SECURITY AND COMMUNICATION NETWORKS
Security Comm. Networks. 2008; **1**:57–74
Published online in Wiley InterScience
(www.interscience.wiley.com) DOI: 10.1002/sec.13

Architecture and protocols of the future European quantum key distribution network

Mehrdad Dianati[1*,†], Romain Alléaume[1], Maurice Gagnaire[1] and Xuemin (Sherman) Shen[2]
[1]*GET-ENST, Network and Computer Science Department, CNRS UMR 5141, 46 rue Barrault F-75634, Paris, Cedex 13, France*
[2]*Department of Electrical and Computer Engineering, University of Waterloo 200 University Ave. West, Waterloo, ON, Canada, N2L3G1*

Summary

A point-to-point quantum key distribution (QKD) system takes advantage of the laws of quantum physics to establish secret keys between two communicating parties. Compared to the classical methods, such as public-key infrastructures, QKD offers unconditional security, which makes it attractive for very high security applications. However, this unprecedent level of security is mitigated by the inherent constraints of quantum communications, such as the limited rates and ranges of an individual point-to-point QKD link. A QKD network, which can be built by combining multiple point-to-point QKD devices, can alleviate the constraints and enable point-to-multi-point key distribution based on QKD technology.

The European project, secure communication based on quantum cryptography (SeCoQC) aims at deploying a prototype QKD network, which will be demonstrated in September 2008, by developing the architecture and the protocols, as well as the specific hardware for long-range QKD networks. This paper discusses the important aspects of the architecture and the network layer protocols of the SeCoQC QKD network. Copyright © 2008 John Wiley & Sons, Ltd.

KEY WORDS: network security; quantum key distribution

1. Introduction

Implementation of many cryptographic schemes, such as encryption and authentication algorithms, relies on a proper *key distribution* scheme among the legitimate communicating parties. The strength of a cryptographic algorithm is directly linked to the difficulty of obtaining the secret key by the adversaries; thus, key distribution schemes can be identified as one of the most sensitive parts of the security systems in communication networks. Therefore, it is highly desirable to develop unconditionally secure key distribution systems, i.e. the adversaries cannot obtain shared secrets regardless of their computing resources.

Classical key distribution schemes either rely on *trusted couriers* or are developed based on the principles of *public-key cryptography*. The trusted courier method is known since the ancient times. A trusted courier travels between different legitimate users to distribute the secret keys, hoping that he will not be intercepted or corrupted by the adversaries. Although trusted courier is considered

*Correspondence to: Mehrdad Dianati, ENST, Network and Computer Science, Paris, France.
†E-mail: m.dianati@surrey.ac.uk

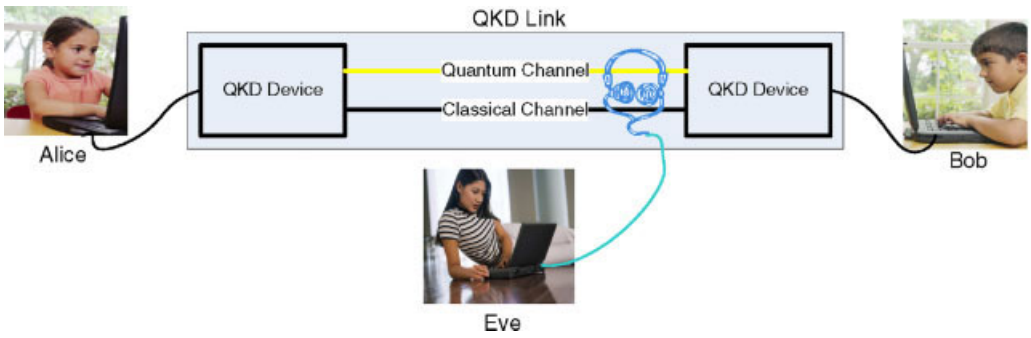

Fig. 1. A basic QKD system.

as a primitive method, it is still popular for highly sensitive applications, such as the government and intelligence information, where other techniques cannot convince sceptical users. Despite the continuing interest in trusted courier method, only practical security can be invoked by this approach, which has to be backed by enforcement of many security measures. In addition, trusted couriers become costly and unpractical for large systems. Public-key cryptography rely on the difficulty of solving certain mathematical problems within a reasonable time. The required computing resources to solve these problems become practically unreachable when long keys are used.

The most popular public-key cryptographic based key distribution algorithm is the Diffie–Hellman (D–H) key exchange [1] (has been independently developed by Merkle as well [2]). In this approach, two parties, who desire to establish a shared secret, choose their own private keys and compute their public keys using a certain publicly known one-way-function. Knowing the public key of the other party and his own private key, each party can easily compute the shared secret. However, an adversary has to solve the inverse of the one-way-function, which requires exponentially growing computing resources as the length of key increases. In real communication systems, usually public keys have to be certified by a trusted certification authority. Key distribution schemes based on public cryptography suffer from few significant weaknesses: (1) they can only offer computational security; (2) finding efficient algorithms to compute the inverse of one-way-functions has not been proved to be impossible and (3) emerging powerful computers, e.g. quantum computers, would pose real threats to their security.

Quantum key distribution (QKD), invented by Bennett and Brassard [3] in 1984, based on some earlier ideas of Wiesner [4], is an alternative solution for key distribution. Relying on the principles of quantum mechanics, a basic QKD system, as shown in Figure 1, can establish a shared secret with unconditional security between two communicating parties, namely Alice and Bob [5,6]. Security of QKD relies on the fact that it is impossible to gain information about non-orthogonal quantum states without causing detectable perturbation [7]. Regardless of the computing capacity of Eve, she cannot obtain information about the shared secret between Alice and Bob. As shown in Figure 1, a basic QKD system deploys a pair of QKD devices which are connected by a quantum and a classical channels. After the transmission of information over the quantum channel, the classical channel is used to perform error correction and privacy amplification. If the protocol succeeds, Alice and Bob establish unconditionally secure keys with each other. Note that any attempt by Eve to gain information about the secret keys will create detectable and correctable errors according to the principles of quantum mechanics.

Practical deployment of QKD systems is currently limited by few technical challenges. QKD links can only operate over point-to-point connections. Furthermore, the current QKD links have limited key generation rate and cannot be deployed over arbitrarily long distances. Development of QKD networks appears from this perspective a necessary step towards effective deployment of QKD. A network of QKD devices can extend over longer distances. Furthermore, a network structure enables resource sharing in order to improve utilisation.

There are three approaches for developing QKD networks: (1) network of optical nodes; (2) network of quantum nodes and (3) network of trusted relaying nodes. Optical nodes use beam splitting, switching, multiplexing and demultiplexing in order to create point-to-multi-point QKD systems [8]. Active optical switching can also be used to allow selective connection of any two QKD nodes with a direct quantum channel. The BBN Darpa quantum network

DOI: 10.1002/sec

[9] contains an active 2-by-2 optical switch in one node. This networking model however cannot extend the reachable distance of QKD. Indeed, the extra amount of optical losses introduced inside the optical nodes will shorten the maximum span of quantum channels. The second approach is based on quantum nodes, which are able to propagate entangled quantum states through quantum memories. This technique can effectively compensate the degradation of quantum signal along its propagation through a quantum channel and extend the range of QKD systems [10–14]. However, building practical quantum relays remains a technically difficult problem. The third approach is based on classical trusted relays. QKD networks based on trusted key relays follow a simple principle: local keys are generated over point-to-point QKD links connecting adjacent QKD devices. Session keys, which are finally used for secure communication between a pair of applications, are generated by one side and forwarded to another party over the key distribution network. A typical path may comprise multiple QKD nodes and links. Local keys among adjacent QKD devices, generated by QKD technology, are used to encrypt the session keys before forwarding to next hop. One-time-padding between adjacent QKD nodes is used to ensure unconditional the session keys. Variants of this networking approach are used in the BBN QKD network [8] and the SeCoQC QKD network, as well.

The objective of the SeCoQC project is to develop the first European QKD network for long-range high security communications. In this project, different from the previous works, we have opted to improve the resilience of network against varying session key traffic load by implementing large key stores in the trusted nodes. Adjacent QKD devices always operate at maximum key generation rate when the corresponding key stores are not full. If the network is in underload-state, the surplus is stored in key stores which can be used when the session key traffic over a link is more than its maximum key generation rate. Due to certain application and constraints of the SeCoQC QKD network, we propose a customised architecture and protocol stack in order to enable efficient consumption of key materials. The architecture and protocol models are inspired by the Internet model. In this paper, we introduce the networking structure and relevant details of the network layer protocol of the SeCoQC QKD network.

The rest of this paper is organised as follows. In Section 2, we discuss the architecture and the components of the SeCoQC QKD network. The important network protocols are described in Section 3. In Section 4, some concluding remarks are given.

2. Architecture of the SeCoQC QKD Network

In this section, we introduce the architecture, the major protocols, and the components of the SeCoQC QKD network. We discuss the static structure of the network, the addressing scheme and the architecture of individual components. Some related data structures are also specified.

2.1. QKD Network Structure

The high level architecture and the topology of the prototype SeCoQC QKD network are shown in Figure 2. The network consists of quantum backbone (QBB) nodes and links which are enhanced QKD devices and links as specified in the following subsections. The entire structure provides a backbone network for key distribution. QBB nodes and links are similar to routers and point-to-point communication links in a conventional network. The client computers are connected to the QBB nodes across the network inside private and secure sites. Alternatively, a client computer can be connected to a quantum access node (QAN) which a specialised access node with limited routing functionalities.

The application programs that intend to have secure communications over a public network, such as the Internet, use the QKD network for establishing unconditionally secure session keys. In a typical scenario, the QKD network does not carry real data traffic among application programs. The sole responsibility of the QKD network is to distribute secret keys with unconditional security. The main data communications, as shown in Figure 2, are performed over the public networks such as the Internet. Upon the request of an application program, namely the master application program, the ingress QBB or QAN node, which is the access point for the master application program, examines the possibility of providing a path for the requested destination. The destination application program, namely the slave application program, must also be connected to another QBB or QAN node, which is referred to by the egress QBB or QAN node. If the QKD network has enough resources, the ingress node accepts the request; otherwise, the request is rejected. If the path is successfully established, the ingress QBB or QAN

DOI: 10.1002/sec

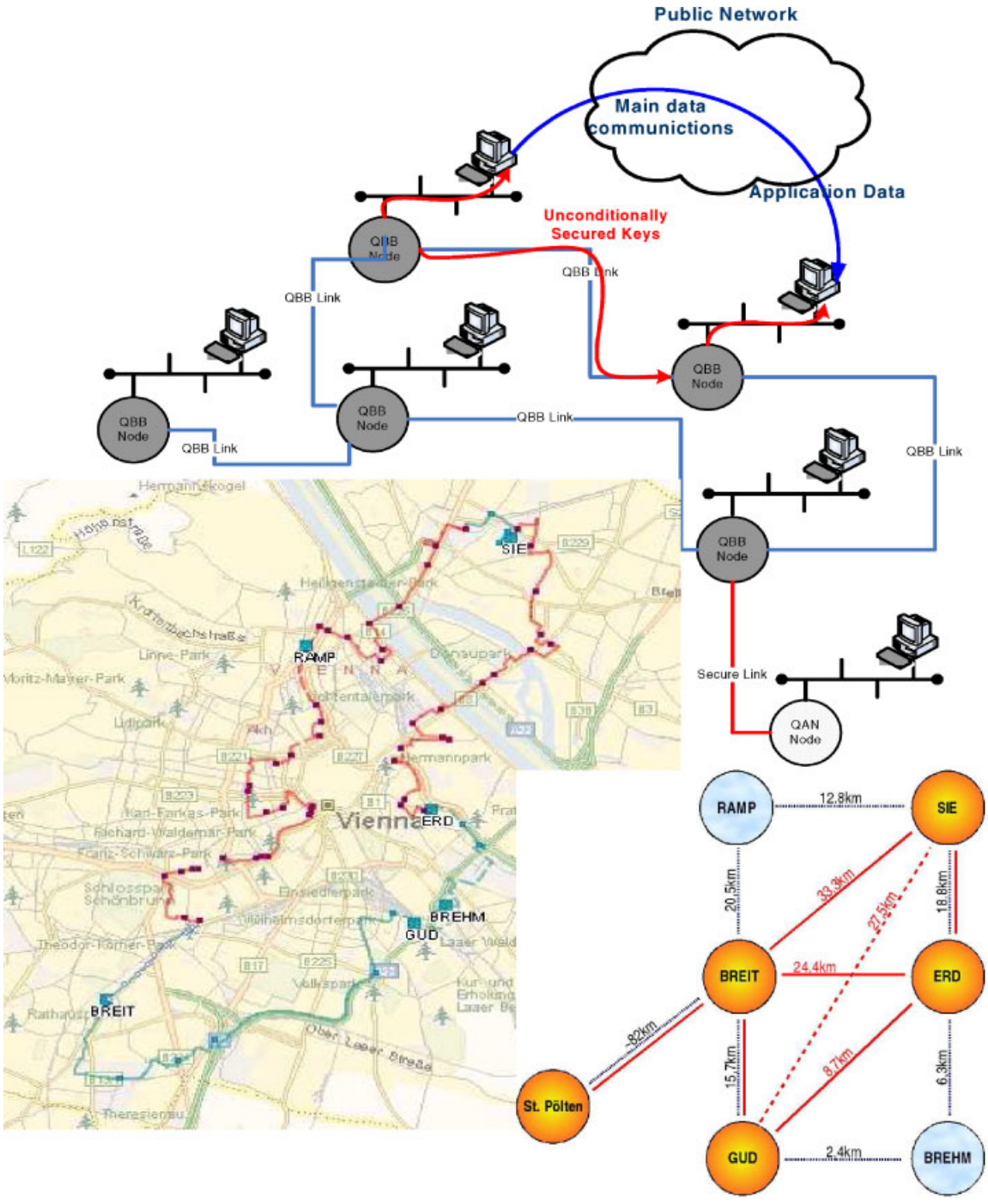

Fig. 2. The structure of the SeCoQC QKD network.

node generates random session keys and forwards them through the QKD network to the egress QBB/QAN node, which delivers the keys to the slave application program. In its path through the QKD network the session keys are secured through hop-by-hop one-time-padding, using the local keys. Local keys, that are unconditionally secure, are established over QBB links between adjacent QBB nodes. As shown in Figure 2, QBB nodes are located inside private and secure sites. Regarding that the local keys are unconditionally secure and that the QBB nodes are inside secure locations, we can conclude that the QKD network

DOI: 10.1002/sec

can be used for distribution of unconditionally secure session keys. Note that the QBB links, which connect adjacent QBB nodes, are deployed over potentially unsecured locations; thus, they are vulnerable to different types of security threats. However, hop-by-hop one-time-padding and authentication assure security of session keys. Furthermore, redundant paths over the QKD network can be utilised by the routing and forwarding protocols to combat denial of service (DoS) attacks.

In the following subsections, we provide further details of the important design aspects and components of the SeCoQC QKD network.

2.2. The QKD Network Services

In this subsection, we discuss the QKD network services and their requirements. The basic service of the QKD network is to provide unconditionally secure and synchronised keys which are used for secure communications over public networks. In a typical scenario an application program, running on a host computer which is connected to a QBB or QAN node, intends to open a secure connection to another application, running on another host which is also connected to another QBB or QAN node. An example could be an email client which initiates a SSL connection to a mail server. Another example is a security gateway attempting to establish a VPN connection through IPSec to another security gateway. Currently, these applications rely on a preset shared secret or certified public keys to generate session keys. However, they can be modified to send their request to a QBB/QAN node to establish a path through the QKD network to the other end point. A key request should contain information about:

(1) The address of the destination node;
(2) The port number of the destination application;
(3) The quality of service (QoS) requirements such as key refreshment rate.

In terms of QoS, the QKD network provides the following service types:

(1) *Best effort:* This service type will be supported by the first version of the SeCoQC QKD network which is specified in this paper. An application is allowed to transmit session keys with an average rate, λ_k, and burst of σ_k. The QKD network do not provide any guarantee of QoS. The (λ_k, σ_k) pair only specifies the upper bounds on the request of the customer. An application with this type of service may receive less than the specified upper bound. Note that, regardless of the type of service, the QKD network provides a reliable key distribution network.
(2) *Guaranteed key rate:* For the next version of the SeCoQC QKD network, we aim to enable guaranteed rate key delivery services, in addition to the best effort service. For this class of services, the QKD network guarantees a certain rate of session keys. For instance, an application may request 128 bits of key materials every 1 second.

An important non-functional requirement of QKD network is load balancing by choosing proper paths which might not be the shortest paths. This is important as the QBB links have limited capacities. This also implies that the algorithms and schemes that heavily rely on encrypted or authenticated signalling messages will not be reasonable choices for the QKD network. The routing and forwarding algorithms also should be resilient against link failure. Failures may occur due to the detection of security threats by some security monitoring agents or physical problems on QBB links and nodes.

2.3. Addressing

Addressing is an important design decision with direct impacts on the efficiency and scalability of a network. As we have learned from the evolution of the Internet, careless design of addressing structures can lead to inefficient utilisation of address space and cumbersome routing algorithms which may not scale easily. Fortunately, there is a significant amount of results, tools and experience from the addressing issues of the Internet that can be used and customised for the QKD network. To take advantage of them, we propose an addressing scheme that is compatible with the existing addressing structure of the Internet. This will enable us to use a variety of the existing tools and schemes to accelerate the development process. It is important to note that the addressing of the QKD network is completely independent of the addressing of the public network in Figure 2.

We do not expect the QKD network to expand like the Internet in terms of number of nodes. An addressing structure similar to IPv4 thus seems to be a reasonable choice for the QKD network. IPv4 addresses are 32-bit identification numbers which are conventionally represented in a dotted format shown in Figure 3. Each octet is an 8-bit integer number

DOI: 10.1002/sec

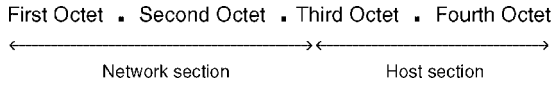

Fig. 3. IPv4 addressing format.

between 0 and 255. An IP address has two sections: a network section to identify the network; and a node section to specify a certain host inside a network. Inside routing tables, sub-net masks or integer suffixes specify the network portion of an IP address. For the latter case, the network suffix proceeds the node address. For example, 192.168.1.5/30 indicates that the leftmost 30 bits of the address represent the network address and the remaining two rightmost bits represents the node address. The process of designing the addressing structure of a network is about making key decisions regarding the division of existing address space.

There are few differences between addressing design of conventional networks and the addressing design of the QKD network. First, unlike the Internet, the QKD network is a private network. The available address space in the Internet is specified by the address allocation authorities. On the contrary, for the QKD network, there is more freedom of using any area of the available IPv4 address space. Second, the hierarchical topology of the conventional networks is not a feasible choice for the QKD network. That is, we cannot divide the network into a backbone and an arbitrary number of autonomous systems. This is due to the fact that it is difficult to make very long range and high capacity QBB links similar to the backbone links in classical networks. This imposes a significant topological constraint on the architecture of the QKD network. It is also an important factor in designing an efficient addressing structure to reduce the complexity of the routing algorithms. Finally, unlike the routers in a conventional networks that only forwards traffic to the host machines, in the QKD network, the QBB nodes are not only forwarding devices but also the access points for client applications. These discriminating factors impose certain constraints which are considered in the following proposal of the addressing structure for the QKD network.

The QKD network is geographically divided into multiple routing areas as shown in Figure 4. Each area contains a group of QBB nodes that are assigned unique

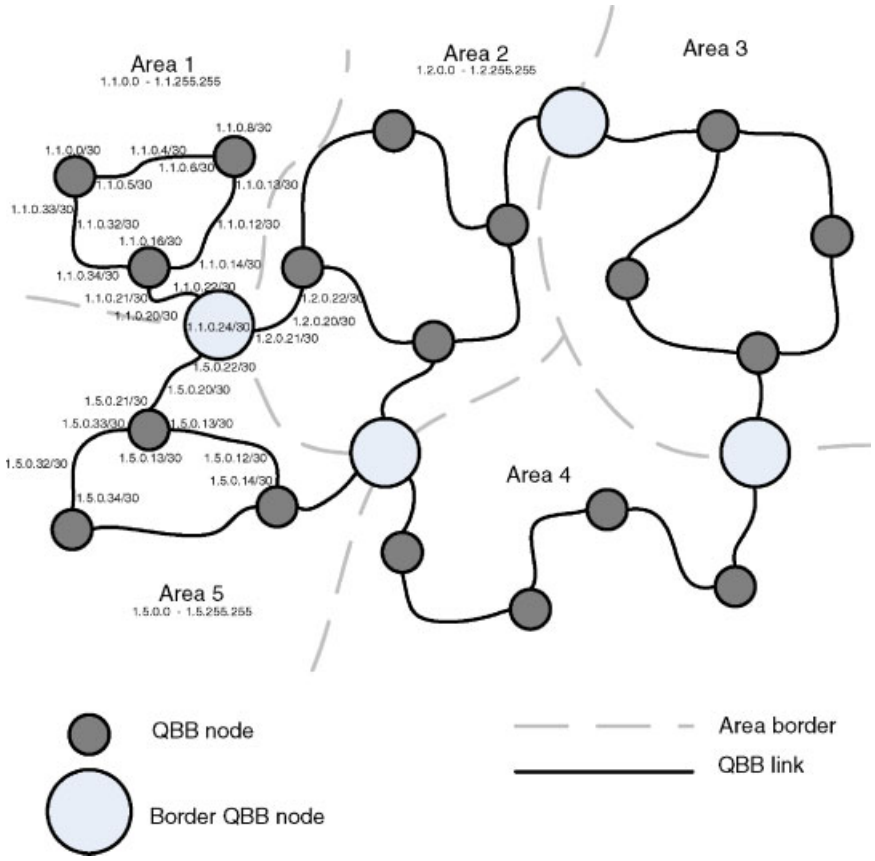

Fig. 4. Geographical division of the QKD network into several routing areas.

DOI: 10.1002/sec

IP addresses from a certain range of the available address space. For instance, a range of addresses from 1.1.0.0 to 1.1.255.255 can be allocated to the QBB nodes and QBB interfaces inside Area 1 in Figure 4. This addressing scheme enables route aggregation to reduce the size of routing tables. QBB nodes inside each area only need to have few entries for all QBB nodes outside its own area. For example, if there are 100 QBB nodes inside Area 4, a QBB node belonging to Area 1 only needs to store a few entries for Area 4. Although the proposed division of the addressing scheme is different from the Internet, it is still compatible with the Internet addressing scheme. Thus, the existing routing protocols such as OSPF can be customised for the QKD network. The area border QBB nodes, that connect two adjacent areas, partially belong to both areas; therefore, each interface of an area border QBB node receives an IP address from one area. The node address of an area border QBB node can optionally be chosen from one of the connected areas.

We explain the addressing scheme inside an area by an example as shown in Figure 4. Let the address ranges 1.1.0.0–1.1.255.255, 1.2.0.0–1.2.255.255 and 1.5.0.0–1.5.255.255 specify the allocated address spaces to Area 1, 2 and 5, respectively. Each space roughly consists of 65 536 individual addresses that can be assigned to any QBB nodes and QBB interface inside the corresponding area. Note that each QBB link is considered as an individual network with two nodes connected to two adjacent QBB nodes. Thus, an address with 30 bits of network portion, i.e. x.x.x.x/30 can provide enough host address space (four host addresses; three if we do not use 00) for each QBB link. Although each QBB node only requires one address, for future expansion, we also assign an address space with four addresses to each QBB node.

Similar to IP networks, we suggest reservation of the address range starting with four leftmost bits 1110 for multicasting and 1111 for future uses. Node address 127.0.0.1 is reserved for loop-back. Also, for each area, x.x.0.0 denotes the area itself, and x.x.255.255 is the area broadcast address.

2.4. QBB Links

A QBB link connects two QBB nodes together; as shown in Figure 5, a QBB link consists of: (1) an arbitrary number of quantum channels for sending quantum signals; (2) a classical channel for signalling and forwarding of session keys. Deploying multiple QKD devices and thereafter multiple parallel quantum

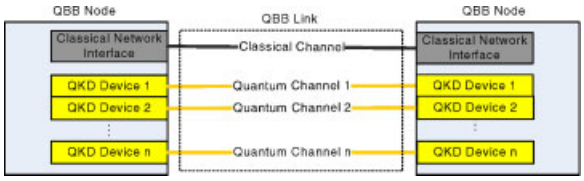

Fig. 5. QBB link.

channels can improve the effective key generation rate between adjacent QBB nodes. The importance of this parallel structure can be understood if we consider the typical dependency between the length of a QKD link and the effective key generation rate as shown in Figure 6. The vertical axis shows the effective key generation rate and the horizontal axis shows the distance between the QKD devices. The key generation rate at zero distance, R_0, is typically in the range from 100 Kbits/s to a few Mbits/s for the current technologies. The corresponding achievable rates are in a range 1–10 Kbits/s over 25 km [15].

It is important to note that quantum channels are subject to a variety of security threats by adversaries. However, proper QKD protocols [3] extract unconditionally secure keys by eliminating the compromised information. Two QKD devices will succeed to extract a secure key from the raw key materials, obtained by measuring the quantum signals, if the quantum bit error rate is below a certain threshold [16]. Otherwise, unconditional security cannot be guaranteed; thus, the raw key bits must be discarded. Obviously, this implies that an adversary can interrupt key stream between two nodes by frequent measurement of the quantum signals which will create too many errors. This is equivalent of physical interruption of wired link or jamming a wireless link, which can be considered as a type DoS attack. Indeed, one objective of the QKD network is to combat such threats by providing redundant paths

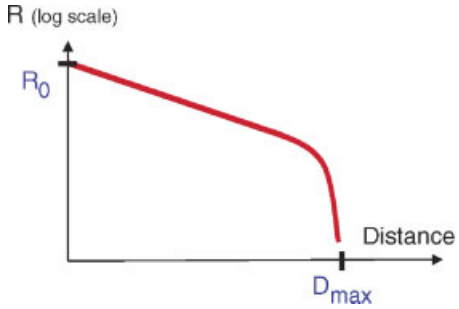

Fig. 6. Typical key generation rate versus distance for a single QKD link.

DOI: 10.1002/sec

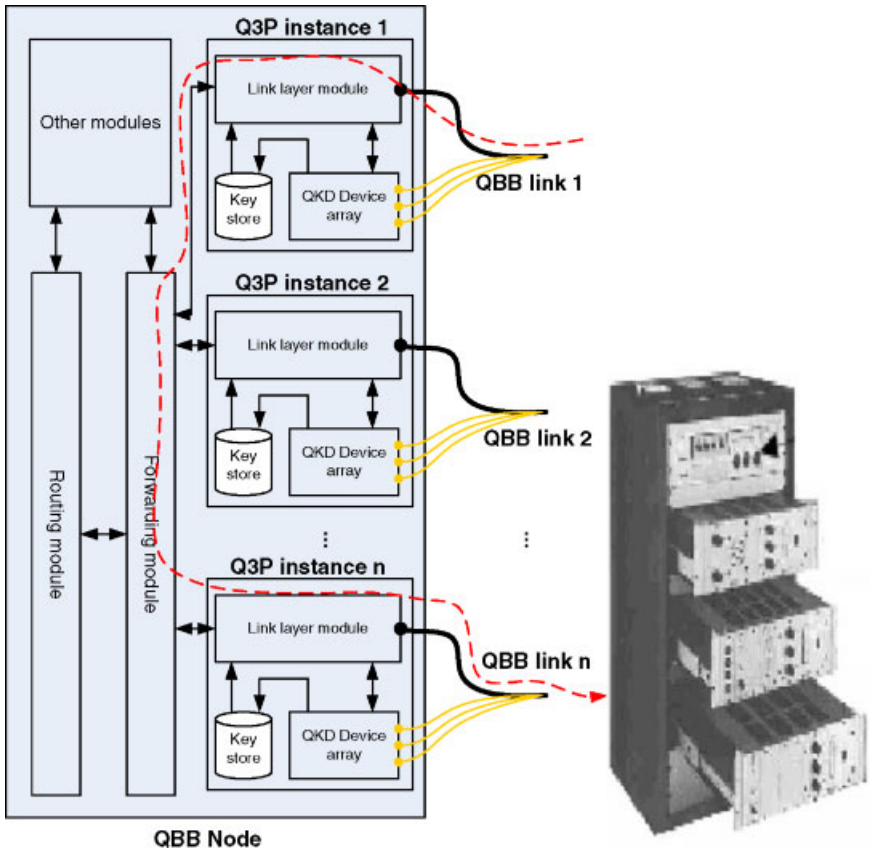

Fig. 7. A QBB node.

among QBB nodes and resilient routing protocols. That is, the network protocols should be able to detect such attacks on the individual links, and react by choosing alternative forwarding paths between QBB nodes.

The classical channel is a virtual channel rather than a physical channel. It could be either established over a public network through a TCP/IP socket or it could be a direct point-to-point link between two adjacent QBB nodes. The classical channels are not secure channels. Proper cryptographic algorithms, using the local keys, which are generated by the QKD devices, must be used if the communication has to be encrypted or authenticated.

2.5. QBB Node

QBB nodes: (1) create, store and manage local keys like QKD devices; (2) forwards session keys across the network like routers and (3) serve as access points for key clients applications. Physically, QBB nodes are special computers with multiple QBB interface ports, enabling connection to multiple QBB links. Each QBB node can also have multiple QKD devices to establish parallel QKD links for each individual QBB link. The QKD devices can operate over different types of quantum channels which employ different technologies. A QBB node also has an interface for a classical channel on each QBB port.

The logical structure and physical package of a QBB node are shown in Figure 7. The major components are: (1) multiple instances of quantum point-to-point protocol (Q3P) [17] modules, which serve as the link layer of the communication protocol stack; (2) a routing module which is responsible for collection and maintenance of the local routing information (e.g. routing tables) and (3) a forwarding module for maintaining paths and making forwarding decisions. The other modules such as the node management, local connection management, random session key generator, security monitor and etc. reside inside the box denoted by *other modules*. Since the focus of this paper is the networking aspect of the QKD network, we do not elaborate on the other modules as they have no or little direct network functionality. The readers may find more technical details of the other modules in our web site, www.secoqc.net.

DOI: 10.1002/sec

Q3P module manages the link level communications issues between two adjacent QBB node. A Q3P module includes a submodule for managing the communications over a classical channel, including functionalities for: (1) authentication; (2) encryption/decryption; (3) fragmentation/assembling; (4) flow control; (5) link level error control and (6) connection management between two adjacent QBB nodes. Q3P module also includes a submodule which acts as an array of device drivers for the QKD device array. This module collects local keys from the QKD devices and delivers them to a key store. Note that the QKD devices constantly generate local keys at their maximum key generation rate. The QKD device drivers also allow the QKD devices to share a single classical channel as it is needed for key generation process inside the QKD devices. Another important submodule of a Q3P module is the key store. The main purpose of the key store is to compensate low key generation rate of QKD devices by accumulating a big database of ready to use local keys which will be used for local cryptographic operations between two adjacent QBB nodes. Having the key stores, QBB nodes can tolerate fluctuations in key consumption by buffering the generated keys.

To serve as a router, a QBB node has modules for key forwarding and routing. The forwarding module examines the destination address of the incoming packets (i.e. session keys that are to be forwarded through the QKD network), performs necessary processing, and forwards them to the appropriate output buffer corresponding to a particular Q3P instance. An example of forwarding path inside a QBB node is illustrated by the dashed curve in Figure 7. A scheduling module prioritises departure of the packets from the buffer to the destination Q3P instance. The routing module creates and maintains the routing information which is used by the forwarding module. In the following, we provide further details of the routing and forwarding modules.

2.5.1. Routing module

The function of the routing module is to maintain local routing tables and help other QBB nodes to update their routing information. For the SeCoQC QKD network, we do not intend to invent the wheel from scratch as there are plenty of well studied and tested routing protocols. On the other hand, there are non-trivial differences among the conventional networks and the QKD network, in terms of capacity and topology, that requires a careful customisation of the existing protocols. We customise OSPF-v2 [18] for the SeCoQC

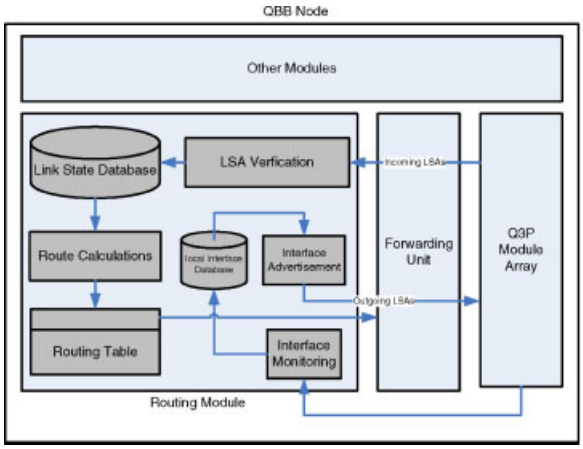

Fig. 8. The routing module of QBB nodes.

QKD network as specified in the following. Since the objective of the SeCoQC project is to build a network with proper quality of service (QoS), in terms of guaranteed key delivery rate to the end applications, choosing OSPF-v2 might appear contradictory as it does not support QoS routing. However, this option will speed up the development process by using plenty of off-the-shelf components. Finding a proper solutions for QoS provisioning is currently under investigation by the network subproject team. We are building a comprehensive simulation environment to study the possibility of implementing different QoS policies.

Figure 8 shows the logical structure of the routing module, the major components, the interactions among them, and the other components of a QBB node. We briefly explain the function of each component in Appendices A–C.

2.5.2. Route computations

QBB nodes implement Dijkstra algorithm to compute a shortest path tree data structure from which a routing table can be extracted. Unlike the link-state database, which is identical for an entire area, each QBB node has a unique shortest path tree and routing table. For example, Figure 9 illustrates the shortest path tree for QBB1 in Figure 18. The corresponding routing table is also shown in Table I. The type of destination is either a network, denoted by 'N', or a QBB node, denoted by 'R'. The destination nodes inside an area are denoted by 'Intra-area'; otherwise, they are denoted by 'Inter-area'. An asterisk sign indicates a non-applicable specification.

Unlike the standard OSPF which only computes the shortest path towards each destination, for the QKD

DOI: 10.1002/sec

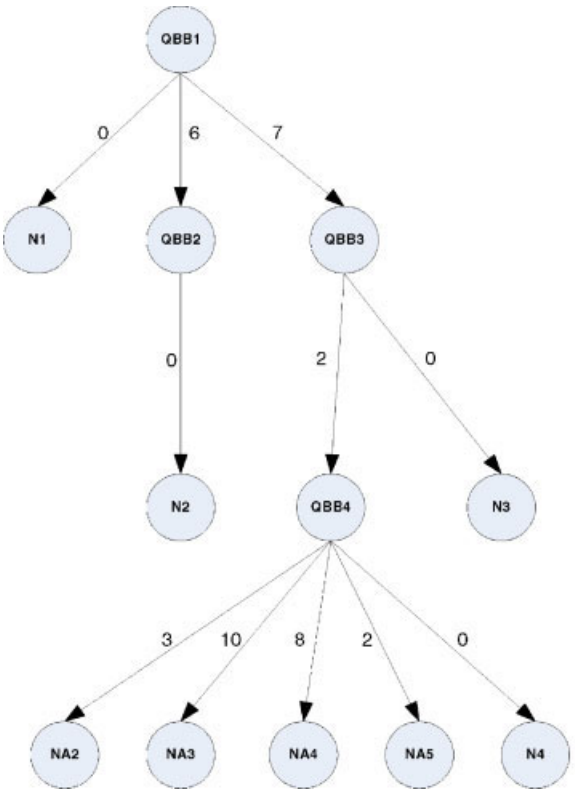

Fig. 9. The shortest path tree for the QBB1 in Figure 18.

network, it is desirable to have information of multiple paths to each destination in order to implement some degree of load balancing with minor modifications to the standard IP forwarding process. A QBB node computes as many routing tables as the number of its interfaces. For instance, if a node has three outgoing interfaces, three separate routing table are computed. These tables can be computed by using the existing functions of the standard OSPF. In order to compute the cost of reaching a certain destination from a particular outgoing interface, we need to set the costs of reaching that destination from the other outgoing interfaces equal to infinity and run the standard cost computation procedure.

Table I. Routing table for the shortest path tree in Figure 9.

Type	Dest-ID	Mask	Area	Path type	Cost	Next hop
N	N1	30	1	Intra-area	0	*
N	N2	30	1	Intra-area	6	QBB2
N	N3	30	1	Intra-area	7	QBB3
N	N4	30	1	Intra-area	9	QBB3
R	QBB2	*	1	Intra-area	6	*
R	QBB3	*	1	Intra-area	7	*
R	QBB4	*	1	Intra-area	9	QBB3
NA	NA2	16	2	Inter-area	12	QBB3
NA	NA3	16	3	Inter-area	19	QBB3
NA	NA4	16	4	Inter-area	16	QBB3
NA	NA5	16	5	Inter-area	11	QBB3

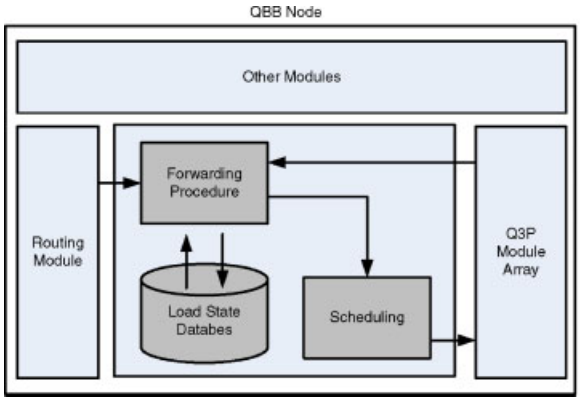

Fig. 10. Forwarding module.

2.5.3. Forwarding module

The structure of the forwarding module of the QBB node is shown in Figure 10. The forwarding module receives the incoming packets from the Q3P modules; then, the information of routing tables and the estimated interface load information are used to decide to the outgoing interface of each packet based the the destination address in the header of the packet. Load state database stores the estimated load information of the outgoing interfaces. Once the outgoing Q3P interface is determined, the packets are queued for transmission inside the scheduling unit. The scheduling implements a FIFO queue for each outgoing QBB link. However, as the QKD network services will evolve to provide QoS guarantee, a more sophisticated scheduling schemes will be required in the future.

In the intermediate QBB nodes, the forwarding process is very similar to that of the ingress QBB nodes. An intermediate QBB node calculates the header checksum and compares it to the value included in the packet header. If the checksums have different values, the packet will be discarded. If the destination address also corresponds to an address assigned to the network address associated with the QBB node itself, it will be delivered to the proper local process based on the host address. Otherwise, the packet will be forwarded to a proper outgoing interface. Before forwarding, the value of the time to live (TTL) field is decremented by 1. If the value of the TTL field is less than 1, an *Internet control message protocol* (ICMP) *Time Exceeded-TTL Exceeded in Transit* message must be sent to the sender,

DOI: 10.1002/sec

and the packet will be discarded. If the value of the TTL field is greater than 0, the checksum value is recalculated and updated. Then, the forwarding process looks for next hop address of the packet according to the destination address and the forwarding policy of the QBB node. If the *Maximum Transmission Unit* of the outgoing interface is different, fragmentation of the packet might be necessary. Otherwise, the packet will be passed to the corresponding Q3P module.

If the destination address of a packet belongs to the local network address associated with the QBB node, it will be delivered to the specified local address. In this case, the forwarding process calculates the header checksum and compares the calculated value with the value included in the header of the packet. If the checksums have different values, the packet will be discarded. Then, the host address portion will be examined; if the host address does not exist on the local network, an ICMP *Destination Unreachable–Protocol Unreachable* message is sent back to the sender QBB node, and the packet will be discarded.

In the following, we specify the procedure for computing the current set of next hops for a destination address. This decision must be made by both the ingress and intermediate QBB nodes. In the standard OSPF, the forwarding decision is made based on the destination address and the shortest path information of the routing tables. This may cause poor distribution of traffic load in the QKD network, resulting in congestion in some nodes, while there are under-utilised links. Since the capacity of the QKD network is relatively low, this is particularly an important drawback of using a simple routing scheme based on the shortest path to a destination.

We propose a simple local load balancing policy which splits the outgoing traffics among multiple paths. The proposed scheme is compatible with the destination-based forwarding and connectionless semantic of the network layer. To enable local load balancing, standard shortest path route calculation of the OSPF is used to create an extended routing table. Furthermore, the dynamic status of active interfaces of a QBB nodes are monitored to collect interface load information. As we mentioned in Subsubsection 2.5.2, the routing algorithm computes as many routing tables as the number of outgoing links. This *Extended Routing Table Computing Module* summarises multiple routing tables into a single *Extended Routing Table*. The structure of the *Extended Routing Table* is similar to that of a standard routing table. However, in the extended routing table there are as many entries for each destination as the number of outgoing links of a QBB node. These entries must be stored in an increasing order in terms of path costs. To track the approximated intensity of traffic on each outgoing link, the forwarding module also maintains a small database, namely *Load State Database*. Each entry of this database specifies the current approximated load on an individual outgoing link. The approximated load of outgoing link i at discrete time t, denoted by $L_i(t)$, is calculated by a low-pass filter as specified by the following equation:

$$L_i(t) = \left(1 - \frac{1}{w}\right) L_i(t-1) + \frac{1}{w} l_i(t) \qquad (1)$$

where w is the constant of the filter and $l_i(t)$ is the instantaneous load of outgoing link i. The instantaneous load is given by

$$l_i(t) = \frac{D_i(t, t-1)}{T(t, t-1)} \qquad (2)$$

where $D_i(t, t-1)$ is the number of transmitted bits between time $t-1$ and t and $T(t, t-1)$ is the time span between discrete times t and $(t-1)$.

Upon reception of a packet, the forwarding procedure examines the destination address and computes a set of next hops from the extended routing table. Starting from the cheapest next hop, the forwarding process fetches the approximated load of the corresponding next hop. If the load is below a critical level, $L^{(cr)}$, for the tagged outgoing link, the packet will be delivered to the appropriate Q3P module. Otherwise, the forwarding procedure will examine the next cheapest path in the list. If all the outgoing links are overloaded, the packet may be dropped or sent over a default outgoing link. Note that the algorithm chooses an alternative path if only the link is very close to congestion. This will minimise possibility of routing loops.

3. QKD Protocol Stack

We adopt a layered architecture for the communications protocol stack of the QKD network as shown in Figure 11. The protocol stack consists of four layers: (1) QKD link layer (QKDLL); (2) QKD network layer (QKDNL); (3) QKD transport layer (QKDTL) and (4) QKD application layer (QKDAL).

3.1. QKDLL

The QKDLL is an interface between the hardware of a QBB link and the upper layers. Since a QBB link is

DOI: 10.1002/sec

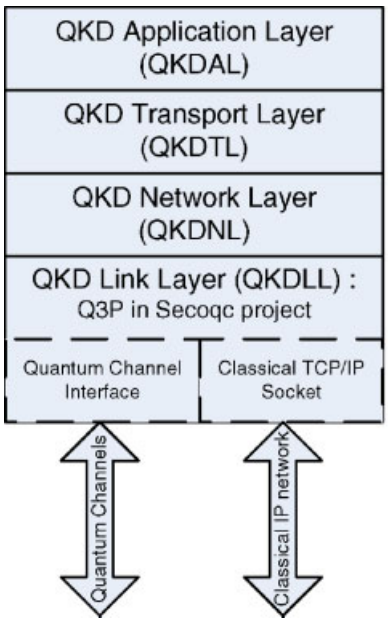

Fig. 11. The communications protocol stack of the QKD network.

composed of multiple quantum channels and a classical channel, QKDLL has two subsections for interfacing the classical and quantum channels. QKDLL hides all the technical details and technological heterogeneity of QKD devices and classical channel to provide a uniform access points for the upper layers. For the current implementation of the SeCoQC project, we consider only point-to-point QBB links. Thus, QKDLL is also referred to as the *Quantum Point-to-Point Protocol*(Q3P) [18]. Excluding the quantum channels, the QKD network is an overlay network over standard classical networks. The classical channel is a virtual channel, i.e. it might be established through a classical IP network. Hence, the QKDLL may implement a classical TCP/IP socket as an interface for the classical channel. The packet structure of the

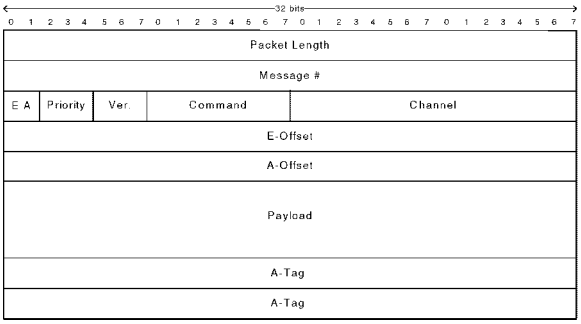

Fig. 12. QKDLL packet [17].

Table II. Description of the fields of a QKDLL packet.

Field	Description
Length	The length of the packet in bytes
Message number	Packet number
E	Packet must be encrypted if this flag is set
A	Packet must be authenticated if this flag is set
Priority	Packet priority
Ver.	Protocol version
Command	Command for control packets
Channel	The channel number (for further implementations)
E-offset	Encryption key offset
A-offset	Authentication key offset
Payload	Data payload
A-tag	Authentication tag

QKDLL is show in Figure 12. The descriptions of the fields are given in Table II.

3.2. QKDNL

The QKDNL enables forwarding of session keys throughout the QKD network. This is done by a proper packaging of the information inside network layer packets, which can be processed by the intermediate QBB nodes that serve as routers of the QKD network. Note that the forwarding process does not directly involve with the implementation of the cryptographic algorithms; instead, it sends the necessary commands to the QKDLL layer. Network services of the QKD network are connectionless; connection establishment/maintenance and reliable delivery are insured by the transport layer.

The structure of a QKD network packet is shown in Figure 13. We have adopted a similar packet structure to the IP protocol. The payload section contains a QKDTL packet or a routing protocol packet. The header is a 20-byte section with several fields as explained in the Appendix D.

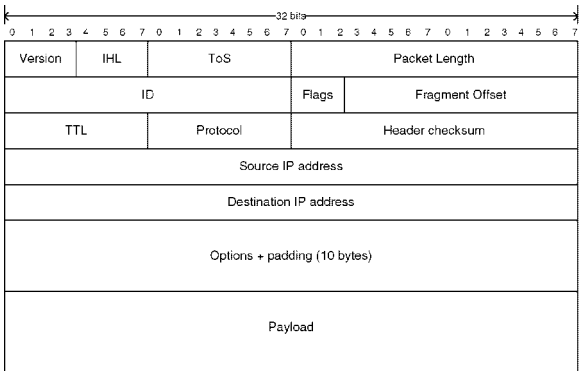

Fig. 13. QKD network layer packet.

DOI: 10.1002/sec

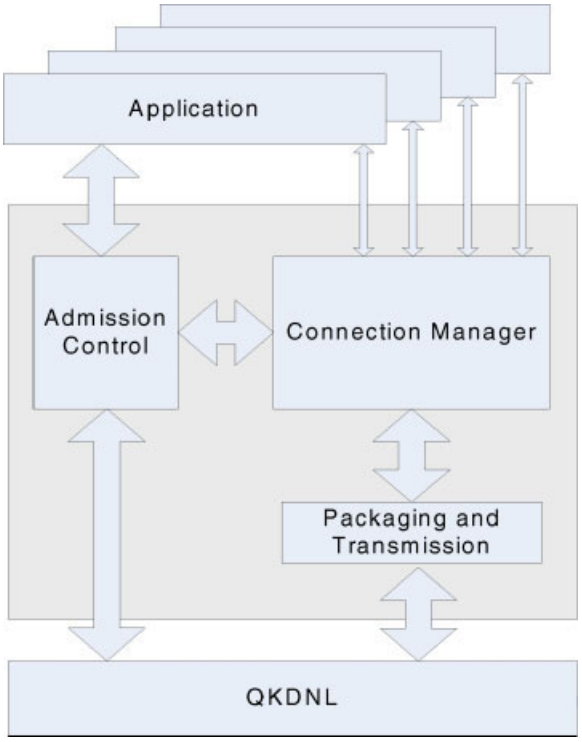

Fig. 14. The logical components of the QKDTL.

3.3. QKDTL

The QKDTL complements the network services and provides a uniform interface point for the client applications. In particular, the QKDTL supports:

(1) Reliable delivery through acknowledgement and retransmission.
(2) Congestion control to sustain the stability of the QKD network.
(3) Connection management to synchronise key consumption by the end applications.
(4) Multiplexing to share network services among multiple connections.

The logical components of the QKDTL and their internal and external interactions are shown in Figure 14

In addition to the aforementioned services, the QKDTL provides the interface layer for the application programs who might use the QKD network. A typical scenario is illustrated in Figure 15. A client application program that is located inside a private network, where there exist a QBB or QAN node, intends to establish a secure connection over a public network with a server application. Note that the server application is also located inside another private network who is connected to another QBB or QAN

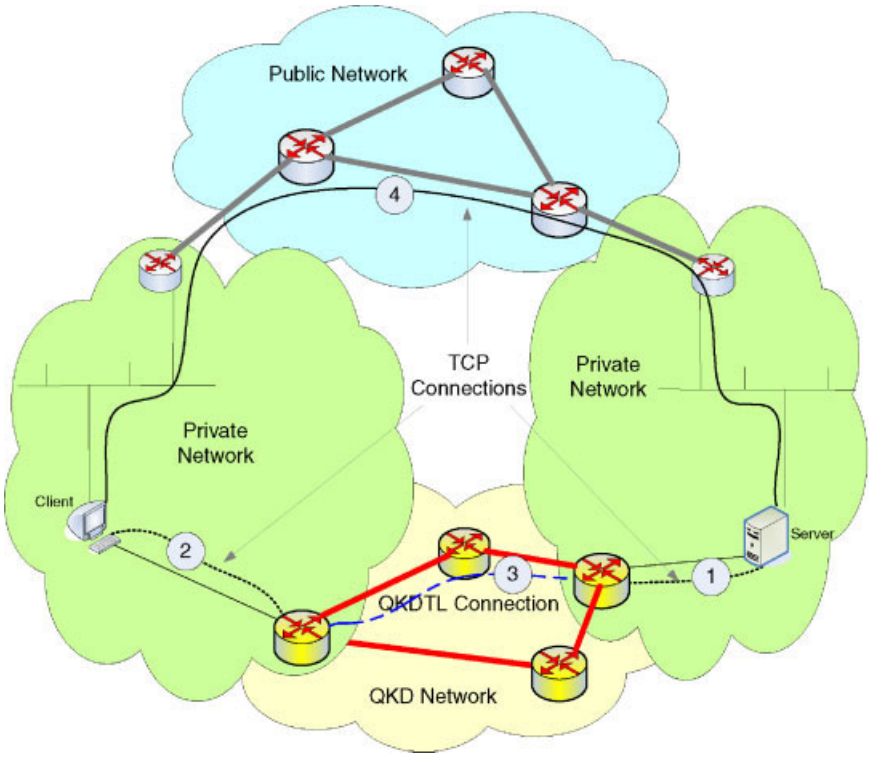

Fig. 15. Initialisation of a QKDTL connection in three phases.

DOI: 10.1002/sec

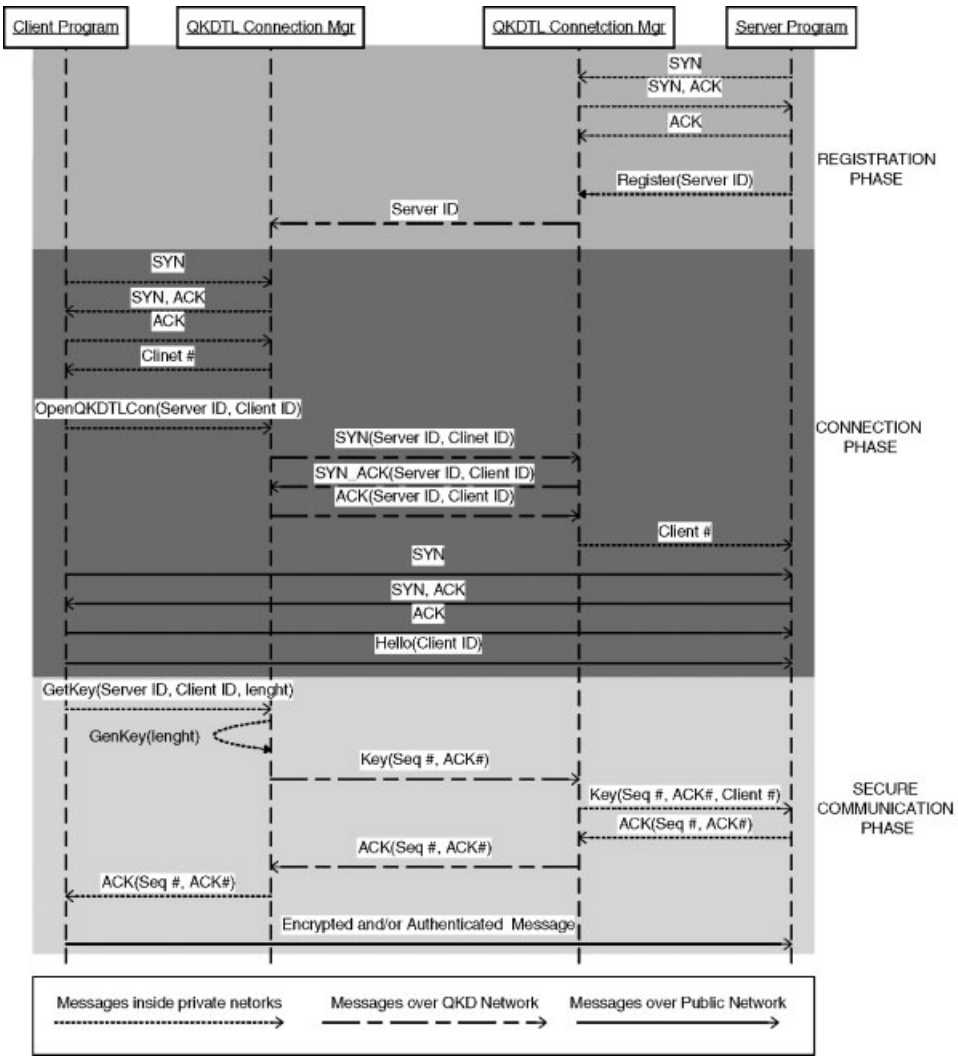

Fig. 16. QKDTL connection setup: sequence diagram.

node. As a precondition, in registration phase, the server application must establish a conventional TCP connection to the corresponding QBB node; then, it must register itself with the QKD network. The registration information is then propagated across the entire QKD network. In the connection phase, the client application connects to its corresponding QBB node through a TCP connection and sends a request for opening a QKDTL connection to an already registered server application. Next, the client side QBB node establishes a QKDTL connection with the server side QBB node. Upon the successful establishment of the QKDTL connection, the client application establishes a conventional TCP connection with the server application over the public network. In secure communication phase, the client and the server applications may request for session keys of certain lengths. The QBB node in the requesting application side generates a random number of the requested length and sends it through the QKDTL connection to the other side. Upon the successful delivery of the key, the QBB node delivers a copy of the key to the requesting application. Finally, the client and the server applications can use the key to perform desired cryptographic algorithms for secure communications over the public network. The detailed sequence diagram of the entire process is shown in Figure 16. The relevant modules can be either implemented inside the application source code or embedded in the operating systems. The structure of the QKDTL packet is shown in Figure 17.

DOI: 10.1002/sec

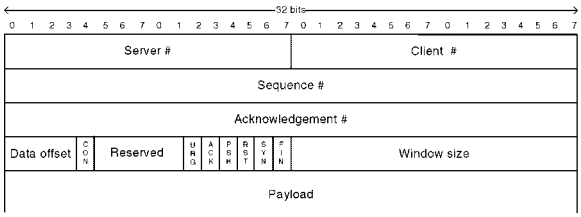

Fig. 17. QKDTL packet.

4. Summary and Conclusions

In this paper, we specified the architecture and the major components of the network layer of the SeCoQC QKD network. The proposed architecture and the corresponding protocol stack are inspired by the Internet model. Our design however is customised to satisfy the relevant constraints of the QKD technology and to enable efficient consumption of key materials. We have opted for an incremental approach to allow gradual increments of system complexity and new functionalities in order to facilitate the development process. We are currently conducting simulation studies in order to evaluate the performance of the network and tune the relevant parts of our design. Our current routing protocol is a customised version of the standard OSPF-v2. The future works will however consider feasibility of virtual circuit based routing protocols such as MPLS in order to enable QoS guarantee.

Appendix A: Link-State Database

Link-state database defines a directed graph which represents the structure of the QKD network, i.e. the QBB nodes and the QBB links among them, within a particular area. It also contains a summarised view of the entire QKD network. Except for the area border QBB nodes, the contents of link-state databases of all nodes inside an area (except for the area border nodes) converge in steady state. This is achieved through the occasional exchange of link-state advertisement (LSA) messages. Area border QBB nodes that participate in multiple areas maintain a separate link-state database for each area.

We illustrate the structure of a link-state database for an individual area using the example QKD network in Figure 18. We focus only on Area 1; however, the same concepts apply for the other areas. In the figure, there

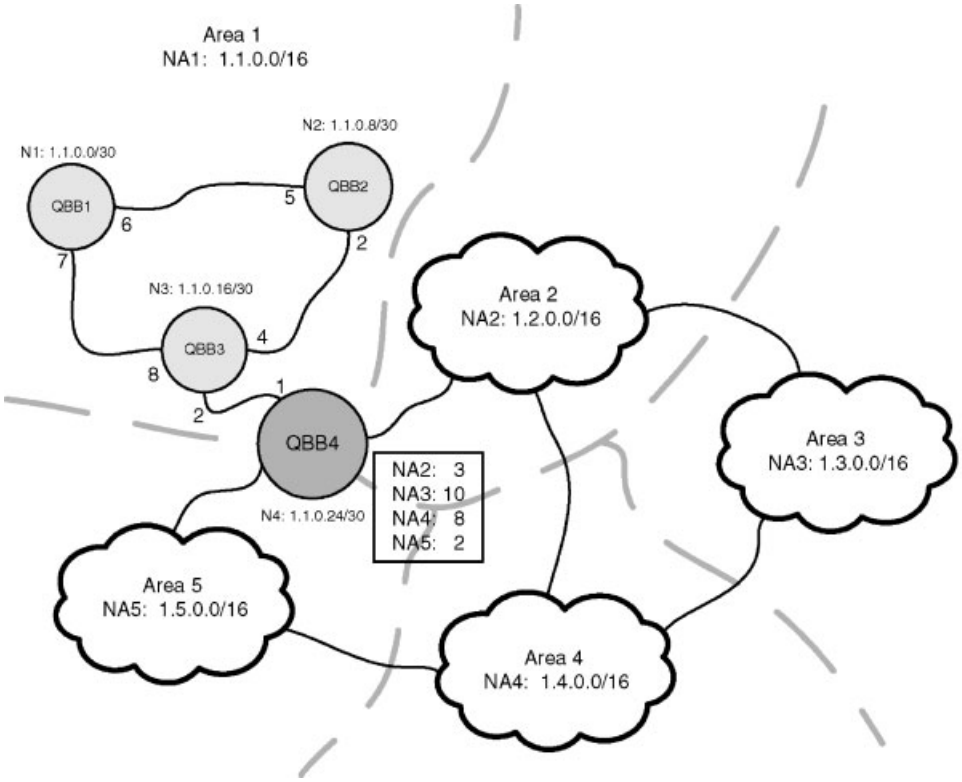

Fig. 18. An example topology of the QKD network to explain the structure of link-state database.

DOI: 10.1002/sec

Table III. Link-state database of the example in Figure 18.

	FROM			
TO	QBB1	QBB2	QBB3	QBB4
QBB1		5	8	
QBB2	6		4	
QBB3	7	2		1
QBB4			2	
N1	0			
N2		0		
N3			0	
N4				0
NA2				3
NA3				10
NA4				8
NA5				2

are four local routers in Area 1; note that QBB4 is also an area border node. An area border node summarises its knowledge of other areas and advertises it to all the routers inside its own area. There are also four networks, denoted by N1–N4, in Area 1. The networks, which do not belong to Area 1, are denoted by NA1–NA5, belonging to Area 1–Area 5, respectively. The numbers on the interface ports of QBB nodes represent the cost of sending a network key packet of certain size through the outgoing link to an adjacent QBB node. For example, the cost of sending a packet from QBB1 to QBB2 is represented by scalar number 6. Note that the cost on the opposite direction might be different as shown in the Figure 18. For example, the cost of sending a packet from QBB2 to QBB1 is denoted by scalar number 5. Link costs can be assigned by an administrator or a scheme inside a QBB node can compute the cost according to a predefined set of rules. Cost metrics can also be adjusted dynamically for load balancing purposes (such approaches must be considered carefully as they might cause heavy overhead and instability as well). The cost of reaching the networks outside the area is summarised by the area border routers. For example, in Figure 18, QBB4 has summarised the cost of reaching the nodes inside NA2–NA5 in a table which is shown beside the node. In our example, the general cost of reaching a node in Area 4 is 8.

The link-state database for Area 1 in the example of Figure 18 is shown in Table III. The table specifies the existence and cost of links among QBB nodes, networks and other area networks. Note that the area border QBB node, QBB4, has connections to other areas.

Appendix B: Local Interface Database

Each QBB node monitors its outgoing links to the adjacent nodes and adjacent networks through periodic handshakes, and collect the relevant information. This information is used by the routing protocol for the route advertisement process. The local interface database contains the same fields as the standard OSPF to enable using standard codes. However, some fields may not be used or used for slightly different purposes. The list of relevant fields in the local-interface database is given in the following:

- *Type:* The OSPF interface type is either point-to-point, broadcast, NBMA, *Point-to-MultiPoint* or virtual link. In SeCoQC project we will have only point-to-point links.
- **State:** Notifies the current state of the link.
- **IP interface address:** The IP address associated with the interface.
- *IP interface mask:* Also referred to as the subnet mask, this indicates the portion of the IP interface address that identifies the attached network. Masking the IP interface address with the IP interface mask yields the IP network number of the attached network.
- *Area ID:* The ID of the area to which the attached network belongs.
- *HelloInterval:* The length of time, in seconds, between the Hello packets that a QBB node sends through the interface.
- *RouterDeadInterval:* The number of seconds before the neighbour QBB nodes considers the link is dead if no Hello packets are received.
- *InfTransDelay:* The estimated transmission and propagation delay in seconds that takes to transmit a link-state update packet over this interface. This field is used to compute the age of link-state advertisement packets after they are received.
- *Hello Timer:* An interval timer that causes the interface to send a Hello packet. This timer fires every *HelloInterval* second.
- *Interface output cost(s):* The cost of sending a network key packet of certain length through the interface, expressed in the link state metric. This is advertised as the link cost for this interface in the link-state advertisement messages.
- *RxmtInterval:* The number of seconds between two consecutive retransmissions of link state advertisement messages.
- *AuType:* Indicates the type of authentication algorithm that must be applied by the Q3P module.

DOI: 10.1002/sec

Appendix C: Routing Table

Each QBB processes the raw information of link-state database to compute routing tables which can be conveniently used by the forwarding module. Unlike the standard OSPF, for the QKD network, each node maintains multiple routing tables for load balancing purposes. The data structure of the routing tables in the QKD network is identical to the structure of routing tables in standard OSPF. The important columns of a routing table are briefly explained in the following (note that only the relevant data fields are listed).

- *Destination ID:* The IP address of the destination node.
- *Address Mask:* This field is used to compute the network address from the destination IP address.
- *Area:* Specifies the area ID of the destination node.
- *Path-type:* A path to a destination node can be either intra-area or inter-area. For an intra-area path, the destination node resides within the same area of the tagged node, while for the an inter-area path, the destination node is outside the area of the tagged node.
- *Cost:* The total cost of forwarding a single packet to a destination through a particular path.
- *Next hop:* The IP address of the next node in the path and the corresponding local interface.
- *Advertising router:* For inter-area paths, this field indicates the ID of the node who has advertised the summary LSA leading to the destination node.

Appendix D: Description of the QKDNL Packet Header

This appendix describes the header of the network layer packets:

- *Version:* Four bits indicating the version of network protocol.
- *Internet Header Length(IHL):* Four bits indicating the length of the internet header in 32-bit words, and thus points to the beginning of the data. Note that the minimum value for a correct header is 5.
- *Type of Service (ToS):* Eight bits defining the *quality of service* (QoS) requirements of a QKD network packet (will not be used in the current implementation of the QKD network).
- *Packet Length:* A 16-bit field indicating the total length of the QKD network packet. This field allows the length of a packet to be up to 65 535 bytes.
- *ID, Flags and Fragment offset:* Will not be used in our implementation as the necessity and security risks of packet fragmentation on the QKD network has not been discussed yet. We keep these fields to preserve the compatibility with standard IP packet.
- *Time to Live (TTL):* This field specifies the maximum time, in seconds, that a packet can live on the network. Each forwarding node decrements this field accordingly. The minimum decrement is 1 second. After TTL field reaches zero the packet should be dropped.
- *Protocol:* This field specifies the upper layer protocol type. For example, OSPF protocol type is 89.
- *Header Checksum:* This field is used for validation of the header of the QKD network packet only. Note that each forwarding node must compute and update this field separately.
- *Source Address and Destination Address:* specifies the origin and the destination of the packet.
- *Options:* A variable length section for future and optional purposes.

Acknowledgements

We would like to thank the members of the NI and NET SeCoQC subproject, in particular Stefano De Crescenzo, Marco Nicoletti, Oliver Maurhart, and Momtchil Peev for numerous helpful discussions on the design of the network architecture. We also acknowledge the financial support from the European Commission through the IST-SeCoQC Integrated Project.

References

1. Diffie W, Hellman ME. New directions in cryptography. *IEEE Transactions on Information Theory* 1976; **22**: 644–654.
2. Merkle RC. Secure communication over an insecure channel. *Communications of the ACM* 1978; **21**: 294–299.
3. Bennet CH, Brassard G. Quantum cryptography: public key distribution and coin tossing. In *Proceedings of IEEE International Conference on Computers, Systems, and Signal Processing*, Bangalore, India, 1984; 175–179.
4. Wiesner S. Conjugate Coding. *ACM SIGACT News* 1983; **15**(1): 78–88.
5. Mayers D. Unconditional security in quantum cryptography. *Journal of the ACM* 2001; **48**(3): 351–406.
6. Shor PW, Preskill J. Simple proof of security of the BB84 quantum key distribution protocol. *Physical Review Letters* 2000; **85**: 441–444.
7. Peres A. How to differentiate between non-orthogonal states. *Physical Letters* 1988; **128**: 19.
8. Sergienko A. The Darpa Quantum Network. In *Quantum Communications and Cryptography*. CRC Press: Boca Raton, FL, 2006; 83–101.

DOI: 10.1002/sec

9. Townsend PD, Phoenix SJD, Blow KJ, Barnett SM. Quantum cryptography for multi-user passive optical networks. *Electronics Letters* 1994; **30**: 1875–1877.
10. Zurek W. Decoherence and the transition from quantum to classical. *Physics Today* 1991; **44**(10): 36–44.
11. Biham E, Huttner B, Mor T. Quantum cryptographic network based on quantum memories. *Physical Review A* 1996; **54**(4): 2651–2658.
12. Dür W, Briegel HJ, Cirac JI, Zoller P. Quantum repeaters based on entanglement purification. *Physical Review A* 1999; **59**: 169–181.
13. Collins D, Gisin N, Riedmatten H. Quantum relays for long distance quantum cryptography, 2003. http://arxiv.org/abs/quant-ph/0311101v1 [23 January 2008]
14. Ekert AK. Quantum Cryptography based on Bell's Theorem. *Physical Review Letters* 1991; **67**: 661–663.
15. Bechmann-Pasquinucci H, Cerf N, Dusek M, Lütkenhaus N, Scarani V, Peev M. Report on a QIT-perspective comparison of the different platforms with respect to the evaluation criteria set in phase I of SECOQC, September 2005; *Secoqc deliverable D-QIT-02*.
16. Dusek M, Lutkenhaus N, Hendrych M. Quantum cryptography. *Elsevier Journal of Progress in Optics* 2006; **49**: 381–454.
17. Maurhart O. Q3P a proposal. *Secoqc deliverable*, 2006. http://www.secoqc.net [23 January 2008]
18. Moy J. RFC 2328, OSPF version 2, *Network Working Group, Ascend Commnications, Inc.*, April 1998.
19. Alléaume R, Bouda J, Branciard C, Debuisschert T, Dianati M, Gisin N, Godfrey M, Grangier P, Länger T, Leverrier A, Lütkenhaus N, Painchault P, Peev M, Poppe A, Pornin T, Rarity J, Renner R, Ribordy G, Riguidel M, Salvail L, Shields A, Weinfurter H, Zeilinger A. Secoqc white paper on quantum key distribution and cryptography, 2007; http://www.secoqc.net/downloads/secoqc_crypto_wp.pdf [23 January 2008]

DOI: 10.1002/sec