KEY MANAGEMENT OF QUANTUM GENERATED KEYS IN IPSEC

Andreas Neppach, Christian Pfaffel-Janser, Ilse Wimberger
Program and System Engineering (PSE), Siemens AG Austria, Gudrunstrasse 11, Vienna, Austria

Thomas Loruenser, Michael Meyenburg
Smart Systems Division, Austrian Research Centers GmbH, Vienna, Austria

Alexander Szekely, Johannes Wolkerstorfer
Institute for Applied Information Processing and Communications, Graz University of Technology, Austria

Keywords: Quantum cryptography, Key management, Virtual Private Networks (VPN), IPsec, Security gateway, Internet Key Exchange (IKE).

Abstract: This paper presents a key management approach for quantum generated keys and its integration into the IPsec/IKE protocol. The solution is used in a security gateway that integrates quantum key distribution (QKD) and IPsec as a system-on-chip solution. The QKD acquisition module and the IPsec part of this prototype are implemented in hardware to provide a high level of integration as well as high encryption throughput. To make use of these fast encryption capabilities, a flexible key management approach is necessary to provide keys just in time. Thus, the presented key management approach focuses on an efficient key update mechanism and minimizes the communication overhead. Furthermore, the presented approach is a first step to integrate QKD solutions into real-world commercial applications using standardized interfaces.

1 INTRODUCTION

Virtual private networks (VPN) are a standard in today's network infrastructure. Nearly all business areas use the Internet to exchange data and information. Many parts of this information are confidential and thus have to be protected. A VPN provides this functionality and ensures the confidentiality and integrity of data transferred between two endpoints, even if the transport medium is insecure.

VPNs typically operate at OSI layer 2 - 4. One popular protocol is the Internet Protocol security (IPsec) (Kent, 2005). It operates at layer 3, and protects all protocols in upper layers (e.g. HTTP, FTP or SMTP). This simplifies the integration into existing network infrastructures. Many IPsec implementations are available for different operating systems and enable client PCs to perform remote access to e.g. company intranet or confidential network resources. In case of e.g. data replication of large databases or interconnection of a headquarter with its subsidiaries, IPsec implementations in software are not suitable to fulfill high throughput requirements. In this case a VPN gateway with hardware accelerated IPsec stack will be used to perform the cryptographic operations.

The cryptographic keys in IPsec are negotiated using the Internet Key Exchange protocol (IKE). IKE is defined in (Kaufman, 2005) and negotiates security parameters for the IPsec sessions. The key exchange is based on the Diffie-Hellman key exchange protocol (DH) defined in (Diffie and Hellman, 1976). DH is based on the problem to solve the discrete logarithm (Menezes et al., 1997). Up to now, no algorithm is known, that is able to solve this problem in polynomial time (Menezes et al., 1997). However using short key update rates, the DH key exchanges increase the network load and decrease the overall throughput of the VPN by consuming much computational power.

In our work, we focus on an efficient VPN gateway that uses quantum key distribution (QKD) as an alternative key exchange method in IKE. QKD is an

Neppach A., Pfaffel-Janser C., Wimberger I., Loruenser T., Meyenburg M., Szekely A. and Wolkerstorfer J. (2008).
KEY MANAGEMENT OF QUANTUM GENERATED KEYS IN IPSEC.
In *Proceedings of the International Conference on Security and Cryptography*, pages 177-183
DOI: 10.5220/0001926601770183

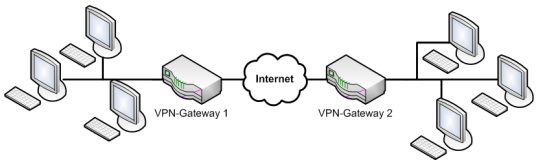

Figure 1: Network Architecture.

emerging field in security engineering. The technique was invented in 1984 by C. Bennett and G. Brassard (Bennett and Brassard, 1984) and enables theoretically secure distribution of random strings. The security of QKD is based on the laws of quantum physics. All practical attempts to eavesdrop QKD generated keys can be noticed.

Using the QKD generated random strings, we implemented a fast key update mechanism for the underlying IPsec engine. The goal was to get a key update rate of at least one key update per second. Therefore, the prototype is implemented as a System-on-Chip (SoC). The IPsec engine and parts of the QKD module are implemented in configurable hardware (Lorünser et al., 2008). This was mainly necessary to accelerate encryption and to offload constant cryptographic tasks. The key management is implemented in software.

The prototype can be used to get a basis of requirements for QKD integration in existing applications using standardized interfaces. In this paper, we focus on the key management approach, showing the key management protocols and internal interconnection of key management related components.

In the remainder of this paper we present the overall architecture of the prototype in Section 2. Section 3 describes the key management approach and presents the adaptations of the IKE protocol. Section 4 concludes the paper.

2 ARCHITECTURE

The overall network architecture of the prototype is presented in Figure 1. The system acts as a security gateway and provides an end point of a secured IP tunnel. All data arriving at gateway 1 from the protected LAN is processed using a policy database. Depending on the policy the data is either rejected, transferred in plain or encrypted using the QKD generated keys.

Figure 2 shows the components of the VPN gateway solution. The QKD module generates the secure random bit strings that are processed by a key management module. The key management approach is based on a key distribution component and an adaptation of the IKE protocol. Using IKE, so called security associations (SAs) are established. These SAs are

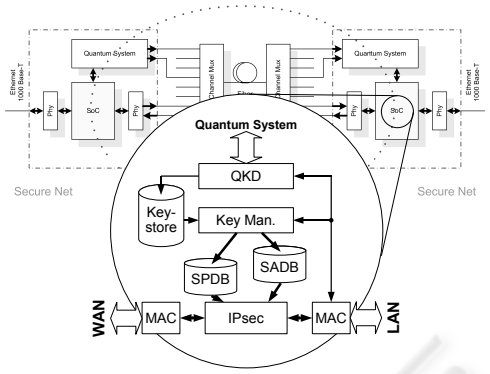

Figure 2: Prototype Architecture.

stored in the security associations database (SADB) and contain QKD generated keys that are used by the IPsec engine to process arriving data packets. Details about the QKD and IPsec hardware modules are presented in the next subsections. The key management approaches are presented in Section 3.

2.1 QKD

The QKD sub-system is responsible for the generation of the shared secrets and it is used to replace the original IKE scheme. In general, a quantum key-exchange system provides provably secure distribution of random secrets over a quantum channel, which can either be an optical fibre link or a free-space line of sight connection.

Light particles (photons) are encoded on one side and transmitted to the second peer. After continuously measuring the arriving photons the two parties gain correlated data to establish a provably secure shared secret. The distillation of the final key out of the correlations is done by means of software processing and the implementation is referred to as QKD stack.

Therefore, a complete QKD system consists of a quantum optical segment, a data acquisition system (DAQ) and a processing segment (QKD stack). The latter requires an authenticated classical communication channel for final key distillation which does not need to be confidential. The authentication is necessary to prevent from man-in-the-middle attacks like it is also the case in DH key exchange.

It is seen, that, in comparison to nowadays very popular key exchange protocols and algorithms, QKD needs a considerable hardware effort. We developed electronics and a DAQ core for interfacing with entangled QKD systems applying the BB84E protocol. The DAQ unit is integrated in the SoC and can handle up to 30 million events per second (MEvents) peak rate with a timing resolution down to 80 picoseconds.

The software processing for the QKD stack runs on the integrated processor and it can generate up to 3 kbps final key material.

QKD introduces a new quality for symmetric key exchange primitives. It enables provable security and high key-update rates. But it also has some drawback. Due to the non-cloning theorem and the attenuation QKD is limited in distance. The optical signal cannot be repeated and the damped signals lead to lower rates over distance. Current QKD links provide key rates in the kilobit range over distances up to 100 km. Moreover, some new protocols and advances in single photon detection will allow for rates in the Mbps region.

2.2 IPsec

The Internet Protocol Security (IPsec) suite of protocols defines a collection of standards to secure IP traffic. It provides a framework to secure communication at the network layer by adding authentication data and/or encryption to IP packets. Our implementation supports only the Encapsulating Security Payload (ESP) protocol in tunnel mode. ESP in tunnel mode is the most useful setting for gateway-to-gateway VPNs. It provides data authentication, integrity and confidentiality to IP packets. ESP encrypts the whole original packet and adds authentication data and an ESP header. Encrypted ESP packets guarantee even the anonymity of sender and receiver because this information is encrypted too. The final ESP packet contains the IP addresses of the two gateways as source and destination address.

IPsec defines a bunch of symmetric ciphers. Our implementation makes uses of the Advanced Encryption Standard (AES) (National Institute of Standards and Technology (NIST), 2001). In accordance to the recommendations of VPN-B in (Hoffman, 2005) our implementation uses AES-128 in CBC mode for data confidentiality and AES-XCBC-MAC-96 (Frankel and Herbert, 2003) for data authentication. AES is processed by hardware modules.

The whole IPsec handling of the prototype is performed by an IPsec offload engine, as the power of the embedded PowerPC CPU on the system-on-chip is very limited. To support throughput rates in the gigabit region, most components of the IPsec functionality are implemented in hardware. The software subsystem is only used to configure and monitor these hardware modules.

The IPsec engine is composed of three main components: A filter module, an encryption unit and a routing module. The filter module holds a copy of the security policy database (SPD) and decides if an arriving IP packet should bypass the system, or if it has to be encrypted/decrypted, or if needs to be discarded. The encryption unit consist of a key store, which resembles the SADB, and AES encryption and decryption modules. The key store can hold up to 32 SAs per direction and manages their lifetime without software interaction. After leaving the encryption unit, the packets are forwarded to their destination gateway by the routing module. Routing is necessary because the destination IP address is known just after decrypting the ESP packet. The according Ethernet MAC address of the destination has to be obtained for each packet.

The software part of the IPsec engine runs on a Linux 2.6 kernel. It is responsible for the configuration of the hardware modules. The software receives the configuration and key material via the PF_KEY interface. Hence it is compatible with all standard IPsec key management implementations.

2.3 WBEM

VPN gateways offer complex functionality. Configuration of its services usually involves setting many parameters on at least two endpoints. A secure management interface has to assure that the configurations on all devices are synchronized. To prevent temporary security holes a strict sequence of commands needs to be executed. This is error-prone and leads to security risks, especially when traditional configuration tools like secure shell (SSH) are used.

For this reasons, we decided to use Web-Based Enterprise Management (WBEM) (DMTF,) as management technology. It is one approach of solving the management problem in heterogeneous networks. WBEM is a set of standards that has been developed by the Distributed Management Task Force (DMTF). WBEM-based solutions provide out-of-the-box capabilities to make policy deployment on different machines transparent to the user.

Our implementation uses the Small Footprint CIM Broker (SFCB) from IBM (Standards Based Linux Instrumentation,) as WBEM server running on each gateway. In addition we developed a graphical tool that allows to easily configure and monitor VPN gateways.

3 KEY MANAGEMENT

The presented key management approach focuses primarily on high key update rates for IPsec. Additionally, the management of the generated QKD keys is of interest. Figure 3 shows the key related operations of

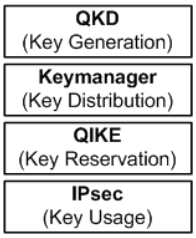

Figure 3: Key Management Overview.

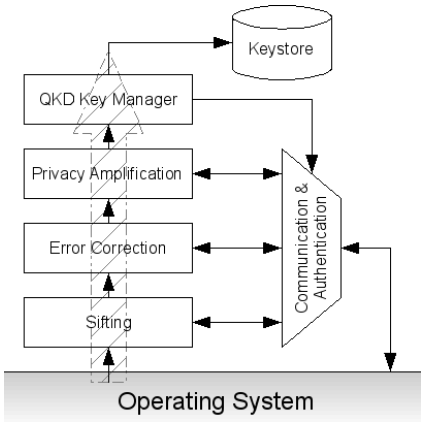

Figure 4: QKD Software Architecture.

the prototype. First, the QKD module generates the random bit strings and buffers them. This buffer is accessed by a separate process (called Keymanager) that establishes the end-to-end key buffers for local applications. The Keymanager distributes the generated keys to registered applications and ensures that key buffers of local and remote applications remain synchronized. One of these applications is an adaptation of the IKE protocol called QIKE. The main task of QIKE is to negotiate SAs and parameters to reserve a QKD key stream. Using the negotiated parameters, a key buffer can be requested from the Keymanager and the SADB can be filled with SAs containing QKD generated keys. These SAs are used later on by the IPsec module that processes arriving data packets.

3.1 QKD Key Generation

In our implementation the QKD sub-system is continuously generating secrets. The DAQ unit streams the incoming raw data into a buffer and hands it over to the QKD stack, which distills final keys. The incoming raw data are sifted, error corrected, privacy amplified and pushed into the keystore. We make use of well established protocols and algorithms for the QKD stack.

The necessary classical communication between the two parties is managed over TCP/IP connections

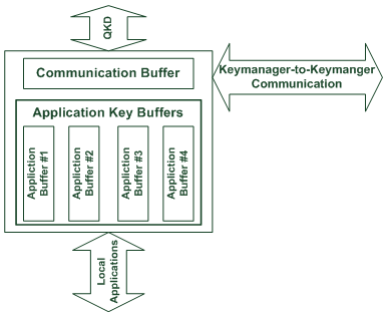

Figure 5: Allocated key buffers.

in plaintext but authenticated. The software stack is programmed under heavy usage of multi-processing and Linux/Unix based interprocess communication to optimize CPU and network usage. Its overall architecture is shown in 4.

During the sifting phase the bases of the protocol are exchanged. VPN gateway 1 starts the process after receiving a fixed block of data from the measurement core.

The data are then passed to the error correction stage, which run a performance optimized version of the Cascade protocol (Brassard and Salvail, 1994). Optionally, the stack supports low density parity checking (LDPC) as error correction scheme.

After correcting all errors and confirmation of the bit error rate, the stream enters the privacy amplification stage. This module shrinks the keys according the amount of information revealed in the public conversation during the error correction and the error rate itself. The privacy amplification is in principle a parametric hash function based on the universal 2 classes of hash functions. The keys are pushed into the keystore over a very simple interface.

To prevent man-in-the-middle attacks all communication has to be authenticated and checked for integrity. This is done by applying message authentication codes (MAC) developed for theoretical secure systems based on evaluation hashes (Shoup, 1996). Therefore, all communication is routed through the dedicated authentication module. It calculates and verifies the MAC tags. To do this it needs key material itself.

3.2 Keymanager

The Keymanager is a multi-threaded server running as a UNIX system daemon and is necessary to provide a network layer for QKD. It is necessary to interconnect several QKD links and, furthermore, to distribute the continuous stream of keying material from the QKD sub-system to different consuming applications.

The Keymanager has two main types of threads. Client threads handle local communication on the local host and link threads establish connections to remote hosts and manage the client threads' key buffers. The Keymanager is based on the master-slave principle. The master is the one Keymanager with the lower IP address. The communication between local and remote Keymanager daemon is based on a binary TCP/IP protocol that is protected through configurable hashed MACs (HMAC) against man-in-the-middle attacks and through the inclusion of time tags and message sequence counters against replay attacks. The keys necessary for the HMAC are taken from the quantum link.

During the initialization phase of the control channel, a pool of keys with a well known size is filled on both keymanagers with keys from the QKD device (see communication key buffer in Figure 5). The first bits are used for authenticating the control channel and the second bits are stored for reset purposes. The rest is used for the subsequent control messages.

In the basic configuration the client listening port is only active on the local host interface. Any attack on the Keymanager would thus originate from the local host which can be prevented more easily.

The message exchanges to establish common key buffers for a local and remote application are presented in Figure 6. First, the client application registers to the client thread and submits an application ID and the host address of the Keymanager the client is registering to. If the application ID has already been registered the call for registration fails. A wrong message to the Keymanager results in a client's termination to lower the risks of denial of service (DOS) attacks.

Second, the client application sends a connection request containing the remote node's address, the maximum key length (*maxlen*) that will be requested, the desired key rate (*rate*) and a timeout. If all those parameters are valid and the requested key rate can be satisfied by the local Keymanager, it either tries to connect the client to the remote Keymanager if the local node is the designated master or it queues the client's connect request until a valid connect request from the remote Keymanager is received otherwise. If the connection cannot be established to the remote system within a defined time interval, a timeout occurs and the local client-Keymanager connection is terminated.

After the connection phase, an application can query its keystore to get the current fill level or to get a key of a specified length. If the requested length cannot be satisfied only the amount available is returned. The key data and the actual size of the data retrieved

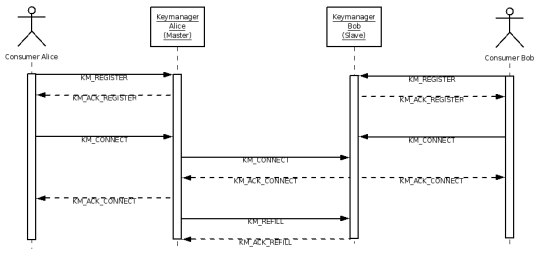

Figure 6: Message flow during connection establishment.

is sent back to the client.

Every client connection has associated key buffers whose size can be calculated using Equation 1, where *time* is the amount of time for which to try to hold keying material.

$$buf_{size} = \max(len_{max}, time \cdot rate) \quad (1)$$

Whenever the fill level of a key buffer drops below a buffer threshold which can be calculated using Equation (2) with a configurable global threshold parameter, a refill procedure for all buffers is started.

$$buf_{threshold} = \max(len_{max}, size \cdot global_{threshold}) \quad (2)$$

The key refill procedure has the following priorities. First, the key store to authenticate the message exchanges between local Keymanager and remote Keymanager is served. If keying material is available, it is assigned to the client key stores with a fill level below their thresholds, according to their registered key rate. If further keying material is left, all remaining key stores are served in proportion to their registered rate.

3.3 QIKE

The implementation of QIKE is based on Racoon, a popular IKE daemon developed by the KAME project (http://www.kame.net/racoon) and IPsec-Tools (http://ipsec-tools.sourceforge.net). It is an open source implementation and distributed under the BSD license. Standard IKE consists of two phases. In the first phase, a so called ISAKMP SA is established. ISAKMP means Internet security association key management protocol and is specified in (D. Maughan and Turner, 1998). The ISAKMP SA is used within the IKE protocol to protect all further message exchanges in phase 2. The negotiated SAs in Phase 2 are called IPsec SAs and are used to establish an encrypted session between two IPsec end points.

The following subsections present the major adaptations of IKE to enable QKD support and high key update rates.

3.3.1 Authentication in QIKE

In IKE, authentication of the communicating devices is based on pre-shared key (PSK) or asymmetric cryptography. Like already mentioned our solution is based on QKD and thus uses the PSK approach. Similar to one-time passwords, we use QKD random strings for authentication purposes. The first communication needs an initial shared pre-shared secret since the communication to the Keymanager is triggered by QIKE. After the first Phase, a key buffer is requested that is filled with new shared secrets obtained from the QKD module.

3.3.2 Phase 1

The first communication phase of QIKE is similar to IKE. First, security association proposals are exchanged. A proposal contains parameters like encryption algorithm, authentication algorithm and key lifetime. Additionally, we introduced the QKD key rate to be reserved and an application ID to establish a key buffer using the Keymanager. After the ISAKMP SA is negotiated, QIKE authenticates the messages using the configured shared secret. If the authentication succeeds, the communication to the Keymanager starts. The QIKE process registers on both sides to the Keymanager using the application ID and the remote IP address. Furthermore, the QKD key rate is used to request a key buffer. If the both QIKE daemons perform their registration in time, a key buffer based on the negotiated QKD key rate and the maximum key size is allocated.

The maximum key size is computed using the requirements of the negotiated encryption and authentication algorithms. Using AES-128 for encryption (National Institute of Standards and Technology (NIST), 2001) and AES-XCBC-MAC-96 for authentication (Frankel and Herbert, 2003), a maximum key size of 256 bits has to be requested (not considering the initialization vector for CBC encryption). Using a key lifetime in seconds or milliseconds, the key rate can be computed without negotiation. However, because of synchronization problems with lifetimes in milliseconds, we decided to choose a lifetime in bytes. Thus, the key rate varies depending on the utilization.

3.3.3 Phase 2

To establish an IPsec SA another SA proposal exchange takes place. This exchange is protected by the ISAKMP SA. The initiator sends an array of SA proposals to the responder, which chooses an appropriate one. Similar to Phase 1, the SA proposal contains

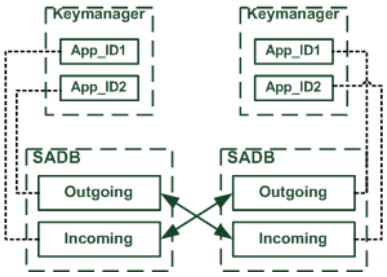

Figure 7: IPsec SADB and QKD Key Buffers.

beside the standard IKE parameters, QKD key rate, an application ID and the maximum number of SAs supported by the SADB. The IPsec SA generation is based on the IKE Quick Mode.

The QKD key rate specifies the rate that is requested from the Keymanager. The application ID identifies the initiator and responder QIKE daemon. The Keymanager uses these IDs and establishes the requested application key buffers. For each direction a separate buffer is allocated. Thus, the key buffer of the outgoing IPsec connection of the initiator QIKE is identical to the key buffer of the incoming IPsec connection of the responder QIKE and vice versa (see Figure 7).

The parameter of the maximum number of supported SA in the SADB is necessary to enable a high-speed IPsec implementation. If the IPsec engine has to wait for new key material, the throughput of the prototype will be decreased. Therefore, multiple SAs can be stored in the SADB. The SADB follows the FIFO principle, in difference to the Linux SADB that uses LIFO. Additionally, only SAs that are currently in use can expire. In some SADB implementations, the SA lifetime in time decreases as soon as the SA is stored into the SADB.

Our solution uses a key lifetime in bytes, since a lifetime in milliseconds will cause synchronization problems. If a lifetime expires, the IPsec engine notifies QIKE using the PF_KEY interface. QIKE uses this notification and generates a new one by requesting new key material from the Keymanager. In difference to IKE, no renegotiation of the SA is performed. This renegotiation is primarily necessary to exchange new DH values and to trigger the remote IKE daemon to check the SADB and to delete not expired SAs.

Instead of renegotiation of expired SAs, QIKE is based on the ISAKMP informational messages. These messages can contain delete notifications that inform the remote daemon about expired SAs. ISAKMP delete messages are unidirectional and no response is expected from the remote daemon. If QIKE receives a delete message, the SADB is checked and not expired SAs are replaced with new

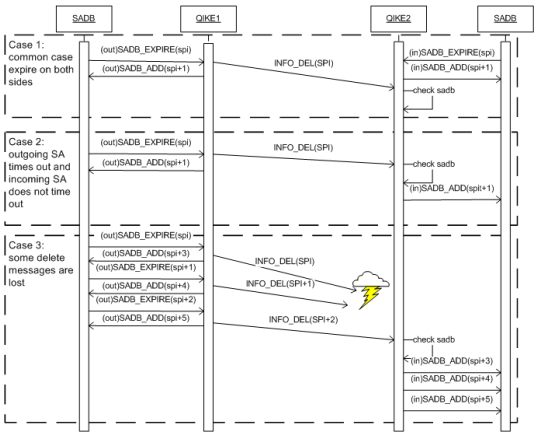

Figure 8: Delete messages and SADB synchronization.

ones (see Figure 8). The ISAKMP informational messages are protected using the ISAKMP SA.

Nevertheless, synchronization problems can occur if ISAKMP delete messages are lost. In our prototype, we use increasing SPIs per connection. This gives the possibility to resynchronize the SADB even if a delete messages gets lost. The QIKE process checks the SPI in the delete payload with the SPIs in the SADB and can perform the SADB resynchronization by requesting the correct keys from the Keymanager. In IKE, it is recommended that the SPI number is chosen randomly. Up to our knowledge, this is primarily necessary to circumvent DoS attacks using IPsec packets with estimated SPIs. Because of the hardware implemented IPsec engine, this type of attack is not applicable to our prototype. The hardware can handle all possible Gigabit scenarios and hence the performance is not decreased. Another possibility to implement random SPIs is to allocate a separate key buffer from the Keymanager and to use the cryptographic keys as SPIs.

4 CONCLUSIONS

In this work, we presented a prototype for QKD-enabled IPsec. Up to our knowledge, it is the first prototype on such an advanced level that tries to integrate QKD in a security gateway. To serve the demanding throughput requirements of commercial applications, many components of the system are implemented in hardware. The presented key management approach acts a starting point for further integration and development approaches and verifies that high key update using QKD in IPsec are possible. Up to now, we simulated a key update rate of one update per second in software. The simulation was performed using User Mode Linux and allows the simulation of complete networks including hosts, clients and gateways on a single PC. Further tests measuring the CPU consumption and robustness against packet-loss will be performed in the near future using the final SoC prototype.

REFERENCES

Bennett, C. H. and Brassard, G. (1984). Quantum Cryptography: Public Key Distribution and Coin Tossing. In *Proceedings of International Conference on Computers, Systems and Signal Processing*.

Brassard, G. and Salvail, L. (1994). Secret key reconciliation by public discussion. *Lecture Notes in Computer Science*, 765:410–423.

D. Maughan, M. Schertler, M. S. and Turner, J. (1998). Rfc 2408: Internet security association and key management protocol (isakmp).

Diffie, W. and Hellman, M. E. (1976). New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654.

DMTF. Web-Based Enterprise Management (WBEM). website.

Frankel, S. and Herbert, H. (2003). RFC 3566: The AES-XCBC-MAC-96 Algorithm and Its Use With IPsec. RFC 3566 (Proposed Standard).

Hoffman, P. (2005). RFC 4308: Cryptographic Suites for IPsec. RFC 4308 (Proposed Standard).

Kaufman, C. (2005). RFC 4306: Internet Key Exchange (IKEv2) Protocol. RFC 4306 (Proposed Standard).

Kent, S. (2005). RFC 4303: IP Encapsulating Security Payload (ESP). RFC 4303 (Proposed Standard).

Lorünser, T., Querasser, E., Matyus, T., Peev, M., , Wolkerstorfer, J., Hutter, M., Szekely, A., , Wimberger, I., Pfaffel-Janser, C., and Neppach, A. (2008). Security Processor with Quantum Key Distribution.

Menezes, A. J., van Oorschot, P. C., and Vanstone, S. A. (1997). *Handbook of Applied Cryptography*. Series on Discrete Mathematics and its Applications. CRC Press. ISBN 0-8493-8523-7.

National Institute of Standards and Technology (NIST) (2001). FIPS-197: Advanced Encryption Standard.

Shoup, V. (1996). On fast and provably secure message authentication based on universal hashing. *Lecture Notes in Computer Science*, 1109:313–328.

Standards Based Linux Instrumentation. Small Footprint CIM Broker (SFCB) Website. website.