Software Development for B92 Quantum Key Distribution Communication Protocol

Shirantha Wijesekera
University of Canberra
Australia
d.wijesekera@student.canberra.edu.au

Dr. Sajal Palit
University of Canberra
Australia
Sajal.Palit@canberra.edu.au

Dr. Bala Balachandran
University of Canberra
Australia
Bala.Balachandran@canberra.edu.au

Abstract

In cryptography, Quantum Key Distribution (QKD) is a highly secure transmission method used to send secret keys from sender to receiver. This paper discusses about the software implementation of the first two phases of QKD: (1) Raw Key Extraction and (2) Error Estimation. This paper also discusses about future work proposed to enhance these two phases. A TCP/IP based Client-Server concept has been used to implement the communication between two users in C++ language.
Keywords: *Quantum Key Distribution (QKD), TCP/IP, B92, BB84, QBER, IPsec, Socket Programming*

1. Introduction

Quantum Key Distribution systems transmit the secrete key, which are derived from random numbers, one photon (one bit) at a time in a polarized state. If intercepted by an eavesdropper or due to other atmospheric interferences etc, this state will change, and an error will be detected at the receiving side [1].

There are several QKD protocols available. Most widely used is being the BB84 [3]. B92 (Charles Bennett), a slight variation of BB84, is another well known QKD protocol [4].

BB84 coding scheme, invented by Charles Bennett and Gilles Brassard, is the first quantum cryptography communication protocol. This coding system uses 4 non-orthogonal polarization states identified as *horizontal*, *Vertical*, *45°* and *135°*. This protocol operates with transmitting party (Alice) sending polarized qubits to the receiving party (Bob) via the Quantum channel. Once the quantum transmission finishes, Bob publicly communicates to Alice which measurements operators he used for each of the received bit. Alice then informs Bob which of his measurement operators choices were correct.

The B92 quantum coding scheme is similar to the BB84, but uses only 2 out of the 4 BB84 non-orthogonal states. It encodes classical bits in two non-orthogonal BB84 states. In addition to this, Bob simply sends the positions of the bases to retain, keeping the protocol simpler and faster to operate.

This B92 software has been developed in parallel with a hardware project in progress at the Monash University. Eventually this software is to be tested against the quantum transmissions performed at the Monash project.

2. Quantum Cryptography

Quantum cryptography uses the mechanics for secure communications to allow two users of a common communication channel to create a body of shared and secret key information. This key information, which generally takes the form of a random string of bits, can then be used as a conventional secret key for secure communication.

The Quantum key transmission happens in two stages (figure 1).

Stage 1: Quantum Channel (One way communication)

This transmission, could happen in either through Free space or Optical fiber. At present this implementation is being done at the Monash University.

Stage 2: Classical Channel (Two way communication)

This phase deals with recovering identical secrete keys at both ends.

During this stage Alice & Bob communicate over a Classical channel in 4 main phases:

1) Raw key extraction (Sifting)
2) Error Estimation
3) Reconciliation
4) Privacy Amplification

For this implementation, the internet is chosen as the Classical channel.

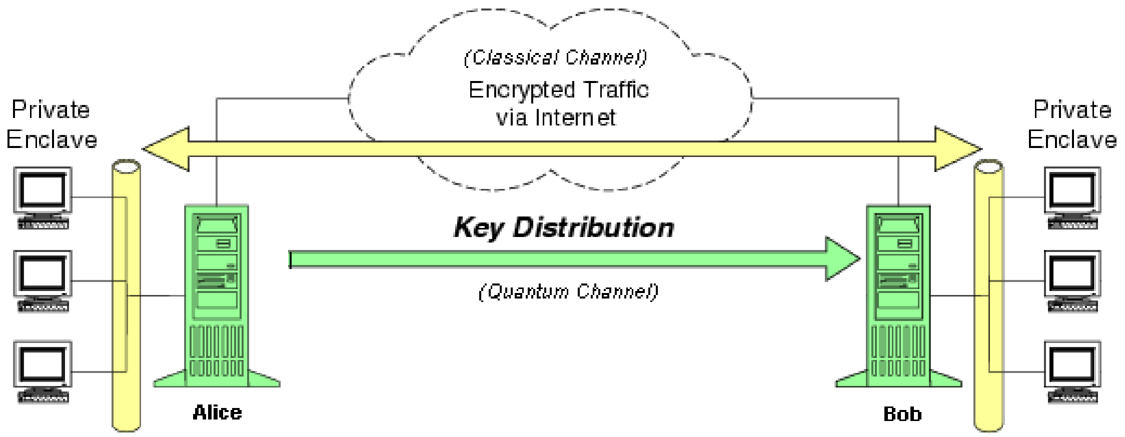

Figure 1. Simplified block diagram of a point-to-point QKD link in concept

Quantum Network

Quantum Key Distribution techniques are emerging as useful building blocks in highly secure networks. The quantum network marries a variety of QKD techniques to well established internet technology in order to build a secure key distribution system employed in conjunction with the public internet or, more likely, with private networks that employ the internet protocol suite [7]. At present there are large number of such private networks in widespread use around the world with customers desire secure and private communications. The merge of QKD technologies to these networks proves feasible and appealing in certain contexts.

Today, secure communication between cryptographic gateways or more indeed between individual computers on the internet, is provided by the well defined architecture of IPsec [6]. It specifies protocols, algorithms, databases and policies for secure communication and implemented by a set of cryptographic protocols for (1) securing packet flows and (2) Internet key exchange [8]. Therefore QKD technology fits well with the internet security architecture delivering guarantied secure internet traffic via quantum cryptography.

Figure 2 shows a multi-layer approach for the QKD protocol. Those layers outline the degree of freedom each layer exhibit.

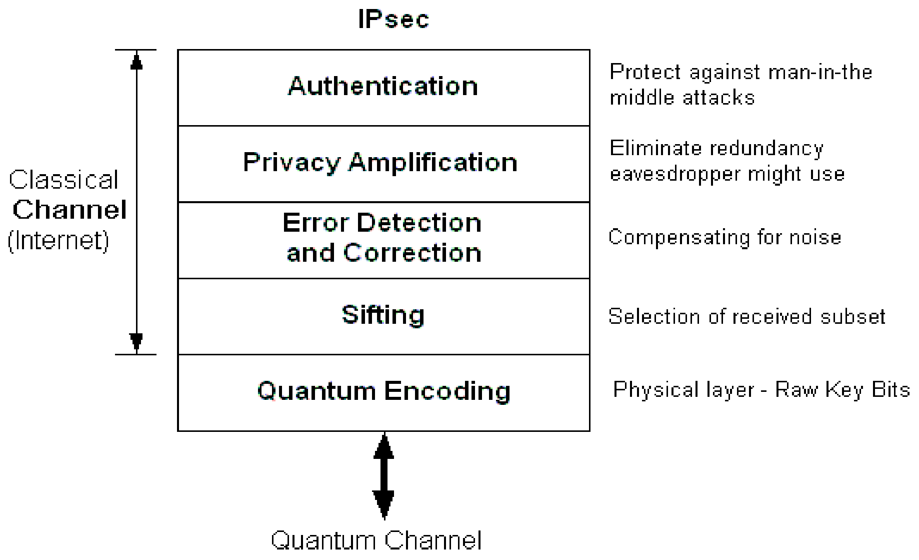
IPsec
Authentication
Protect against man-in-the middle attacks
Privacy Amplification
Eliminate redundancy eavesdropper might use
Error Detection and Correction
Compensating for noise
Sifting
Selection of received subset
Quantum Encoding
Physical layer - Raw Key Bits
Classical Channel (Internet)
Quantum Channel

Figure 2. Internal structure and functionality of KQD protocol suite

3. System Implementation

3.1. Architecture

TCP/IP based client-Server architecture is chosen to implement the communication platform of classical channel. This communication could happen over the internet with Alice acting as the server, while Bob acting as the Client. This set up has been developed and tested at the University of Canberra test labs.

Alice listens on the specified port while Bob initiates the socket based communication by sending connect request to the server side. Upon receiving the connect request, Alice accepts the connection, establishing the communication link with Bob.

3.2. Index Files

The software implementation depends on the key bits recorded at Alice and Bob's ends. These key bits are to be recorded in set of files, known as "Index Files". Since the original key transmitted by Alice in the Quantum Channel could contain many bits (gigabits), there will be multiple index files generated at either ends.

Those index files will act as the input to this software development project.

3.2.1. Index files at Alice's end. All the key bits that Alice transmits in the Quantum Channel are to be recorded into index files at her end. These files holds the original key that Alice transmitted to Bob.

Examples of the bits recorded in those index files:-

1,0,1,1,0,0,0,1,1,0,0,1,0,1,1,1,0,0,0,1,1,0

Where "," being the delimiter

3.2.2. Index files at Bob's end. During the Quantum transmission, Bob too records the key bits that he received from Alice in Index files. These bits will not be identical to what Alice has transmitted due to the random bases used by Bob's photon detector, eavesdropper attacks, channel noise, dark counts of the photon detector etc..

Therefore the index files recorded at Bob's end will comprise non-receptions. Non-receptions are the bit positions that Bob should have received, but not receive a bit.

Examples of the bits recorded at Bob's index files:-

1,1,,,0,0,,0,1,,1,0,0, ,0, ,1,,1,0,0, ,1,1,0

Where "," being the delimiter

With the use of delimiter to separate each key bit, it is easy to identify the of non-reception bits.

3.3. Program Structure and Protocol

Both Alice and Bob maintains a C++ class to hold individual parameter values of each index file. This class comprises of: Key bits, total number of bits, non-receipt bit positions etc.

At start up, Alice and Bob reads all the index files and populates the respective parameters in their data structures.

Figure 3 shows the protocol used between Alice and Bob.

The software has been developed in C++ language using UNIX socket programming. In order to establish the communication path, the Server (Alice) first *listen*s to a specified port. The Client (Bob) sends the *connect* message to the specified port in Alice's machine. Upon receiving the *connect* request, Alice sends *accept* call to Bob establishing the communication path between them. This link will be used during the whole communication that happens on the classical channel until the key is recovered.

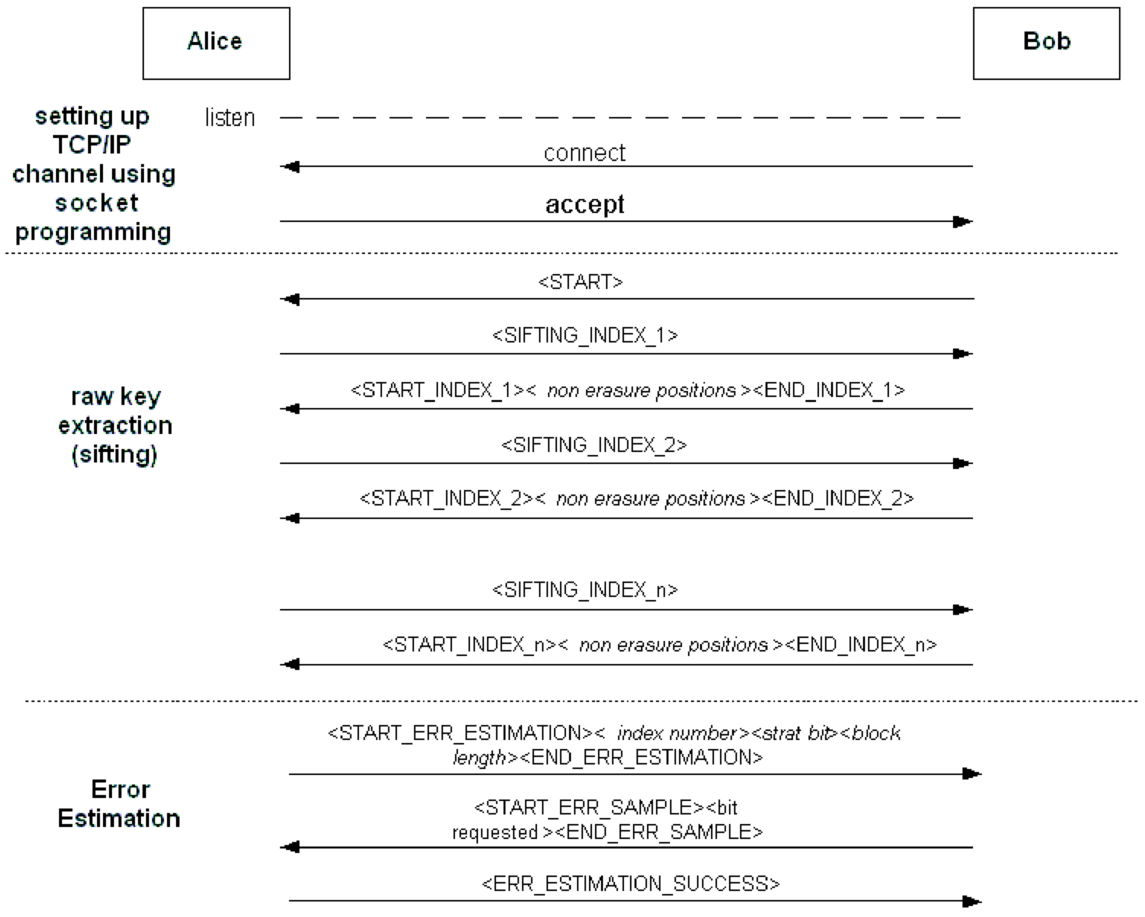
Alice
Bob
setting up TCP/IP channel using socket programming
listen
connect
accept
<START>
<SIFTING_INDEX_1>
<START_INDEX_1>< non erasure positions ><END_INDEX_1>
<SIFTING_INDEX_2>
<START_INDEX_2>< non erasure positions ><END_INDEX_2>
<SIFTING_INDEX_n>
<START_INDEX_n>< non erasure positions ><END_INDEX_n>
raw key extraction (sifting)
<START_ERR_ESTIMATION>< index number><strat bit><block length><END_ERR_ESTIMATION>
<START_ERR_SAMPLE><bit requested ><END_ERR_SAMPLE>
<ERR_ESTIMATION_SUCCESS>
Error Estimation

Figure 3. The protocol

3.3.1. Raw Key Extraction (Sifting). During this process, which happens at start up, Bob sends the non-erasure bit positions to Alice by running through the index file data loaded into the memory. Alice in turn, processes her index files and keeps only those corresponding bits.

At the end of this process, both Alice and Bob will have index files of identical lengths after removing all non-receipt bits. This process is called Raw Key Extraction and the keys recovered after this phase are known as *Alice's Raw Key* and *Bob's Raw Key*.

3.3.2. Error Estimation. This process starts with Alice requesting Bob to send a block of bits of length "L" from a particular index file.

This request has the following format:
<STRAT_ERR_ESTIMATION> <INDEX FILE NUMBER> <START BIT> <LENGTH> <END_ERR_ESTIMATION>
All the above values can be read as configurable parameters to the program.

Upon receiving this message, Bob sends the requested block of bits to Alice.

Alice calculates the Bit Error Rate (**e**) of the Quantum transmission.

where $\mathbf{e} = \dfrac{\text{Number of bits in error} \quad \text{X } 100}{\text{Total number of bits in the block}}$

Alice then compares with the maximum error rate allowed (e_{max}). This value is also known as Quantum Bit Error Rate (QBER).

- If $\mathbf{e} <= \mathbf{e}_{max}$ they accept the quantum transmission and proceeds to the next phase called Reconciliation. Both Alice and Bob removes those bits which are publicly revealed from their index file(s).
- If $\mathbf{e} > \mathbf{e}_{max}$ Alice sends ABORT message to Bob indicating the quantum transmission contains errors to a level where they cannot recover the key from the bits received. In this case, they seizes the session by terminating the program.

4. Conclusions

At present, the first two stages of B92 protocol has been implemented in C++ language on Linux platform. GNU has been used as the compiler. This set up has been successfully tested with multiple index files at the University of Canberra test lab.

Lot of performance improvements have been done to improve the quality of the software. Initially all index files were processed by writing to various intermediate temporary files. This caused a heavy overhead as the program consumes considerable amount of time during bit comparisons etc when doing file processing. To avoid this inefficiency, a STL list structure has been implemented to hold the index file data. Due to this modification, most of the computations and bit comparisons are done in-memory. This has resulted in improving the efficiency by about 60%.

Also some of the important values have been fed to the program as configurable parameters. With this set up, the program can be operated by setting different values to suit any requirements. One such parameter is the QBER, where this value is used to calculate the error rate of the quantum transmission. QBER of the quantum transmissions could be impacted by various issues (described earlier) causing it to vary per each transmission. Therefore by having the QBER as a configurable parameter, this software can be used to run even for simulation purposes by setting different values. Some of the other values that set as configurable parameters are: block size, starting position, index file that are used to calculate the error rate in the Error Estimation phase.

5. Future Work

As for the immediate work, the next phase (Reconciliation) will be implemented. This is the most critical phase of this software, since it deals with removing all errors introduced during the quantum transmission. This phase involves binary searching and parity comparisons etc.

The other most important future work planned being the modification of this software to operate on both B92 and BB84 modes. At present the software designed to cater B92 protocol only. Except for the first phase, both B92 and BB84 operates in almost identical fashion [5].

6. References

[1] C.H. Bennett et al., "Experimental Quantum Cryptography," *J. Cryptology,* vol. 5, no. 1, 1992, pp. 3–28.

[2] Charles H. Bennett "Quantum Cryptography: Uncertainty in the Service of Privacy" Science 257, 752-3 (1992)).

[3] C.H. Bennett and G. Brassard "Quantum Cryptography: Public Key Distribution and Coin Tossing", Proceedings of IEEE International Conference on Computers Systems and Signal Processing, Bangalore India, December 1984, pp 175-179.).

[4] C. H. Bennett, "Quantum cryptography using any two nonorthogonal states," Phys. Rev. Lett. 68, 3121-3124 (1992).

[5] Samuel J. Lomonaco, A Quick Glance at Quantum Cryptography (1998)

[6] Bassam Aoun, Mohamad Tarifi , Quantum Networks

[7] http://www.iop.org/EJ/article/1367-2630/4/1/346/nj2146.html
Building the Quantum Network, Chip Elliott, BBN Technologies, New Journal of Physics 4 (2002) 46.1-46.12

[8] Kent S and Atkinson R 1998 Security architecture for the internet protocol Preprint RFC .