Quantum Code Breaking and Key Distribution
A Brief Overview and Simulation

Evan Anderson

5/6/2016

1 Introduction

Quantum Computation has become a very interesting topic in the past few decades with its potential world changing applications. These include superpolynomial speedup of solving classical problems as well as quantum encryption. In the first section of the project, classical encryption techniques will be discussed with particular focus on the Rivest, Shamir, Adleman system (RSA), one of the most popular and famous public key distribution cryptosystem. Shor's algorithm will then be discussed in the next section as just one example of how quantum algorithms can be used to break classical cryptosystems (RSA) with a relatively small time complexity. This paper will then discuss schemes in which a quantum key distribution (QKD) system has been developed. In particular the BB-84 and E-91 protocol will be discussed. In the last section the paper will discuss a simulation written in python of the E-91 protocol. Finally a brief summary will be given along with recommendations for further improvements which could be made to the simulation.

2 Classical Encryption and Quantum Code Breaking

2.1 Classical Key Distribution Cryptosystems

The most prevalent and convenient cryptosystems today use the public key distribution method. This method works by creating a private key and subsequently a public key which has been generated from the private key. This public key can then be given to other agents to encrypt data which they wish to send to the private key holder who uses the private key to decrypt it. Some examples of these include Digital Signature Algorithm (DSA), the Secure Sockets Layer (SSL) and RSA which is prevelantly used throughout internet communications. [2]

RSA involves four steps, key generation, key distribution, encryption and decryption. Key generation involves the most steps and can be seen in the following:

Key Generation:

1. Choose two unique prime numbers p and q and compute their product $n = p * q$
2. Using Euler's Totient Function, compute $\phi(n) = \phi(p) * \phi(q) = n - (p + q) + 1$
3. Choose an integer e such that $1 < e < \phi(n)$ and is co-prime with $\phi(n)$ ($gcd(\phi(n), e) = 1$)
4. Find d where $d \equiv e^{-1}$ ($mod\ \phi(n)$)

At the end of these steps, e and n are distributed as a public key and can be used to encrypt data. d and n represent the private key and are used to decrypt messages sent which have been encrypted by the public key. p and q must also be kept private to prevent calculation of d but are not used in decryption. The encryption/decryption method is outlined below.

Encryption:

1. Armed with the public key values e and n, choose an integer m such that m is co-prime to n and represents a message. Long messages can be split into multiple m values.
2. Generate a ciphertext c where $c \equiv m^e$ ($mod\ n$)
3. Send c to the owner of the private key

Decryption:

1. Receive ciphertext c
2. Armed with private key integers d and n, calculate m by solving $c^d \equiv (m^e)^d \equiv m$ ($mod\ n$)
3. Read message M from integer m

While certain details such as finding large prime numbers p and q or choosing an appropriate integer m have been left out, the overall process is not terribly complicated. RSA thus works on the premise that it is very easy to find three large numbers e, n, and subsequently d such that for all m the act of encrypting and decrypting works. Additionally, d which is the basis of the private key is computationally hard to find given n, e and even m.

d is hard to find as it would involve factoring n to find the prime factors p and q to calculate d (see step 1-4 in key generation). In fact, the fastest known classical algorithm

to factor large numbers is the general number field sieve which has time complexity of $O(exp(1.9 * Log(n)^{\frac{1}{3}} * Log(Log(n))^{\frac{2}{3}})$. Table 1 shows how many steps would be required and the subsequent time it would take to factor given one operation per nanosecond (1 GHz processor).

n	$O(exp(1.9 * Log(n)^{\frac{1}{3}} * Log(Log(n))^{\frac{2}{3}})$	time to factor
10^2	66	< 1 sec
10^{50}	$2.1 * 10^{11}$	217 sec
10^{100}	$4.4 * 10^{15}$	50 days
10^{200}	$2.2 * 10^{21}$	70774 years
10^{600}	$2.3 * 10^{34}$	$7.4 * 10^{17}$ years

Table 1. Execution time of the best classical algorithm for factoring large integers assuming one operation per ns.

It should be noted that 1 GHz is relatively slow, however it is only an order of magnitude 6 slower than today's super computers. This would translate to roughly $7.4 * 10^{11}$ years for 600 digit numbers when ran on a super compter. Today, RSA utilizes 2048 bits which can indeed represent decimal numbers with over 600 digits. [1, 8] As can be seen, the power of RSA (and any feasible classical cryptosystem) lies directly in the inability to crack them in a reasonable amount of time. Quantum algorithms however have superpolynomial speedup and can crack such systems with ease.

2.2 Shor's Algorithm

Shor's Algorithm, created in 1994 by Peter Shor, is one of the most notable algorithms in quantum computing. It is what helped launch QC from being a more obscure sub field in computer science and mathematics to a subject of more wide scale interest. As mentioned previously, Shor's algorithm is able to factor large numbers and subsequently crack RSA with a relatively small time complexity. This means if a decent sized (number of qubits) quantum computer is realized, then all encrypted messaged and files become available to the owner of said quantum computer.

Shor's algorithm contains a number of steps and can be broken up into classical and quantum components. The classical part is as follows:

Classical steps:

1. Start with a number n you wish to factor
2. If n is even, then the prime factor 2 exists
3. If n is odd, choose a random positive integer a ¡ n

4. Find $gcd(n, a)$ using Euclid's Algorithm
5. If n and a are not co-prime $gcd(n, a) \neq 1$ then a is a non-trivial factor
6. If n and a are co-prime, then run the quantum portion of the algorithm to find r
7. If r is odd or $a^{\frac{r}{2}} \equiv -1 \; mod(n)$, start from step 1 with a new value for a
8. Otherwise, $gcd(a^{\frac{r}{2}} - 1, n)$ and $gcd(a^{\frac{r}{2}} + 1, n)$ are non trivial factors of n

The quantum portion of the algorithm is a sub-routine of the classical portion.

Quantum steps (step 6 in the classical portion):

1. Choose numbers Q and q_A where $Q = 2^{q_A}$ and $n^2 \leq Q < 2n^2$
2. Choose a random integer x that is co-prime to n
3. Setup two registers A and B both with all their qubits set to 0. This gives the initial state
   $|\psi_0\rangle = |0\rangle_A |0\rangle_B$
4. Apply the Hadamard operation to each qubit in register A, creating an equally probable superpostion of all possible interger values (0 to $Q - 1$). The state is now
   $|\psi_1\rangle = \frac{1}{\sqrt{2^{q_a}}} \sum_{j=0}^{2^{q_a}-1} |j\rangle_A |0\rangle_B$
5. Apply the transformation U_x such that the resulting state becomes
   $|\psi_2\rangle = \frac{1}{\sqrt{2^{q_a}}} \sum_{j=0}^{2^{q_a}-1} |j\rangle_A |x^j mod\ n\rangle_B$ (See Appendix A, problem 6.6 to see how these are equivalent minus a global phase factor)
6. Measure the state of register B such that it collapses to some $|x^{b_0} mod\ n\rangle_B$ and leaves register A in a superposition of values. The $|j\rangle_A$ values will be seperated by an integer amount as a result of collapsing B and thus can be written in the form $|ar + b_0\rangle$. Thus leaving state
   $|\psi_3\rangle = \frac{1}{\sqrt{2^{q_a}/r}} (\sum_{a=0}^{\frac{2^{q_a}}{r}-1} |ar + b_0\rangle_A) |x^{b_0} mod\ n\rangle_B$
7. Compute the inverse quantum fourier transform (QFT^{-1}) of $|\psi_3\rangle$ leaving the state
   $|\psi_4\rangle = \frac{1}{\sqrt{r}} (\sum_{k=0}^{r-1} exp(-2\pi i \frac{k}{r} b_0 \, |\frac{k * 2^{q_A}}{r}\rangle_A) |x^{b_0} mod\ n\rangle_B$
8. Measure register A, collapsing it to some $|\frac{k * 2^{q_A}}{r}\rangle_A$ where $0 \leq k \leq r - 1$
9. Rerun steps 1-8 $O(Log(Q))$ times to find r and return to the classical portion of the algorithm

Step 9 needs slightly further explanation. There are 3 cases for finding r that the final $|\frac{k*2^{q_A}}{r}\rangle_A$ can collapse to. Case one is trivial, when $k = 0$, $\frac{k*2^{q_A}}{r} = 0$ and nothing can be learned about r. In the second case, if k and r are co-prime, then the fraction is in its smallest form and r is directly known. The third case is when k and r are not co-prime and thus reduces further (e.g. $\frac{2}{4} \rightarrow \frac{1}{2}$). Continuing to run the quantum portion of the algorithm however will reveal the true r. Additionally, some more work needs to be done if r is not a factor of 2, namely the value of $|\psi_4\rangle$ in step 7 is not entirely correct. A continued fraction trick can be induced to solve this issue and is explained in chapter 6 of Explorations in Quantum Computing by Colin P. Williams. To illustrate the quantum portion of the algorithm, an example where $n = 15$ has been worked in Appendix A, problem 6.4.

The punchline of Shor's algorithm is that it has a polynomial in $Log(n)$ time complexity, specifically $O(Log(n)^2 Log(Log(N))Log(Log(Log(N))))$. Table 2 compares this with values from the best known classical algorithm for factoring.

n	Shor's t.c.	Shor's t.t.f.	Classical t.c.	Classical's t.t.f.
10^2	14	< 1 sec	66	< 1 sec
10^{50}	97967	< 1 sec	$2.1 * 10^{11}$	217 sec
10^{100}	$4.9 * 10^5$	< 1 sec	$4.4 * 10^{15}$	50 days
10^{200}	$2.4 * 10^6$	< 1 sec	$2.2 * 10^{21}$	70774 years
10^{600}	$2.7 * 10^7$	< 1 sec	$2.3 * 10^{34}$	$7.4 * 10^{17}$ years
$10^{1000000}$	$2.1 * 10^{14}$	2 days	$2.3 * 10^{652}$	$7.4 * 10^{635}$ years

Table 2. Comparison of quantum and classical number factoring algorithms for decimal numbers of various lengths. Time complexity (t.c.) is calculated and time to factor (t.t.f.) assumes one operation per ns.

As seen from table 2, Shor's algorithm can factor extremely large numbers very quickly. All current RSA encryption can in fact be cracked in under a second given a quantum computer running Shor's algorithm at 1 GHz. It isn't until numbers such as $10^{1000000}$ where Shor's algorithm begins to take a noticable amount of time. Increasing the processing speed solves this problem. Fortunately, quantum physics also allows for quantum cryptosystems which are not vulnerable to such algorithms, or any known algorithm for that matter! [2, 9, 3]

3 Quantum Key Distribution

Before talking about QKD protocols, it is first important to understand the concept of a one-time pad (OTP). An OTP is an unconditionally secure cryptosystem. An OTP works by two clients having the same private key which is used to encrypt and decrypt a single cipher. The cipher itself has no information about the private keys and is therefore meaningless to anyone who intercepts it. The private key is then discarded once used to

encrypt and decrypt a cipher. The primary drawbacks of an OTP system are that the private key itself must be transmitted securely in the first place and secondly, the private keys used must be at least as long as the cipher itself and in the end is discarded. This forces new private keys to be generated and exchanged frequently. Exchanging the keys in a secure manner is not an easy task, for example Alice physically giving Bob a hard drive, other communication channels are not inherently secure. Due to these drawbacks, an OTP system is very impractical to use classically.

QKD solves both the primary drawbacks associated with classic OTP systems. QKD systems are able to ensure that an exchange of a private key via a possibly insecure channel is secure. While the keys themselves are still consumed at the same rate, not having to physically exchange keys via a secure channel removes this as a problem. QKD additionally solves the problem of being able to generate true random keys as opposed to pseudo-random classically, but this is not considered a major problem. [2]

3.1 BB-84 Protocol

The BB-84 QKD protocol is named after Charles H. Bennett and Gilles Brassard who created the protocol in 1984. This protocol works by capitalizing on the polarization states of single photons. The polarization bases that are used are $\{|\updownarrow\rangle, |\leftrightarrow\rangle\}$ and $\{|\nwarrow\rangle, |\nearrow\rangle\}$ and are related such that $|\nwarrow\rangle = \frac{1}{\sqrt{2}}(|\updownarrow\rangle + |\leftrightarrow\rangle$ and $|\nearrow\rangle = \frac{1}{\sqrt{2}}(|\updownarrow\rangle - |\leftrightarrow\rangle$. With these bases in mind, the process of encryption by Alice and decryption by Bob is linear:

1. Alice generates a long list of truly random bits.
2. Alice then encodes the bits as polarization in photons. The polarization of a photon in a given basis determines its value and is agreed upon before hand. For example $|\nwarrow\rangle$ and $|\updownarrow\rangle$ represent 0 in their respective basis and $|\nearrow\rangle$ and $|\leftrightarrow\rangle$ represent 1. Alice encodes these in a given basis with equal probability and sends them to Bob.
3. Bob receives the photos and chooses a basis for each one and measures them. If Bob chose the same basis as Alice, he knows that it represents a 0 or 1. If he chose the incorrect basis, he only knows with 50
4. Bob then tells Alice the basis for which he measured each bit but not the outcome (0, 1 or unknown).
5. Alice then relays to Bob which bits they used the same basis for.
6. Alice and Bob discard all bits in which they used different bases and now have matching random bits which can be used as a private key.

At the end of step 7, Alice and Bob have a private key that was transmitted via a potentially unsecure channel. They work under the assumption that their bits match.

However, there could have be an eavesdropper, Eve, who while trying to read the bits being sent from Alice changes the polarization of the photon and thus the bit Bob received. The BB84 protocol protects against this in the following way:

1. Eve intercepts a photon from Alice and chooses a random basis (much like Bob does) to measure it.
2. Eve must then re-encode (she does not know the encoding Alice used) the bits she sends on to Bob
3. Bob proceeds to make measurements
4. In order to check if Eve exists, Bob and Alice agree to publicly announce a subset of their bits along with the polarizations
5. Eve is caught if Alice encoded in one basis, Eve re-encoded in the second basis and and Bob measures in Alice's basis but the bits do not agree.

As an example of step 5 - Alice encodes a 0 as $|\nwarrow\rangle$, Eve receives that the re-encodes in the $\{|\updownarrow\rangle, |\leftrightarrow\rangle\}$ basis. Bob then measures in the $\{|\updownarrow\rangle, |\leftrightarrow\rangle\}$ basis and gets a 1. Alice and Bob then tell each other, of the subset of agreeing polarizations and bits, what the values of said bits are. They notice a discrepancy on the aforementioned bit - Alice encoded a 0 but Bob measured a 1 using the same basis. This alerts them that Eve exists and they can abandon this set of bits and try again until no evidence of Eve exists.

It is important to note that had Eve chosen the same basis as Alice and Bob, she would have gone undetected for this particular bit. Therefore, if Bob, Alice, and Eve use the optimal strategy of randomly picking an encoding basis, the probability of Eve being detected is $\frac{1}{2}$(Alice encodes correctly) $*$ $\frac{1}{2}$(Bob measures correctly) $= \frac{1}{4}$. The percentage of catching Eve is then $1 - (\frac{3}{4})^N$ where N is the number of bits in which Bob and Alice have the same polarization and agreed to announce.

It can be seen then, using the BB84 protocol that a private key can be generated and transmitted through a potentially unsecure communication channel while detecting if anyone has tampered with the bits being transmitted. [2, 4]

3.2 E91 Protocol

The E91 protocol was developed in 1991 by Artur K. Ekert and utilizes Bell's theorem which states "No physical theory of local hidden variables can ever reproduce all of the predictions of quantum mechanics." The E91 protocol works very similar to the BB84 protocol:

1. An entangled pair of spin-$\frac{1}{2}$ particles are generated in the singlet state ($\frac{1}{\sqrt{2}}(|\uparrow\downarrow\rangle - |\downarrow\uparrow\rangle)$). Alice can be the one who generates this, or some source between Alice and Bob. A member of each pair of particles is sent to either Alice or Bob along some axis, call it the z-axis.

2. Alice and Bob choose a measurement axis perpendicular to the z-axis to measure their with varying azimuthal angles from the vertical x-axis. Alice and Bob must choose an analyzer along one of three axis $a_i, b_j (i = 1, 2, 3)$ respectively where $\phi_1^a = 0$, $\phi_2^a = \frac{\pi}{4}$, $\phi_3^a = \frac{\pi}{2}$, $\phi_1^b = \frac{\pi}{4}$, $\phi_2^b = \frac{\pi}{2}$ and $\phi_3^b = \frac{3\pi}{4}$. Measurement thus reveals a $+1$ (spin-up, 1) or -1 (spin-down, 0) which they record.

3. Alice and Bob announce publicly the orientations of their analyzers. These will fall into three groups. The first is where Alice or Bob did not register a particle and is dismissed. The second is where they used the same orientation (e.g. $\frac{\pi}{2} = a_3, b_2$), and the third where they used different orientations. In the case that the same orientation was used, due to entanglement they should receive directly anti-correlated values. For example, Alice measuring $+1$ (spin-up), forces Bob to measure -1 (spin-down) provided they are along the same axis.

4. Alice and Bob discard all entries in which the axis do not match then Bob flips all his bits which will be guaranteed to match Alice's due to their anti-correlation. They now have a matching secure private key which was transmitted via an unsecure channel.

5. To detect a potential eavesdropper Eve, Alice and Bob can use their non-matching analyzer results. Consider the correlation coefficient $E(a_i, b_j) = P_{++}(a_i, b_j) + P_{--}(a_i, b_j) - P_{+-}(a_i, b_j) - P_{-+}(a_i, b_j)$ where, for example, $P_{++}(a_i, b_j)$ is the probability of both Alice and Bob measuring $+1$ (spin-up) given analyzer orientations a_i and b_j. Then the discarded results from Alice and Bob form $S = E(a_1, b_1) - E(a_1, b_3) + E(a_3, b_1) + E(a_3, b_3)$. If both Alice and Bob announce their results, due to quantum mechanics more specifically Bell's Theorem and entanglement, one should obtain $S = -2\sqrt{2}$. However, if Eve has tried to intercept and detect the particles, it will break the entanglement of the particles, disturbing the state of a given particle and subsequently Alice and Bob will find $S \neq -2\sqrt{2}$, proving Eve's existence. [2, 5]

4 QKD Simulation

The code for simulating the E91 protocol was written in python and can be found in Appendix B along with sample outputs in Appendix C. It can also be found on github at `https://github.com/kidfiction/e91-simulator`. The simulation relies on the commonly available NumPy (numpy)[7] package as well as the Quantum Information Toolkit (qit)[6]. The numpy package enabled use of various mathematical functions and tools such

as arrays, values for Pi and cos/sin functions. The qit packaged was used to help simulate and generate quantum states. It was also used to perform basic unitary operations on the various simulated quantum states.

The simulation itself accepts 3 parameters. The firs is the number of entangled particles should be generated. The second is the detection tolerance to find Eve. That is to say, at what measured S value, compared to the expected $-2\sqrt{2}$ should the program throw an alert that Eve has been detected. The final parameter adjusts the rate at which Eve intercepts a particle. While in theory, Eve only intercepting a subset of particles doesn't give her full access to the final key, it is fun to vary and see how many particles she must measure in order for Alice and Bob to detect her presence.

Once the simulation has concluded, the program outputs various statistics. These include the observed S and its relative deviation from $-2\sqrt{2}$. Subsequently if the deviation is higher than the tolerance input, it alerts that Eve has been detected. It also outputs the number of particles Eve intercepted and the ratio of them in which she chose the correct polarization to match Alice. Finally it outputs the length of the generated private key, and if debugging is turned out it will output said private key. This statistic is output regardless of the detection of Eve.

5 Discussion

Feasible classical encryption works on the an a-symmetric basis where a single owner of a private key generates public keys to distribute. The private keys can then be used to encrypt messages sent to the owner of the private key who uses it to decrypt the cipher. These encryption schemes such as RSA work on the basis that cracking them would classically take a very long time. RSA in particular works under the assumption that factoring very large numbers takes exponential time complexity. However, quantum algorithms have been shown to be able break these encryption schemes in polynomial time. This would essentially make all private communications no longer private.

Fortunately, if a quantum computer running such an algorithm is ever realized, it is very likely that a quantum key distribution cryptosystem would also be developed. These systems work slightly differently than the classical a-symmetric ones. Instead they use a one time pad in which all parties share a private key. This is not feasible classically as there is no realistic way to transmit the private key securely and frequently. The BB84 and E91 protocols however, using properties of quantum mechanics are able to transmit private keys securely and detect if anyone is attempting to eavesdrop.

A simulation of the E91 protocol was created and demonstrates the basic principles behind a quantum key distribution system including the key distribution between Alice and Bob and how they would go about detecting Eve. There are some improvements that can be made. Two of the main ones are error detection and privacy amplification which were outside the scope of this project. Error detection has to deal with the fact

that maintaining a maximally entangled state is not easy and perturbations can ultimately effect the state, much like that of Eve doing a measurement. Similarly, privacy amplification deals with taking the generated private keys and modifying them even more to become more secure. Unfortunately the program is still just a simulation; the particles are not really entangled and nothing is truly random. Quantum effects are mimicked by a probabilistic state machine. One must simply wait for the real thing. I hope to see everything discussed within this project in my lifetime.

References

[1] Bennett, C., and Brassard, G., "Quantum cryptography: Public key distribution and coin tossing". (reprint) Theoretical Computer Science **560**, 7-11 (2014).

[2] Ekert, A. K., "Quantum Cryptography Based on Bell's Theorem". Phys. Rev. Let. **67**, 6 (1991).

[3] Rivest, R. L., Shamir, A., and Adleman, L., "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", Commun. ACM **26**, 96-99 (1983)

[4] Shor, P., "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer", Siam J. Comput. **26**(5), 1484-1509 (1997).

[5] Website for the Quantum Information Toolkit `http://qit.sourceforge.net/`

[6] Website for the NumPy python package http://www.numpy.org/

[7] Wikipedia entry for RSA `https://en.wikipedia.org/wiki/RSA`

[8] Wikipedia entry for Shor's Algorithm `https://en.wikipedia.org/wiki/Shor%27s_algorithm`

[9] Williams, C. P., "Explorations in Quantum Computing". Springer-Verlag London, 2011