High Speed Quantum Key Distribution System Supports One-Time Pad Encryption of Real-Time Video[1]

Alan Mink, Xiao Tang, LiJun Ma, Tassos Nakassis, Barry Hershman, Joshua C. Bienfang, David Su, Ron Boisvert, Charles W. Clark and Carl J. Williams

National Institute of Standards and Technology, 100 Bureau Dr., Gaithersburg, MD 20899
alan.mink@nist.gov; xiao.tang@nist.gov

ABSTRACT

NIST has developed a high-speed quantum key distribution (QKD) test bed incorporating both free-space and fiber systems. These systems demonstrate a major increase in the attainable rate of QKD systems: over two orders of magnitude faster than other systems. NIST's approach to high-speed QKD is based on a synchronous model with hardware support. Practical one-time pad encryption requires high key generation rates since one bit of key is needed for each bit of data to be encrypted. A one-time pad encrypted surveillance video application was developed and serves as a demonstration of the speed, robustness and sustainability of the NIST QKD systems. We discuss our infrastructure, both hardware and software, its operation and performance along with our migration to quantum networks.

Keywords: Quantum Key Distribution, BB84/B92 protocol, reconciliation, privacy amplification, quantum networks

1. INTRODUCTION

Encryption-based security services are critical to modern telecommunications. But current encryption systems could be defeated by quantum computers or other advances in deciphering techniques. Quantum Key Distribution (QKD) systems, along with a One-Time Pad cipher [1], offer provably secure and unbreakable encryption. A properly implemented quantum key distribution (QKD) system can create a shared, secret cryptographic key over an unsecured optical link [2-4]. Such a system uses the fundamental quantum properties of single photons to guarantee the security of the shared key. A One-Time Pad cipher requires a high key generation rate for practical application since one bit of key is needed for each bit of data to be encrypted. NIST has developed a high-speed QKD [5, 6] test bed, incorporating both a free-space and fiber system. These systems demonstrate a major increase in the attainable rate of quantum key generation, over two orders of magnitude faster than other systems. The test bed provides an infrastructure for QKD metrology, research, testing, calibration and technological development.

NIST's approach to high-speed QKD is based on a synchronous model with hardware support. This has greatly improved overall key generation rate as more of the protocols, originally implemented in software, are moved to hardware. Currently the system generates sifted keys at a rate in excess of 2 Mb/s with a bit error rate of about 3%. Sifting is the first reconciliation layer required by the QKD protocol. The current system is limited by the single photon detection rate, while the hardware capacity for sifting keys is about 50 Mb/s. Reconciled and privacy amplified key are generated by software at about 1 Mb/s for the current sifted rate. Reconciliation is the second correction layer of the QKD protocol. Privacy amplification is the last layer of the QKD protocol. When the sifted key rate can be increased, the reconciliation and privacy amplification software implementation can easily be speeded-up by using parallel processing over multiple processors because of the coarse granularity of each process. Alternatively, these software algorithms can be moved to hardware.

[1] The identification of any commercial product or trade name does not imply endorsement or recommendation by the National Institute of Standards and Technology.

Quantum Information and Computation IV, edited by Eric J. Donkor, Andrew R. Pirich, Howard E. Brandt,
Proc. of SPIE Vol. 6244, 62440M, (2006) · 0277-786X/06/$15 · doi: 10.1117/12.664114

A one-time pad encrypted surveillance video application serves as a demonstration of the speed, robustness and sustainability of the NIST QKD system. We will discuss our infrastructure, both hardware and software, its operation and performance.

2. SYSTEM CONFIGURATION

The NIST QKD systems are based on polarization encoding. Their system configuration is shown in figure 1. Alice and Bob are PC-based commercial off the shelf computers using a Linux operating system. A pair of custom high-speed data handling printed circuit boards (PCBs), designed and implemented at NIST, communicate with Alice and Bob via their PCI bus. On each PCB, there is a field-programmable gate array (FPGA) and gigabit Ethernet serializers/deserializers (SerDes): one for the classical channel and four for the quantum channel. A 1.25 Gbit/s coarse wavelength division multiplexer transceiver at each end of the link is used to form the bi-directional classical channel at wavelengths closely straddling 1550 nm. Alice's board generates classical and quantum data-streams at a synchronized 1.25 GHz. Bob's board recovers and synchronizes to that clock from the received classical channel data-stream, which uses a common telecommunication 8B/10B encoding scheme [10].

Alice and Bob are also connected via a uni-directional quantum channel that is parallel to the classical channel. The quantum channel use an 850 nm wavelength in order to take advantage of high speed, high detection efficiency Si-APDs (Silicon–Avalanche PhotoDiodes, which are single photon detectors). The APDs only operate below 1,000 nm and 850 nm is a well-known telecommunication wavelength. To reduce the jitter of the APDs we convert each 800 ps electrical pulse (1.25 GHz signal) from the PCB to a 100 ps optical output pulse emitted from the multimode VCSELs (Vertical-Cavity Surface-Emitting Lasers) [11]. There is one VCSEL for each photon encoding state (i.e., 4 VCSELs for BB84). Much of the photonics are similar for both the free-space and fiber systems, but there are obviously some differences that are not discussed here but their details are discussed in [5, 6].

Our infrastructure protocol stack, which includes the PCBs and the software algorithms but not the photonics, is identical for both the free-space and fiber systems, as shown in Figure 2. The lowest layer is the hardware/firmware that handles the generation and management of the high-speed quantum data streams as well as the sifting algorithm. The next layer is the device driver within the operating system that handles scalar reads and writes to the PCB as well as DMA transfers for reading the ordered sifted bitstream from the PCB. The next layer, the first user software layer, is the reconciliation layer. This layer reads the sifted bitstream and conducts a cooperative dialog with its peer to reconcile differences between Alice and Bob. The reconciled bitstream is passed to the next layer that performs privacy amplification. Privacy amplification is a compression type algorithm that uses subsets of the reconciled data to compute a new, smaller set of data that significantly reduces any information an eavesdropper may have acquired. The privacy amplified bitstream is passed to the next layer that stores and doles it out on demand to the various applications. We discuss the PCB, the operation of the firmware on the PCB and the reconciliation layer in what follows.

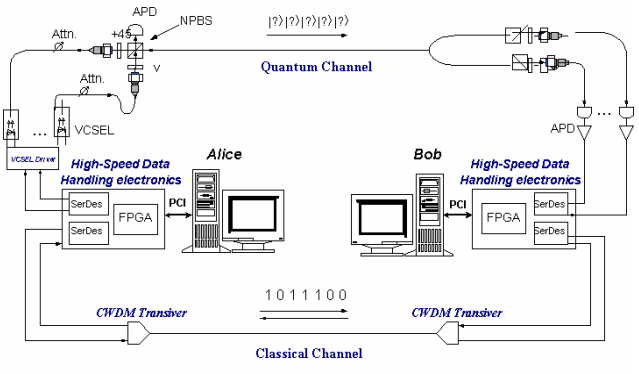

Figure 1. NIST QKD System Configuration

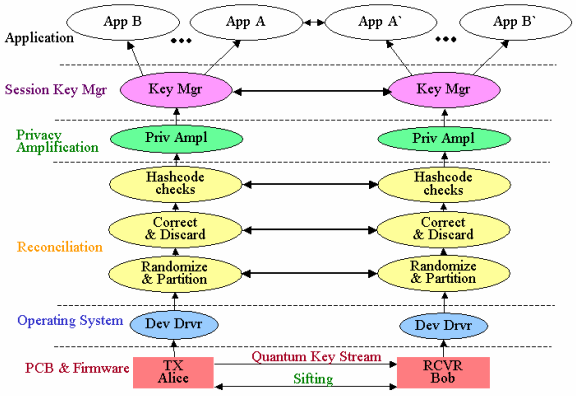

Figure 2. NIST QKD infrastructure protocol stack

3. PCB AND FIRMWARE

The major functional portions of the PCB pair are shown in figure 3. Each PCB contains an FPGA and a pair of SerDes. Each SerDes can support up to four bi-directional Gigabit channels. The FPGA exchanges 10-bit parallel data with the SerDes at 125 MHz. The SerDes converts between 10-bit parallel data at 125 MHz and serial data at 1.25 GHz. For synchronous communication, such as the classical channel, no additional support is needed. Synchronous communication requires data to be sent continuously so that the receiver can recover the transmit clock and use it to correctly extract the bits of the data stream. When there is no real data to send, predefined idle characters are sent to fill any potential absence of data. Clock recovery is an important aspect of synchronous communication because although the transmitter and receiver use similar clocks (oscillators), the clocks are not exactly the same. If an inexact clock were used to extract bits from the data stream, it would result in errors. For example a typical 125 MHz oscillator has a precision of about 10^{-5}, which is +/- 1,250 Hz from the nominal frequency. This could result in a difference of up to 2,500 clocks per second between the transmitter and receiver.

The large losses in the quantum channels result in received data being unsynchronized, sparse and random. A SerDes requires a continuous, synchronized data stream since one of its functions is to recover the clock of the received data stream. To alleviate this problem we developed special circuitry to condition and prepare the quantum data for processing by the SerDes. As the quantum signals arrive at Bob's PCB, their phase (sub-bit timing) is aligned by a programmable delay chip to a GHz clock driving a pair of flip-flops. That GHz clock is the recovered clock from the received classical channel and is identical to the transmit clock. Because of jitter, the quantum signals are not stable and so we use flip-flops to stabilize them. The stable flip-flop output is re-aligned with the replicated classical channel data stream by a second programmable delay chip and then XORed onto the replicated classical serial data stream. This XOR process "piggybacks" the sparse quantum data onto a well-formed synchronous data stream that can now be processed by the SerDes. The resultant synchronous data stream is then sent to the SerDes. Once processed by the SerDes, the parallel data is passed to the FPGA where it is again XORed with the now parallel classical data stream, leaving the original sparse quantum data stream. This dual XOR process to piggyback onto the classical data and then remove the classical data is necessary since the FPGA is not fast enough to directly sample signals at GHz rates, but the SerDes is. Part of the startup configuration procedure involves aligning the quantum channels with the classical channel, to determine the setting of the delay chips and a related delay setting within the FPGA.

To explain the operation of the FPGA firmware, we walk through the flow of the modules shown in Figure 4. The Random Number Generator module on Alice's FPGA generates two bit-streams of pseudo random data at up to 1.25 Gbit/s; one stream for the bit value and the other for the basis. Their combinations define the four quantum polarization states. Each state fires one of the VCSELs. These streams are temporarily stored in the Match Memory as well as passed to the Send Data modules where each 2048 bit pairs are grouped in a packet of state information. Each packet is then passed to the Transmit/Receive module where they are synchronously sent to Bob along with a "Sync" message on the classical channel.

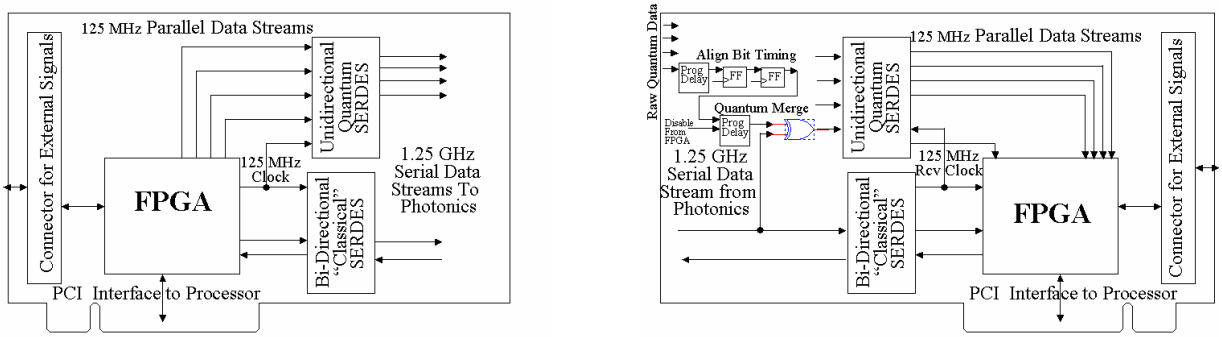

Figure 3. PCB Functional Block Diagrams of Alice (left) and Bob (right).

When a "Sync" message is received by the Transmit/Receive module in Bob's FPGA, it begins the capture of one packet's worth of data from the Quantum channels. Although the first quantum bit leaves Alice at the same time as the first bit of the "Sync" message, it can arrive sometime latter than the "Sync" message since quantum data follows a different detection path than the classical data. To accommodate this delay, we specify, via the PCI interface, the proper delay to the start of the quantum data packet. Four quantum packets are captured in parallel, one on each quantum channel and each channel has its own delay value. The four quantum packets are passed to the Recover Quantum Data module where they are aligned and then searched for rising edges that denote a detection event. The location within the quantum packet and the associated quantum channel of all detection events are passed to the Reformat & Distribute Quantum Data module. This module reformats the data into a set of triples consisting of packet location, basis and bit value. For each packet this set is temporarily stored in a FIFO and also passed to the Classical Message Control module to be sent back to Alice for sifting. The version sent back to Alice does not contain the bit values that will build the secret keys.

Sifting discards detection events when Alice and Bob's basis differs. When Alice's Receive Data module gets a packet's worth of triples, it passes that information to the Sift module. The Sift module compares the basis value for that location and packet against the correct values stored in the Quantum Data Match Memory. If they match, then the bit value stored in the Match Memory is placed in the PCI FIFO since it is missing from the triple, forming Alice's stream of ordered Sifted bits. The Sift module also sends an acknowledge list back to Bob, consisting of bit locations within the packet that had correct basis values. When Bob's Classical Message Control module gets the acknowledge list, it passes that list to the Sift module which extracts the set of triples from its Temp FIFO and discards all entries that aren't on the acknowledge list. For those items that are on the list, the bit value is place in the PCI FIFO forming Bob's stream of ordered Sifted bits. These Sifted bits are passed to an application program running on the CPU by a device driver in the operating system through a DMA (Direct Memory Access) transfer. DMA is a fast memory transfer that does not require CPU intervention, thus allowing the CPU to continue computation during the transfer.

By using an FPGA in our design we have been able to revise the operation of our system to accommodated and handle difficulties with the photonics as well as testing and configuration of our QKD system. The most notable feature has been the Test Data Memory that allows us to load any data patterns of quantum bits from an application on Alice's CPU and send them to Bob in a number of differently controlled packets. Through this feature we are able to test the network and compare observations against known data. We also use this feature to align the quantum channels by sending known data to Bob to determine the delay needed for each quantum channel. Other features that have been added include "spacing" in which we can group a number of sequential bits together that specifies a detection window. Along with spacing we can filter out portions of that window to reduce effects of jitter on the quantum channel. We can disable edge detection for a time after an edge has been detected to prevent registering an after-pulse from the detectors, a condition that could introduce additional errors and affect the security of the key. We can specify a quantum channel idle character, so that the 1's and 0's on a quantum channel remain balanced between packets. This prevents long strings of "0"s on the quantum channels and is useful in keeping the VCSEL active. We can specify a mode in which the transmission rate can be significantly reduced. This allows the boards to operate with quantum sources and detectors of much lower speeds.

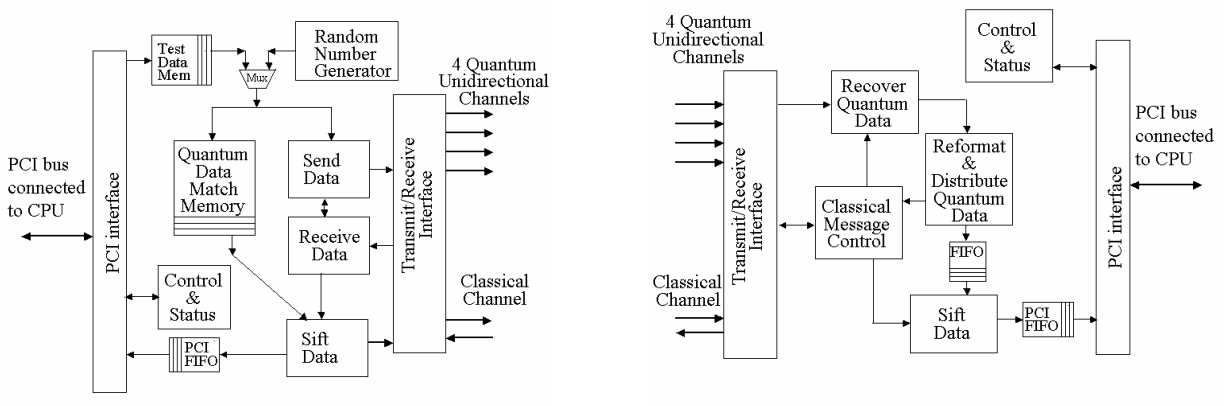

Figure 4. FPGA Functional Modules of Alice (left) and Bob (right).

4. RECONCILIATION AND PRIVACY AMPLIFICATION

To match the performance of our hardware the reconciliation and privacy amplification algorithms [8] also had to be enhanced to achieve significant performance improvements. These enhancements include treating distinct data subpopulations differently, adopting forward error correction and using large sequential chucks of data from the sifted data stream. As a result the number of iterations and round trip communication have been reduced. We distinguish three data subpopulations, corrected, uncorrected and uncorrectable. We divided each large chunk of bits into small segments (5-100 bits each) and categorize each segment as belonging to one of these three subpopulations. We consider a segment "uncorrectable" if there are three or more bits in error. Uncorrectable segments are discarded as they are discovered, thus eliminating expending effort and time to recover a small amount of bits. Forward error correction is much more efficient than parity alone, but care must be taken to use codes that don't expose too much information about the data being corrected. Large chunks of data allow a significant margin for bits to be discarded to protect any that may be exposed and still retain a significant portion of bits. We use chunk sizes in the rage of 64K to 1M bits.

The reconciliation algorithm conducts repeated passes on the data until the estimated error rate falls below a preset threshold. In each pass, both Alice and Bob randomly reorder the bits using an identically seeded pseudo-random number generator algorithm, using a new seed each time they reorder. These seeds are extracted from the previous set of privacy amplified keys, except for the first dataset, and are never exposed to eavesdropping. The reordered set of bits is divided into small segments and the parity, even or odd, for each segment is computed. Alice sends Bob her parity list to compare against his. On the first pass Bob uses the parity lists to estimate the initial dataset error rate and sends that information back to Alice. When entries in the two parity lists differ, Bob computes a Hamming error correction code on that segment and sends that list back to Alice. No correction codes are computed for segments whose parities match. Alice uses that Hamming code list to identify segments needing correction and attempts a 1-bit correction to segments where feasible, placing those segments into the corrected subpopulation, and when not feasible marks those segments as uncorrectable. Alice makes a list of the processing done on each segment. This list is also sent to Bob. Both Alice and Bob use that same list to determine which segments to keep and which segments to discard. For those segments kept, a number of bits are discarded based on the information exposed from the parity and error correction codes that were exchanged for those segments. Based on the corrections made, the remaining error rate is computed and used to determine if another pass is needed. During each pass, the corrected and uncorrected subpopulations are handled separately. For a number of passes only one of these subpopulations requires processing, further increasing efficiency. Lists are compiled in their entirety and then sent, thus increasing communication efficiency by reducing overhead. When reconciliation completes, both Alice and Bob compute a 64-bit hash on the remaining bits. If the hash codes don't match, the entire chuck is discarded. If they match, the remaining bits are then privacy amplified

Privacy amplification [7, 12] is a hash function transformation process that further reduces the amount of key in a manner that eliminates whatever an eavesdropper might know. The amount of information an eavesdropper might know is based on the sifted bit error rate determined during reconciliation and is entirely applied to potential eavesdropping, rather than distributing that error rate between eavesdropping and system losses. Both Alice and Bob perform identical privacy amplification without exchanging any information. We build a matrix, D, from the reconciled bits of dimension N x 1024, N = reconciled bits/1024. We also build a matrix, G, which consists of randomly generated 1s and 0s of dimension 1024 x S, where $S < 1024$ and depends on the sifted bit error rate. The higher the sifted bit error rate, the smaller the value of S. Both Alice and Bob generate matrix G using the same identically seeded pseudo-random number generator algorithm used in the reconciliation process. From these, we generate a privacy amplified matrix, A, of dimension N x S, whose i-th row (1..S) is obtained by XORing selected rows of D. The i-th column (1..S) of G consists of a 1024 element vector that selects which rows (1..1024) of D are to be XORed together. The computational complexity of this operation is NxN/2, i.e., of order N^2.

To increase the performance of these algorithms our implementation uses threads [9] to spawn a number of lightweight parallel tasks. Each parallel task performs the complete reconciliation and privacy amplification algorithms. A separate task accesses and parcels out the sifted bits from the PCB. Another task collects the privacy amplified bits for distribution to applications and acts as our session key manager. Each reconciliation and privacy amplification task is categorized as coarse grain computation because they require large amounts of computation. Coarse grain computation is

known to execute efficiently in a parallel processing environment and in our limited experiments on a few processors we have seen linear speedup. The performance of our infrastructure stack is shown in Figure 5. These measurements are for a dual processor system. Running on a single processor results in half the privacy amplification data rate. The left axis shows the privacy amplified key generation rate and the sifted key rate needed to sustain it as a function of the sifted key error rate. The right axis shows the percent of the sifted key that is retained after reconciliation and privacy amplification, as a function of the sifted key error rate. As the error rate increases, the privacy amplified key rate and bits retained decrease as expected, but the sifted key rate required to sustain that rate actually increases. This occurs at high error rates, because many of the bits are quickly relegated to the uncorrectable subpopulation and discarded. Also the number of bits left to be privacy amplified, an N^2 operation, is smaller.

5. ENCRYPTED VIDEO APPLICATION

A one-time pad cipher is a proven secure encryption method. It requires one bit of key for each bit of data to be encrypted. Thus for high message traffic or streaming data, a high-speed stream of encryption key bits is required. A compensation for the high key rate is the simple encryption/decryption algorithm, a bit-by-bit XOR operation of the data stream with the key stream, adding little overhead to an application. To demonstrate the performance of our QKD system we have implemented an encrypted surveillance video application, as shown in Figure 6.

This application uses a commercial web cam and a commercial media encoder and player, all of which run on standard Windows based PCs. The web cam output is processed by the media encoder and sends a UDP video data stream to our attached Alice (Linux) machine. Our encryption application, running on Alice, receives the UDP stream as well as a stream of privacy amplified keys from the local QKD Infrastructure stack and performs the one-time pad encryption on the UDP stream. The now encrypted UDP stream is sent over the internet to Bob, also a Linux machine. Our decryption application, running on Bob, receives the encrypted stream as well as the matching stream of privacy amplified keys from its local QKD Infrastructure stack. It then uses the key stream to decrypt the UDP stream and sends that UDP stream to its attached Windows based PC, which is running the media player that decodes and displays the video on the PC monitor. The result is continuous video, although delayed by a few seconds, being displayed from the web cam. Buffering through the chain of computers and processes causes the delay. The video data rates are programmable and range from 128–1,024 Kb/s, not including overhead data.

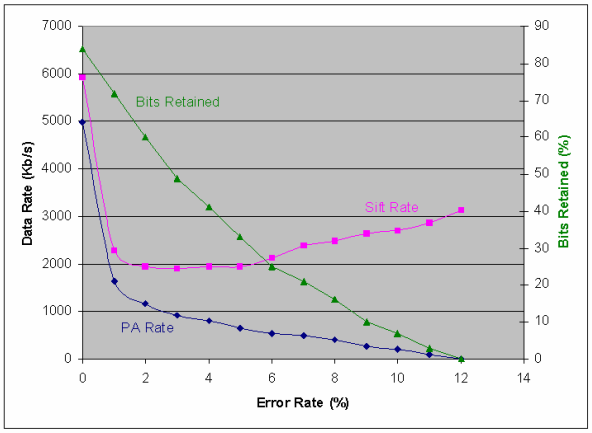

Figure 5. Infrastructure Stack Performance

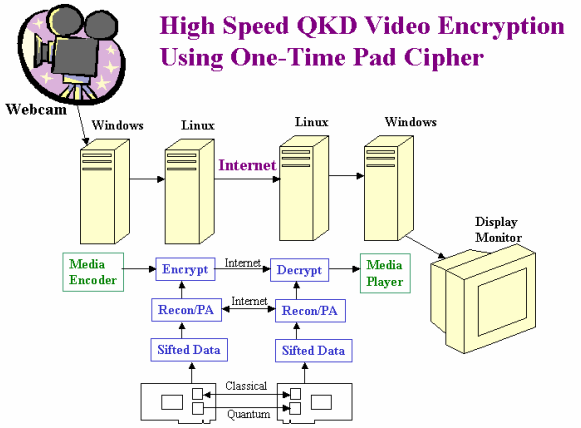

Figure 6. QKD Encrypted Video Application Setup

6. FUTURE DIRECTIONS

A robust infrastructure stack supports the NIST high-speed QKD systems. The programmability of the FPGA on the PCB allows reconfiguration of this infrastructure to support various QKD photonic networks. It currently supports both a free-space and fiber polarization based QDK system. With some slight reprogramming, it can support a planned phase-based system. As we move more of the infrastructure into hardware, we see a significant performance increase. By

moving sifting entirely into hardware, the performance capacity for that layer went from 1 Mb/s to 50 Mb/s. We are in the process of moving reconciliation and privacy amplification into hardware and anticipate a performance capacity increase from 1.6 Mb/s to 10-20 Mb/s of privacy amplified key. Although that same increase is possible through parallel processing, condensing it into a PCB makes it more compact and portable, making the goal of integration and deployment of high-speed QKD systems with standard telecommunication systems more practical. With that in mind, we are investigating quantum networking. We are developing a simple quantum LAN by linking one Alice to two Bobs through an optical switch. From there we plan to expand such a quantum LAN by adding additional nodes and also experimenting with various topologies. Along with such configurations go the development of associated protocols and how those protocols can be integrated with existing standard network protocols.

ACKNOWLEDGEMENT

This work was partially supported by the Defense Advanced Research Projects Agency under the DARPA QuIST program.

REFERENCES

1. http://en.wikipedia.org/wiki/One-time_pad, accessed Feb 2006.
2. C. H. Bennet and G. Brassard, "Quantum cryptography: Public key distribution and coin tossing," in *Proceedings of IEEE International Conference on Computers, Systems and Signal Processing* (Institute of Electrical and Electronics Engineers, Bangalore, India,1984), pp. 175-179.
3. C. H. Bennett, "Quantum cryptography using any two nonorthogonal states," Phys. Rev. Lett. 68, 3121-3124 (1992).
4. N.Gisin, G. Ribordy, W. Tittel, and H. Zbinden, "Quantum cryptography," Rev. Mod. Phys. 74, 145-195 (2002).
5. J.C. Bienfang, A. J. Gross, A. Mink, B. J. Hershman, A. Nakassis, X. Tang, R. Lum D. H. Su, C. W. Clark, "Quantum key distribution with 1.25 Gbps clock synchronization," Optics Express. 7, 2011-2016 (2004).
6. Xiao Tang, Lijun Ma, Alan Mink, Anastase Nakassis, Barry Hershman, Joshua Bienfang, Ronald F. Boisvert, Charles Clark, and Carl Williams, "High Speed Fiber-Based Quantum Key Distribution using Polarization Encoding", Proceedings of SPIE Vol.5893, Optics and Photonics Conference, San Diego, California, USA, July 31 – August 4, 2005.
7. Charles Bennett, Gilles Brassard, Claude Crepeau, and Ueli Maurer, "Generalized Privacy Amplification", Proc. 1994 IEEE International Symposium on Information Theory, pp. 350, Jun 1994
8. A. Nakassis, J. Bienfang, and C. Williams, "Expeditious reconciliation for practical quantum key distribution," Proc. of SPIE Quantum Information and Computation II, Volume 5436, Orlando Florida, 12-14 Apr 2004, pp 28-35
9. http://www.yolinux.com/TUTORIALS/LinuxTutorialPosixThreads.html, accessed Feb 2006.
10. http://en.wikipedia.org/wiki/8B10B, accessed Feb 2006.
11. http://en.wikipedia.org/wiki/VCSEL, accessed Feb 2006.
12. http://en.wikipedia.org/wiki/Quantum_cryptography, accessed Feb 2006.