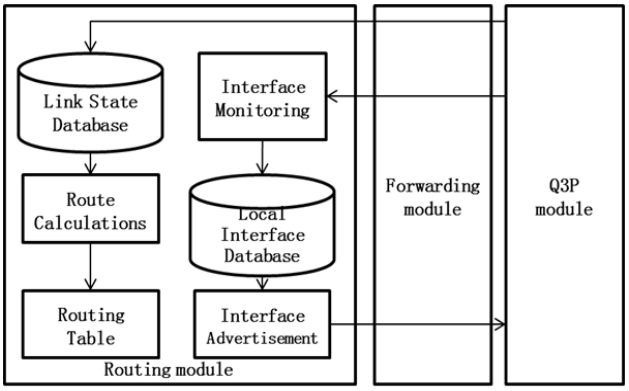

Figure 2. SECOQC network layer modules [8].

Forwarding module: The forwarding module resolves the coming packet header and forwards the packet to the expected interface according to the destination address and the forwarding table.

The design of SECOQC network, makes full use of the advantages of the existing Internet network layer experience, but does not consider the characteristics of QKD systems and the differences with the Internet lead to the following questions:

1) Excessive key consumption. We need to send large amounts of link state advertisement packets, while in the QKD network these packets need to be encrypted or authenticated default, which leads to the consumption of large amount of local secure key.

2) It is difficult to achieve random routing. According to the security theory of the trusted relay network [10], random routing for session key transport is necessary to ensure the safety, while this cannot be done in the current QKD network.

B. Classic Software-Defined Network (SDN)

To propose an efficient network layer protocol of QKD network, one can learn from the novel achievements of SDN technologies in classical computer network [11], in order to efficiently solve the problems of excessive key consumption and random routing.

As shown in Fig. 3, the core idea of SDN is to separate the control information and the application data. In the control layer, the logical centralize controller is programmable, and can obtain the global network information to realize centralized network management and configuration. The data layer is composed of Openflow switches, which can complete the data forwarding function according to the information of the controller. The controller sends the flow table information to the switch through the standard interface, thus the switch only need to transmit data packet quickly according to the flow table. With this architecture, SDN technology can efficiently reduce the load of equipment, simplify the network control and management, and reduce the cost of network maintenance [13].

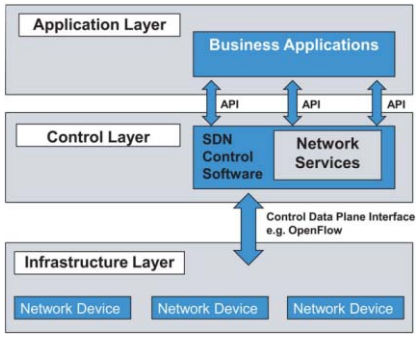

Figure 3. SDN architecture [12].

Introduced the idea of SDN to the QKD network, we propose the Software-defined Quantum key distribution Network (SQN), which mainly has three advantages:

1) Conserve secure key resources. In SQN, the link state information needs only be sent to the controller, will greatly reduce the consumption of the secure key for updating the routing information.

2) Achieving random routing. With the APIs provided by the SQN control layer, SQN can support a powerful network control applications, such as achieving the random routing.

3) Global link monitoring. In SQN, the controller can increase the key management module to realize the control of the link state of the whole network, such as the key generation rate and the quantum bit error rate, so as to monitor the security status of the entire SQN network.

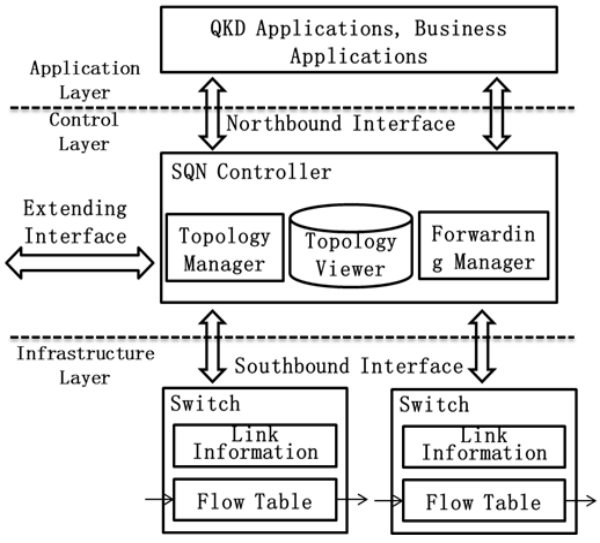

Figure 4. SQN architecture.

III. THE DESIGN OF SQN

A. The Architecture of SQN

Based on the existing SDN technology, combined with the characteristics of quantum key distribution network, we propose the architecture design of SQN, as shown in Fig. 4.

The whole SQN is divided into three layers: infrastructure layer, control layer and application layer. The infrastructure layer is composed of SQN switches which are deployed on the relay nodes. The control layer is composed of SQN controller. SQN switch and SQN controller communicate through the southbound interface, the application layer access the SQN controller through the northbound interface to manage the network.

a) The infrastructure layer

The infrastructure layer is composed by the trusted relay nodes, which called as the SQN switch, which is mainly composed of two functions: link information collection and session key forwarding.

Link information collection. According to the characteristics of the QKD system, it is necessary to collect information about the remain local key, on/off state of the link, generation rate of the link local key, error rate of the link key and so on. Such information can be obtained from a quantum network link layer, such as the Key Store module in Q3P [5], and is aggregated to the control layer from the infrastructure layer through the southbound interface.

Session key forwarding. In SQN, the infrastructure layer needs to maintain the flow table, and its entries are issued by the controller. The infrastructure layer receives the session key message, parses its destination address and forwards it to the right interface according to the matching result.

b) The control layer

In SQN, the control layer is composed by the controller, which is also located in the trusted relay node of the SQN network, so the routing information between the controller and the subordinate switch also needs the local key encryption or authentication. The controller performs two tasks: topology management and routing management.

Topology management. Node information and link information of the infrastructure layer are submitted to the controller, then the controller needs to summarize this information to get a complete view of the network, which contains the key generate rate, key error rate, remaining key and other information of each link.

Routing management. The flow table is the basis for the data exchange of SQN switch, which directly affects the efficiency of data forwarding and the performance of the whole network. The flow table is generated by the controller based on the global network view, or by the application layer according to the application requirements, and then sent to the controller through the northbound interface. Once the flow table generated, the controller sends the flow table information to all SQN switches along the transmission path of the data flow through the active or passive mode.

B. The Critical Modules of SQN

a) Topology management function

Through collecting and summarizing the SQN switch link information, the controller integrates the node and link information of whole network, thus can maintain and manage the global network topology. According to the topology information, the controller can provide data support for the routing computation and application development. This process is called topology management.

Fig. 5 shows the sub-modules involved in topology management. The link information sub-module in the switch collects link monitoring information from the link layer. Then, the switch reports the link information to the controller, or the controller regularly inquires the switch for the link information. Thus, the controller topology management module can get the whole network topology information.

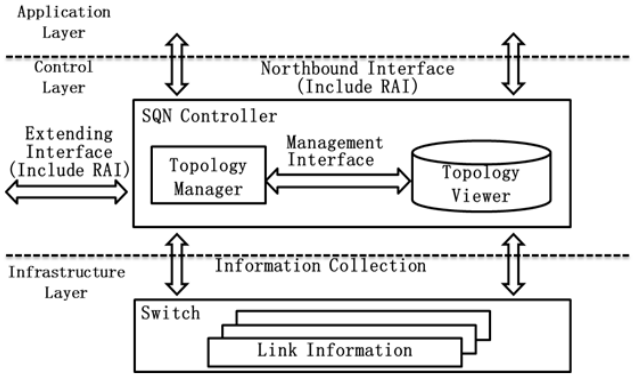

Figure 5. Topology management associated modules.

The topology manager module stores the network topology in the topology viewer database and provides access API to the database. The northbound interface needs to implement the resource access interface (RAI) to provide the interested user with the network information. The extended interface module collects the topology information of the network and reports them to the high-level controller.

b) Forwarding management function

Based on the topology information of the network, the control layer or the application layer calculates the flow table entries and sends them to the infrastructure layer. The infrastructure layer accomplishes the forwarding of the packets according to the matching information in the flow table. This process is called forwarding management.

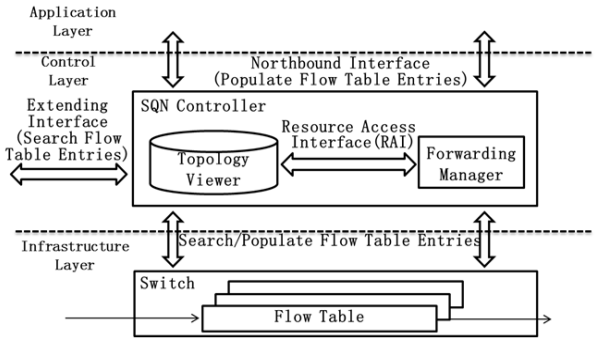

Figure 6. Forwarding management associated modules.

Fig. 6 shows the SQN sub-modules involved in the forwarding management.

There are two schemes to generate a flow table entry. The first scheme: the forwarding management sub-module in the controller access the topology database through the resource access interface to get the view of the whole network. Then, according to the routing policy deployed, global optimal routing strategy can be calculated and stored in the corresponding database. The second scheme: the network administrator access to the network topology data through the northbound interface, and then according to the needs of the application, to generate a specific flow table entries and return the flow table entries to the controller's routing database.

Through the active inquiry of the controller from the switch or the periodical population of the controller, the switch in the data layer can update the flow table in time. Once received the packet, the switch parses the packet to get the destination address and match the address against the flow table. According to the matching result, the packet is forwarded to the right interface. When the destination address is not within the subnet which is controlled by the local controller, the local controller cannot deal with the packet and should query the high-level controller through the extended interface. The high-level controller collects the network view of multiple subnets, and returns the corresponding flow table entries to the lower level controller. The lower level controller then sends the result to the corresponding switch.

IV. PROTOTYPE AND EVALUATION

We now build the prototype of SQN, implement the core module and evaluate the performance.

A. Design of the Prototype System

We conduct the existing SDN platform to simulate SQN, the logic structure of the prototype network is shown in Fig. 7.

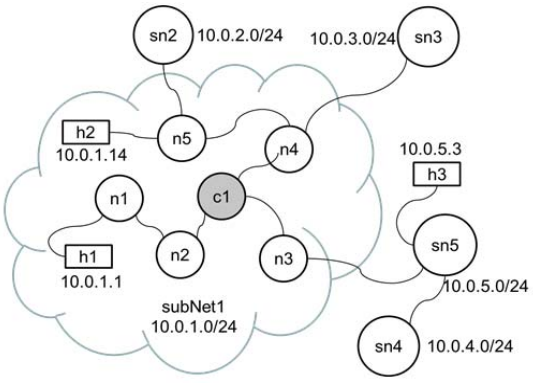

Figure 7. The structure of the prototype system.

The whole SQN network is divided into 5 subnets, which are identified by sn1-sn5, and each subnet is assigned a network address. Each subnet has some network nodes (such as n1, n2, n3) and a corresponding low level controller (such as c1) which is also a network node in the subnet. Each network node has its own host number and IPv4 address.

The implementation of the prototype is based on mininet, and the controller, switch and protocol are as shown in Fig.8.

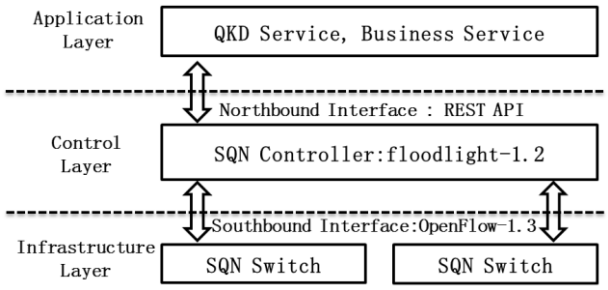

Figure 8. The implementation of the prototype.

B. Forwarding Management Function

According to the communication requirements and network topology, a reasonable flow table should be generated and sent to the corresponding switch. Here, the flow table entries are generated by the application layer and populate to the switch through the controller.

The generation of flow table entries is designed as shown in Algorithm 1.

Firstly, the network topology *G(V, L)* is obtained from the controller through the REST API, and *(Vproduce, Vconsume, Kstorage, Esecurity)* represents the key generation rate, the key consumption rate, the remaining key, and the security of the link respectively.

Algorithm 1: the generation of flow table entries

INPUT: network topology data G(V, L), which includes node information V[], link information L[] = (V1-V2, Vproduce, Vconsume, Kstorage, Esecurity), source node Vsrc and destination node Vdst.
OUTPUT: flow table entry[]
/*Get topology data *G(V, L)* through the REST API*/
1: Curl -s http://<ip>: 8080/wm/topology/all/json
/*Generate shortest path for each interface*/
2:FOR each eth-i IN Vsrc
/*Shield other network interfaces*/
*3: SetDown (eth-except-*i)
4: shortestPath = Dijkstra(Vsrc,Vdst)
5: FOR each Link IN shortestPath
/*If the link meet the requirements, add to routing path */
6: IF (Vproduce, Vconsume, Kstorage, Esecurity) meet requirements
*7: Path[] add (*shortestPath*)*
8: ENDIF
9: ENDFOR
10: ENDFOR
11: FOR each shortestPath IN Path[]
12: R[] = FlowCalculate (shortestPath)
/*Generate a flow entry*/
13: IF R[j].switch == R[i].switch AND R[j].outport != R[i].outport
/*Set the probability */
14: set R[j].p
15: ENDIF
16: ENDFOR
17: FOR each R in R[]
/*Populate flow entry through the REST API*/
18: Curl -d http://<ip>:8080/wm/staticflowpusher/json
19: ENDFOR

Then, combining the network topology, the source node and destination node, the Dijkstra algorithm can be used to

calculate the shortest path between two nodes. Here, the "shortest" means the link loss is smallest, this paper assumes that the higher the link key generation rate (*Eproduce*), the smaller the link loss.

Random routing requires a number of paths to be used randomly, so for each eth interface of the source node, one shortest path should be generated. After the generation of shortest path, it is necessary to judge each link on the path. If the key reserves are not enough, or the security cannot be guaranteed (bit error rate is greater than the threshold) for one link in the path, then corresponding path should be given up. Then, one independent flow table entry is generated for each remaining path.

For the purpose of random routing, the flow table entry of different interfaces of the same node needs to attach a probability value. Thus, when the switch receives the target packet, it can be delivered to a different interface randomly.

Finally, the REST API is used to send the flow table to the controller, and the controller is responsible for sending the flow table to the corresponding switch.

C. Comparison of Key Consumption

This section compares the amount of key consumed by SECOQC and SQN during the network topology management. In application, the network topology will not change frequently, but the link state information needs to be broadcast regularly, such as once per 30s. Since link state information needs to be encrypted or authenticated, this process consumes a large number of local keys.

SECOQC is based on the OSPFv2 protocol, so the relay node needs to broadcast the link state to all nodes in the domain. While in SQN, the relay node only needs to send the link state information to the controller. Thus, the amount of keys consumed in SQN will be greatly reduced.

The following is a comparison of the key consumption in different topology, as shown in Fig. 9.

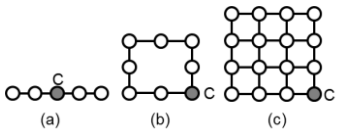

Figure 9. Three topologies.

Suppose single link state information needs to broadcast one data and consume one key. Thus, in the line topology of Fig. 9(a), as the controller is deployed in one node in the line, so the total number of data needs to broadcast is 6 for the other 4 nodes report the link state to the controller in SQN. While with OSPFv2 in SECOQC, the number is 20. Fig. 9(b) is the ring topology, and the number of data needed for SQN and SECOQC is 16 and 64 respectively. In the network topology of Fig. 9(c), the difference widened to 29 and 384.

From above, we can see that with more number of nodes in the topology, SQN is more economical than SECOQC during topology update. It should be noted that, if OSPF is used, the network nodes can calculate the routing table after the topology updating, no longer consume the local key. While in SQN, after the controller calculates the flow table, it should send flow table to the corresponding switches and this process also consumes the local key. However, considering that the flow table only needs to be sent to the required relay node and the frequency of updating the flow table is much lower than the frequency of link state reporting, the total amount of local key consumed in SQN topology management will be much less than the existing QKD network based on OSPF.

V. CONCLUSIONS

In this paper, we propose the concept of software defined quantum key distribution network based on the software defined network technology. In the proposed SQN architecture, the corresponding topology management and forwarding management module are designed. Analysis results show that the design of SQN can efficiently reduce the consumption of local key and highly improve the availability and performance of QKD system.

However, the existing SDN software cannot satisfy the requirements of future SQN, so one of the future work is the development of dedicated SQN switches and controllers.

ACKNOWLEDGMENT

This work was supported in part by the National Natural Science Foundation of China under Grant No. 61379147.

REFERENCES

[1] Elliott C, Pearson D, Pikalo O. Current status of the DARPA quantum network. Proc of Spie, 2005, 5815(138-49).

[2] Peev M, Pacher C, Allaumer, et al. The SECOQC quantum key distribution network in Vienna. New Journal of Physics, 2009, 11(7): 075001.

[3] Sasaki M, Fujiwara M, Ishizuka H et al. Field test of quantum key distribution in the Tokyo QKD Network. Optics Express, 2011, 19(11): 10387-10409.

[4] Shuang Wang, Wei Chen, et al. Field and long-term demonstration of a wide area quantum key distribution network. Optics Express Vol. 22, Issue 18, pp. 21739-21756 (2014)

[5] Liu-Jun Wang, Kai-Heng Zou, et al. Long-distance copropagation of quantum key distribution and terabit classical optical data channels. Phys. Rev. A Vol. 95, Issue 1, pp. 012301-012308 (2017).

[6] Liao, S.-K., Cai, W.-Q., Liu, W.-Y., *et. al.*: Satellite-to-ground quantum key distribution. Nature advance online publication, (2017).

[7] Elliott C. Building the quantum network. New Journal of Physics, 2002, 4(1): 46.

[8] Maurhart O. QKD networks based on Q3P. Springer Berlin Heidelberg, 2010.

[9] Dianati M, All aume R, Gagnaire M, et al. Architecture and protocols of the future European quantum key distribution network. Security and Communication Networks, 2008, 1(1): 57-74.

[10] Wen Hao. Protocols and mechanisms in the quantum key distribution networks [A dissertation for doctor's degree]. University of science and technology of China, 2008 (in Chinese).

[11] Mckeown N, Anderson T, et al. OpenFlow: enabling innovation in campus networks. Acm Sigcomm Computer Communication Review, 2008, 38(2): 69-74.

[12] Fei Hu, Qi Hao, and Ke Bao. A Survey on Software-Defined Network and OpenFlow: From Concept to Implementation. IEEE Communication Surveys & Tutorials, Vol. 16, No. 4, Fourth quarter 2014.

[13] Zhang CK, Cui Y, Tang HY, Wu JP. State-of-the-Art survey on software-defined networking(SDN). Journal of Software, 2015, 26(1): 62−81 (in Chinese).