P- ISSN 1991-8941 E-ISSN 2706-6703 Journal of University of Anbar for Pure Science (JUAPS) Open Access
2008,(2), (1) :01-11

EXTENSION OF SSL/TLS FOR QUANTUM CRYPTOGRAPHY

Sufyan T. Faraj

College of Computers, University of Anbar

ARTICLE INFO

Received: 01 / 04 /2008
Accepted: 24 / 04 /2008
Available online: 30/04/2008
DOI: 10.37652/juaps.2008.15412

Keywords:
Key distribution,
One-time pad,
Quantum Cryptography,
SSL/TLS,
Unconditional security,
Universal hashing.

ABSTRACT

SSL/TLS is the protocol that is used for the vast majority of secure transactions over the Internet. However, this protocol needs to be extended in order to create a promising platform for the integration of quantum cryptography (QC) into the Internet infrastructure. This paper presents a novel extension of SSL/TLS that significantly facilitates such type of integration. This extended version of SSL/TLS is called QSSL (Quantum SSL). During the development of QSSL, a concentration has been made on the creation of a simple, efficient, general, and flexible architecture that enables the deployment of practical quantum cryptographic-based security applications. Indeed, QSSL efficiently supports unconditionally secure encryption (one-time pad) and/or unconditionally secure authentication (based on universal hashing). A simplified version of QSSL based on BB84 (Bennett-Brassard 84) quantum key distribution (QKD) protocol has been implemented and experimentally tested. This has enabled us to experimentally assess our protocol design based on software simulation of the quantum channel events used for QKD.

Introduction

While conventional methods continue to meet the more-demanding information security needs of our increasingly networked world, they face increasing technological challenges. It is well believed now that QC has the potential to counter these threats and help to meet these future requirements, if it can reach a stage of a sufficient maturity. Hence, in order to facilitate the evolution of QC towards a practical "quantum information security era" in which QC becomes more closely integrated with conventional information security systems and communication networks infrastructures, a more collaborated scientific research among specialists from several fields is required.

Throughout this paper, a focus is maintained on the subfield of QKD. QKD basically enables two parties (traditionally referred to as Alice and Bob) to produce the shared secret keys required for secure communications, through a combination of quantum and conventional communication steps.

Today QKD systems can be operated over metro-area distances on optical fibers, and across multi-kilometer line-of-sight "free-space" paths. Thus, in addition to stand-alone point-to-point (PTP) systems, QKD can be integrated within optical communication networks at the physical layer, and with key-management infrastructures.

The work presented in this paper proposes a novel extension of SSL/TLS (Secure Socket Layer/Transport Layer Security) that we call QSSL (Quantum SSL). QSSL allows the integration of QKD

* Corresponding author at: College of Computers, University of Anbar, Iraq.E-mail address: **sufyantaih@yahoo.com**

capabilities within the Internet (or intranet) security architecture. This significantly facilitates applications in the environments of "QKD networks".

Some aspects of QKD networks have been recently addressed in the literature. The "world's first" QKD network that is composed of trusted relays and/or untrusted photonic switches had been continuously running since June 2004 under the sponsorship of the US DARPA [1], [2].

This network uses a modification of IPSec to integrate it with QKD. In Europe, the SECOQC project will culminate in demonstrating information-theoretically secure QKD over a fiber-based MAN in 2008. In this project, a dedicated key distribution network infrastructure has been adopted. It is the so-called "network of secrets" [3].

Despite that the use of SSL/TLS as the consumer of random secret bits obtained from QKD has been already suggested in [1] and [4], the author is not aware of any specific proposal or design explicitly dealing with the extension of SSL/TLS for QKD integration. To the best of author knowledge, this work represents the first explicitly proposed design and implementation of SSL/TLS extensions for use in various QKD networks.

SSL/TLS OVERVIEW

SSL was originally developed by Netscape. SSLv3 was designed with public review and input from industry. Then, the TLS working group was formed within IETF (Internet Engineering Task Force) and published TLSv1.0 [5] that is very close to SSLv3 and can be viewed as SSLv3.1. Later, TLSv1.1 [6], which is a minor modification of TLSv1.0, had been proposed.

The "socket layer" lives between the application layer and the transport layer in the TCP/IP protocol stack. SSL/TLS (or just simply SSL) contains two layers of protocols. The SSL Record Protocol provides basic security services to various higher-layer protocols and defines the format used to transmit data. Also, SSL defines three higher-layer protocols that use the SSL Record Protocol. These three protocols are used in the management of SSL exchanges. The first is the Change Cipher Spec Protocol, which updates the cipher suite (list of a combination of cryptographic algorithms) to be used on SSL connection. The second is the Alert Protocol that is used to convey SSL-related alerts to the peer entity. The third is the Handshake Protocol, which is the most complex part of SSL. Four content types are defined by the Record Protocol. These are the three SSL-specific protocols (change-cipher-spec, alert, and handshake) and application-data, which corresponds to any application that might use SSL.

The Handshake Protocol allows the two communicating parties (client and server) to authenticate each other. It also enables them to negotiate an encryption algorithm, a MAC, and cryptographic keys required to protect data sent in SSL. The Handshake Protocol consists of a series of messages exchanged by client and server, as shown in Figure 1. This exchange can be viewed as having four phases [7]:

Phase 1- Establish security capabilities:

This phase is used to initiate a logical connection and to establish the associated security capabilities. It starts by a client-hello message and ends with a server- hello message. During this phase, the client and server negotiate the SSL version to be used, session ID, compression method, and the cipher suite. They also exchange random structures to serve as nonces. A cipher suite defines a key exchange algorithm and a CipherSpec, which includes encryption algorithm, MAC algorithm, and some other related information. The exchange methods supported by SSL/TLS are: RSA, fixed DH (Diffie-Hellman), ephemeral (temporary) DH, and anonymous DH.

Phase 2- Server authentication and key exchange:

In the beginning of this phase, the server sends its certificate (if it needs to be authenticated). This certificate message is required for any agreed-on key exchange except anonymous DH. Next, a server-key-exchange message can be sent (if required). This message is not needed if an RSA key exchange is used or if the server has sent a certificate with fixed DH parameters.

Then, a nonanonymous server can request a certificate from the client by sending the certificate-request message. Finally, this phase ends with a server-done message. Phase 3- Client authentication and key exchange:

The client begins this phase by sending a certificate message (if the server has requested it). Next, it sends the client-key-exchange message whose purpose is to enable the client and the server to create a pre-master-secret.

The content of this message depends on the key exchange method. The exchanged pre-master-secret is to be used later by both parties to calculate the shared master-secret, which is a 384-bit value that is generated for each session.

Then, CipherSpec parameters are generated from the master-secret using a certain hashing technique. These parameters are a client write MAC secret, a server write MAC secret, a client write key, a server write key, a

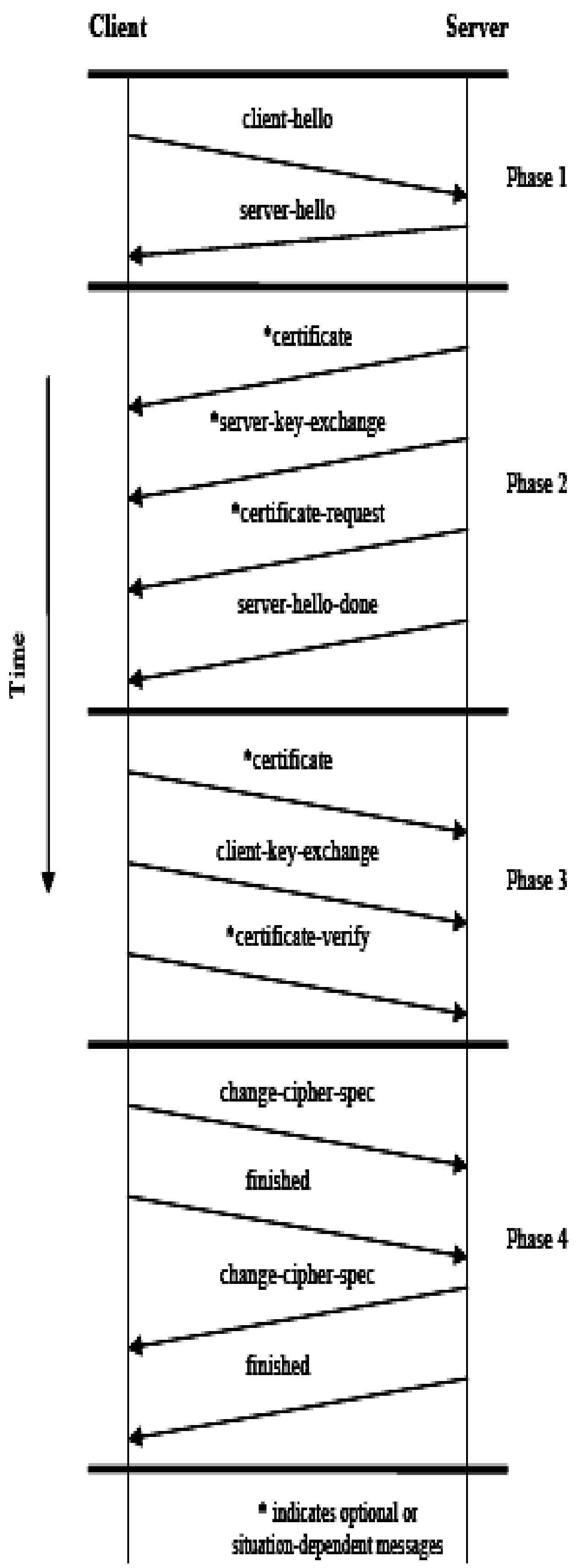


Figure 1: Handshake Protocol Message Exchange.

In two recent RFCs (Request for Comments), three new sets of pre-shared key (PSK) cipher suites have been defined for TLS. These PSKs are symmetric keys, shared in advance among the communicating parties. The first set of these cipher suites uses only symmetric-key operations for authentication. The second set uses DH exchange authenticated with a PSK. The third set combines public-key authentication of the server with PSK authentication of the client. Indeed, these PSK cipher suites may be used in an authentication-only mode, where they can offer authentication and integrity protection with no confidentiality [8], [9].

THE QSSL PROTOCOL

In this section, the Quantum SSL (QSSL) protocol is described. This is mainly done by describing the most important modifications and extensions introduced to the "conventional" SSL/TLS. In the beginning, some important design issues of QSSL are presented as follows:

1- Simplicity and efficiency: During the development of QSSL, we have tried to introduce the minimum possible modifications and extensions to SSL that result in an efficient integration of QC within SSL. This approach also has enabled the avoidance of designing a completely new protocol, which may contain an expected security flaws. Indeed, this integration ha efficiently facilitated both of SSL and QC to get benefits from each other. On one hand, SSL can obtain fresh secret keys from QKD. On the other hand, the required classical public discussions of QC can be conveyed using SSL encapsulation.
2- Generality: QSSL is not directed towards a specific QKD protocol implementation. We have tried to make the extensions as general as possible such that different QKD implementations and phase components can be considered. Indeed, other QC protocols rather than QKD might also be considered in the future.
3- Traditional vs. unconditionally secure encryption: Each of the unconditionally secure encryption using OTP and traditional encryption (such as 3DES, AES, etc.) has its own advantages and requirements. Hence, QSSL supports both types of encryption (Note that "conventional" SSL/TLS does not support OTP).

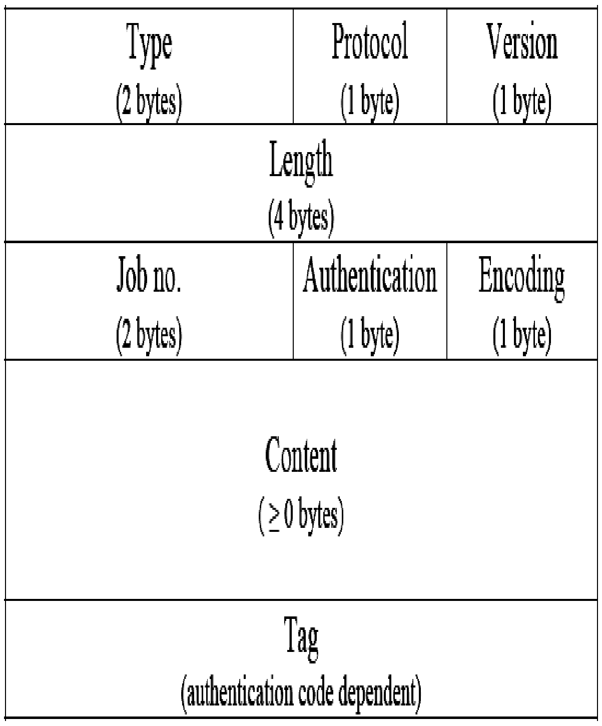
Message authentication: Traditionally used MACs can only offer computationally-secure data integrity. But authentication codes based on universal hashing may offer unconditionally-secure data integrity. However, these authentication codes need secret bits to be initially shared by authorized parties. As QKD can be used as a source for these bits, QSSL supports both types of message authentication for the data traffic. This introduction of the service of unconditionally-secure data integrity for the application.

4- Traffic might be one of the important novel aspects of QSSL.

Various aspects of QSSL are described in the following subsections. Note that for a reason of clarity, QSSL handshake is described as two modes. Mode-1 is based on the traditional public-key initialization of SSL/TLS. Mode-2 is based on PSK cipher suites. The protocol version to be initially used for QSSL is 3.5 (remember that SSLv3 uses 3.0, TLSv1.0 uses 3.1, and TLSv1.1 uses 3.2). Introducing a new SSL/TLS content type

A fifth content type is introduced for the SSL Record Protocol. This new type is to be called quantum-cryptography and it is related to any QC protocol to be integrated within SSL/TLS. By defining this content type, the QC protocol (e.g. QKD) is to be considered as an additional higher-layer SSL/TLS protocol just like the Handshake, the Change Cipher Spec, and the Alert protocols. The message format of the QSSL Quantum Cryptography Protocol is shown in Figure 2. Each message of the Quantum Cryptography Protocol has the following fields:

- Type (2 bytes): Indicates one of the messages used in the public exchange phase of the QC protocol.
- Protocol (1 byte): The QC protocol used (e.g. the BB84 QKD protocol due to Bennett and Brassard [10]).
- Version (1 byte): Enables the use of more than one version of a certain QC protocol. Thus, components of different characteristics can be used to implement any phase of that protocol.
- Length (4bytes): The length of the message in bytes.
- Job no. (2 bytes): Enables the operation of multiple QC protocol jobs (or instances), all belonging to a one QSSL session.
- Authentication (1 byte): Indicates whether this message is authenticated. It may also contain some additional information about this authentication (if any).
- Encoding (1 byte): Indicates if a certain encoding technique is used for the content field of the message. It may also contain some additional information about this encoding (if any). For example, a form of run-length encoding is used for the (sparse) messages of the QKD sifting phase.
- Content ($\geq$ 0 bytes): The parameters and data associated with this message.
- Tag: If the message is authenticated, this field would contain the corresponding authentication tag. Its size depends on the used authentication code.



Type (2 bytes)	Protocol (1 byte)	Version (1 byte)
Length (4 bytes)		
Job no. (2 bytes)	Authentication (1 byte)	Encoding (1 byte)
Content (≥ 0 bytes)		
Tag (authentication code dependent)		

Figure 2: Message Format of the SSL Quantum Cryptography Protocol

Defining new cipher suites

Here, two new sets of cipher suites are defined to be used for QSSL. The first of these sets uses public-key cryptography to initialize the QKD process. This set is to be applied whenever there is no possibility for authorized users to initially have PSKs required for universal hashing. This set may also be used when users believe that the security offered by such cipher suites is adequate for them. This situation represents what we call QSSL Mode-1. The second set of cipher suites uses PSKs to facilitate the use of unconditionally-secure authentication for protecting the QKD public channel. This set can offer provable security and it corresponds to the second mode of QSSL handshaking (QSSL Mode-2).

As far as the first set of cipher suites is concerned, network initialization can be done using any of following "conventional" SSL/TLS key exchange methods, which are: RSA, fixed DH, or ephemeral DH (DHE). However, the use of anonymous DH key exchange is not supported by QSSL because it makes the system vulnerable to man-in-the-middle attacks. Furthermore, this set can only use traditional MACs (such as SHA and MD5) for protecting the QKD public channel. This is obviously due to the assumption unavailability of PSKs required for unconditionally-secure authentication. In contrast, the second set of cipher suites always use unconditionally-secure authentication for protecting QKD public channel discussions. This is implied by using PSKs for system initialization in this latter mode.

It is very important to notice that both sets of cipher suites support the use of traditional encryption algorithms and OTP. Also, they both support either traditional MACs or unconditionally-secure authentication codes to offer data integrity for QSSL application traffic. This is totally justified since both sets are defining QKD to supply the required secret bits.

Cryptograph computations

QSSL uses QKD to directly generate the following cryptographic parameters: client write MAC secret, server write MAC secret, client write key, server write key, client write IV, server write IV, and pre-master-secret. Note that we have used the same terminology of SSL/TLS.

However, the write MAC secrets are used in the generation of both of traditional and unconditionally-secure message digests. Similarly, the write keys are used for both traditional and unconditionally-secure encryption. The write IVs are only generated when traditional block ciphers are used for encrypting application traffic. Whenever unconditionally -secure encryption and/or authentication are to be used, the size of the required write keys and/or the size of the required write MAC secrets have to be negotiated during QSSL handshake.

The pre-master-secret generated from QKD is divided into two parts. The first 48 bytes of it compose the first part, which we call pre-master-secret-1. The remaining bits of the pre-master-secret compose the second part that is to be called pre-master-secret-2. Then, the master-secret is calculated from pre-master-secret-1. The function used for calculating the master-secret in QSSL is similar to that used by SSL/TLS. Next, the master-secret can be used for authenticating the finished handshake messages and for resuming QSSL sessions (when it is allowed). QSSL sessions can be resumed if and only if both of encryption and message digest algorithms used for application traffic are not unconditionally-secure. Otherwise, connections cannot be generated from sessions because this can be fatal for the specified property of unconditional security.

The pre-master-secret-2 is to be used as a PSK for initiating future QSSL sessions. Hence, its size has to be negotiated by users during handshaking such that its size is adequate to enable the use of unconditionally-secure authentication for protecting the QKD public channel. Note that this separation of the pre-master-secret into two independent parts is necessary from the respective of unconditional security.

QSSL Mode-1

This mode represents QSSL handshaking based on using public-key cryptography for initialization. Any of the traditional SSL/TLS key exchange techniques (except for the anonymous DH) can be used. The sequence of message exchange of QSSL Mode-1 is very similar to that of SSL/TLS described previously in Section 2. However, there are some required modifications to be noted. The most important of these are:

1- At least three new parameters should be added for negotiation in the client-hello and server-hello messages. These parameters are the size of write MAC secrets, the size of write keys, and the size of the pre-master-secret. The first of these is to be added whenever unconditionally-secure authentication is required for QSSL application traffic. The second parameter is included when it is intended to use OTP encryption. Finally, the third parameter is added whenever users have the intention to generate PSKs for future Mode-2 initialization (The size of this parameter should be $\geq$ 48 bytes).
2- QKD (or generally any QC protocol) can only be started after both sides mutually authenticate each other. Also, QKD has to be finished before exchanging any change-cipher-spec message. Hence, the whole QKD message exchange is inserted between Phase 3 and Phase 4 of SSL Handshake described previously.
3- QKD continues until the complete generation of the negotiated key sizes. After this point only, change-cipher-spec and finished messages (Phase 4) can be exchanged (This issue also applies to QSSL Mode-2).

QSSL Mode-2

This handshaking mode uses PSK based initialization. This offers a very high speed session initialization compared with the relatively slow public-key cryptography based Mode-1 initialization. Figure 3 shows the basic Mode-2 message exchange. QKD message exchange is also is completely inserted just before the transmission of change-cipher-spec and finished messages.

Besides the three security parameters added to the negotiation by the client-hello and server-hello messages mentioned previously, there is an important modification to server-key-exchange and client-key-exchange messages in this mode. In QSSL Mode-2, these two messages are used for identification and synchronization of PSK pads. At first, the server-key-exchange message is used to carry a "PSK identity hint". The client-key-exchange message, when received, is to be interpreted as a positive acknowledgement of the "PSK identity hint". Otherwise, an unknown-psk-identity alert**.**

SYSTEM ARCHITECTURE AND IMPLEMENTATION

The system architecture of a typical QSSL application is shown in Figure 4. As the figure indicates, QSSL is located between application layer protocols and TCP. On transmission, QSSL accepts data traffic from the application layer and adds the required security protection before sending it down to lower layers. The BB84 is used as the QC protocol in this architecture. QSSL uses the network to send two types of traffic. The first is application data traffic, which is encrypted and authenticated according to user's requirement.

The second is the traffic corresponding to the classical public channel exchanges required for implementing BB84. This latter traffic type is transmitted during QSSL handshake and it has to be authenticated in order to defeat man-in-the-middle attack on the QKD protocol The quantum transmission phase of BB84 is done using the quantum channel. This is mainly a physical layer technology. Hence, management and control of this phase is performed at the lower layers of the architecture. Other phases of

BB84 (namely sifting, reconciliation, estimating Eve's information, and privacy amplification) uses the classical public channel for the required discussions. These phases are implemented using a specific user-mode application module that we call BB84 QKD protocol engine. The input of this engine is raw quantum bits resultant from quantum transmission, while its output is secret bits that are (temporarily) stored in the key storage and. management module. These bits are used by QSSL as cryptographic keys in accordance to the specified cipher suites.

A simplified version of QSSL has been implemented based on Microsoft "WinSock" technology using Visual C/C++ v.6 with system running under Windows XP. This QSSL implementation has been integrated with our previously developed BB84 protocol engine [11].

This protocol engine is based on software simulation of the BB84 physical layer. It also includes a suitable code for performing other BB84 phases. The required unconditionally-secure authentication for BB84 public channel communications has been achieved according to our previously developed authenticated version of BB84 [12], which uses Taylor's authentication code.

The experimental setup consists of two QSSL instances installed onto two PCs (each with 1.7 GHz Intel Pentium IV processor). These two PCs are connected via an Ethernet. Sample results showing the amount of net key expansion gain for different QSSL sessions are illustrated in Figure 5 and Figure 6. In these two figures, all QKD sessions have been performed at a quantum bit error rate (QBER) of 5%. Also, 20% of sifted bits have been used for public comparison to estimate the amount of QBER.

This estimation is required to set the parameters of the reconciliation phase and to decide whether to proceed with the QKD protocol or not. Eve's information about the cryptographic keys generated from all QKD sessions has been reduced well below 10^{-70} bit using the technique of privacy amplification. message has to be sent by the client.

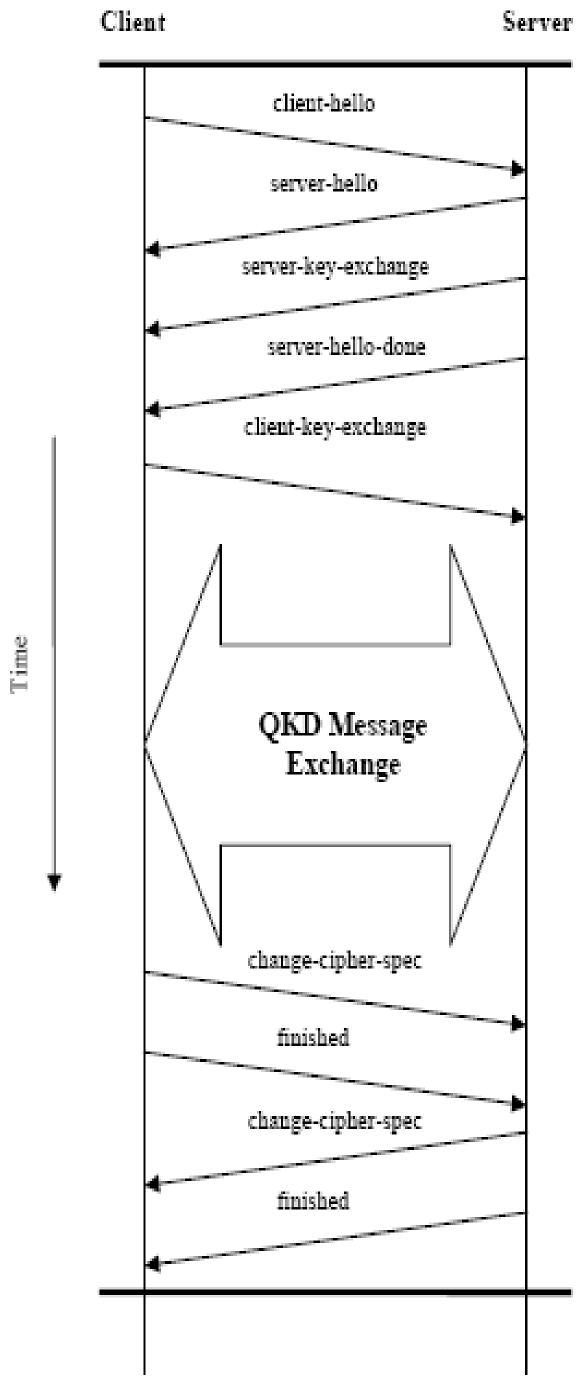


Figure 3: QSSL Mode-2 Handshake.

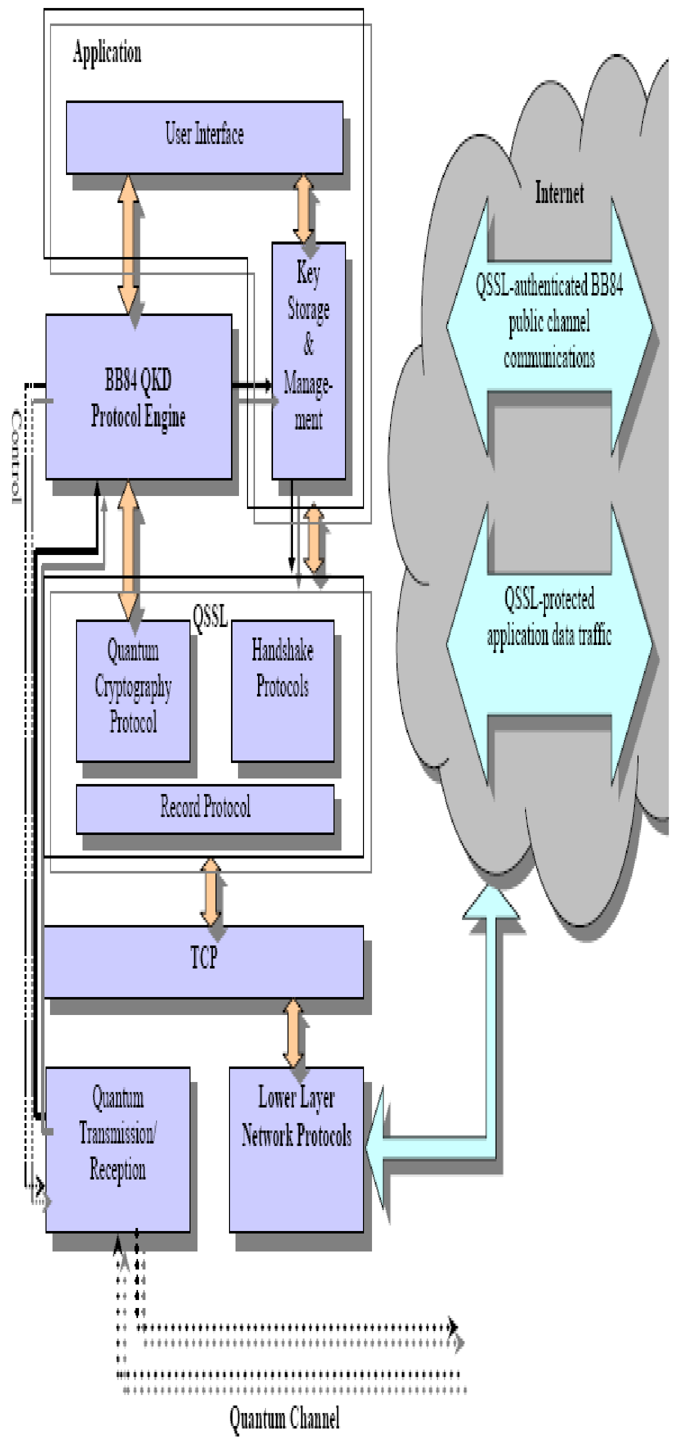

Application
User Interface
Key Storage & Manage-ment
BB84 QKD Protocol Engine
Control
QSSL
Quantum Cryptography Protocol
Handshake Protocols
Record Protocol
TCP
Quantum Transmission/ Reception
Lower Layer Network Protocols
Quantum Channel
Internet
QSSL-authenticated BB84 public channel communications
QSSL-protected application data traffic

Figure 4: System Architecture for a Typical QSSL Application.

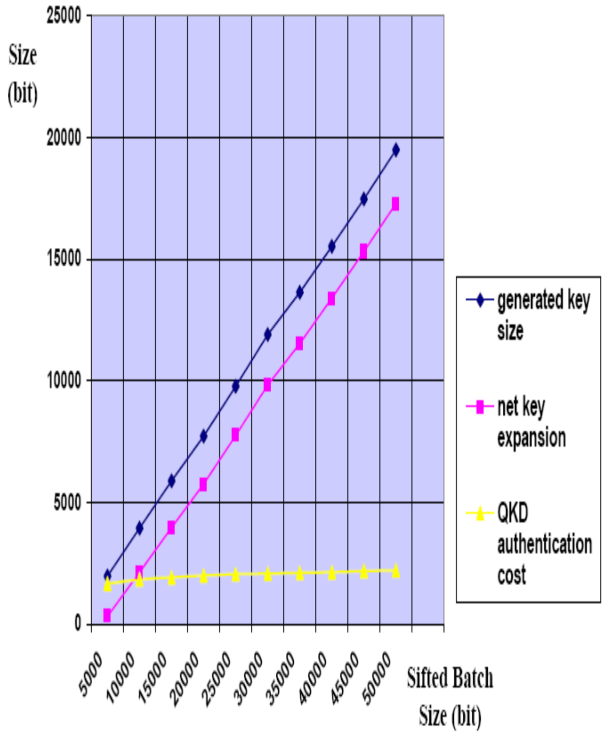


Figure 5: Net Key Expansion Gain for QSSL Mode-2 Sessions.

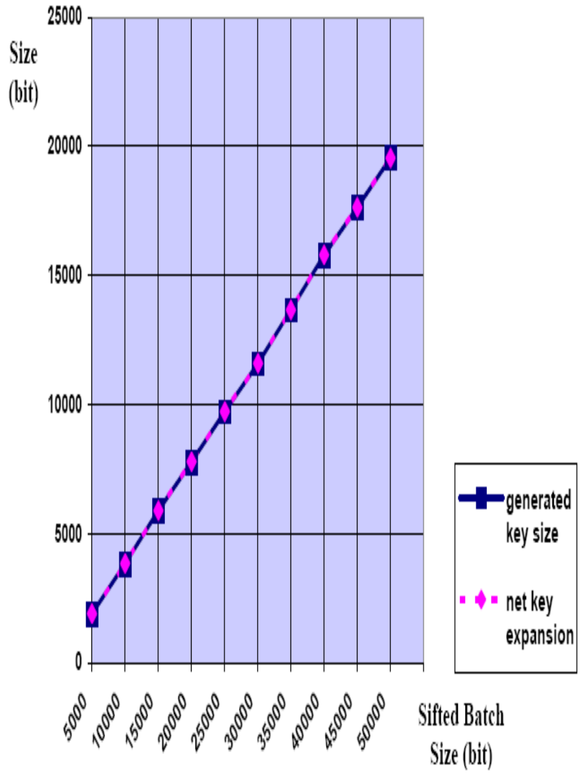


Figure 6: Net Key Expansion Gain for QSSL Mode-1 Sessions.

Both of figures 5 and 6 show key expansion results for different batch sizes of sifted bit strings ranging from 5000 to 50000 bits. Figure 5 represents typical results obtained from QSSL Mode-2 sessions, whereby Taylor's authentication tags of 31-bit size has been used to protect public channel exchanges.

Figure 6 shows typical results for the corresponding QSSL Mode-1 sessions. In this latter case, a traditional SSL/TLS message authentication technique (using SHA-1 algorithm) has been used to implement the authentication of QKD public channel discussion. Thus, the authentication cost of this mode is null. Hence, the amount of key expansion for Mode-1 sessions is greater than that obtained from the corresponding Mode-2 sessions. However, keys produced by Mode-1 sessions do not have the property of unconditional security as those resultant from Mode-2 sessions.

CONCLUSION

It is well justified and prudent now to obtain unconditionally-secure services based on combining QKD with OTP and/or unconditionally-secure authentication codes. However, investigating the full flavor of such services requires multi-disciplinary research efforts. We believe that proposing QSSL is a useful step towards a better understanding of the requirements of integrating QC into the already existent and well-tested information security infrastructure. Inspired by QSSL, our next goal is to present an extension of IPSec for QC. This could lead us to deeper insights on the issue of integrating QC protocols within different layers of the Internet protocol stack.

REFERENCES:

[1] C. Elliott, D. Pearson, and G. Troxel, "Quantum cryptography in practice," ACM SIGCOMM'03 Conference, Germany, August 2003, pp. 227-238.

[2] C. Elliott et al, "Current status of the DARPA quantum network," BBN Technologies, arXiv: quant-ph/0503058, March 2005.

[3] R. Alleaume, Ed., "SECOQC white paper on quantum key distribution and cryptography," Secoqc-WP-v5, Version 5.1, January 2007.

[4] C. Williams et al, "a high speed quantum communication testbed," NIST Proceedings, 2002.

[5] T. Dierks and C. Allen, "The TLS protocol version 1.0," RFC 2246, January 1999.

[6] T. Dierks and E. Rescorla, "The TLS protocol version 1.1," RFC 4346, April 2006.

[7] W. Stallings, Cryptography and Network Security, 3rd Edition, Pearson Education International, USA, 2003.

[8] P. Eronen and H. Tschofeing, Eds., "Pre-shared key ciphersuites for TLS," RFC 4279, December 2005.

[9] U. Blumenthal and P. Goel, "Pre-shared key ciphersuites with NULL encryption for TLS," RFC 4785, January 2007.

[10] C. Bennett and G. Brassard, "Quantum cryptography: Public key distribution and coin tossing," International Conference on Computers, Systems & Signal Processing, India, December 1984, pp. 175-179.

[11] S. Faraj et al, "Optical network models for quantum cryptography," Proceedings of 17th IFIP/Sec2002 Conference, Egypt, May 2002.

[12] S. Faraj, "Unconditionally secure authentication in quantum key distribution," i-Manager's Journal on Software Engineering, Vol. 1, No. 3, January-March 2007, pp. 30-42.

توسيع بروتوكول SSL/TLS لأغراض التشفير الكمي

سفيان تايه فرج

E.mail:sufyantaih@yahoo.com

الخلاصة:

يعتبر بروتوكول SSL/TLS هو أكثر البروتوكولات المستخدمة لاجراء الاتصالات الأمينة عبر الأنترنت. ومع ذلك فإنه يجب توسيع هذا البروتوكول لكي نبني منه أرضية واعدة لدمج تطبيقات التشفير الكمي ضمن الهيكلية الحالية لبنية أمن المعلومات على الأنترنت. يقدم هذا البحث بروتوكولا مبتكرا لتحقيق هذا الاندماج وقد سميناه QSSL. وخلال تطوير QSSL فقد تم مراعاة أن يكون بسيطا وكفوءا وعاما ومرنا قدر الامكان وبما يؤهله لدعم مختلف تطبيقات أمنية الشبكات المبنية على خواص التشفير الكمي. أضافة لذلك، فإن QSSL يوفر – ضمن ما يوفره – امكانية تحقيق التشفير ذو الأمنية غير المشروطة و/أو التوثيق ذو الأمنية غير المشروطة. هذا وقد تم تنفيذ واختبار نسخة من QSSL مبنية على أساس بروتوكول بينيت–براسارد–٨٤ للتوزيع الكمي لمفاتيح التشفير. وقد مكننا ذلك من الوصول إلى القناعة اللازمة لتقويم هذا البروتوكول واثبات نجاعته للغرض الذي تم تصميمه من أجله.