Hacking the BB84 Quantum Key Distribution Protocol

Ryan Torok

March 4, 2018

1 The Target Protocol Implementation

Our eavesdropping implementation attempts to attack the provided key distribution protocol, which was implemented as follows:

1. Both party members wishing to send a secret message know beforehand how many bits the sender intends to send over the quantum channel to generate a key (this was simulated using the `config` file).
2. The sender (call her Alice) generates a random binary string to represent the secret key, and another of equal length to represent which basis each bit of the key will be sent in through the quantum channel.
3. Alice sends three copies of each bit of the key as photons over the quantum channel in the same polarization.
4. The recipient (call him Bob) generates his own random binary string, representing which basis he will measure each of the incoming sets of three photons. Bob measures these photons in his random bases, and records his version of the key, which is correct for all the indices in which he and Alice chose the same basis, except in cases of channel error.
5. Both Alice and Bob publicly announce the bases they chose over the classical channel. Each compares the two basis strings and throw out the bits of their keys where the bases at those indices do not match (sifting).
6. Alice generates a security string consisting of every fourth character of the key beginning with the first character, repeated thrice each. She sends this string to Bob, who will generate a string to match from his original sifted key, which still contained the repeated bits to represent one bit of the key. Bob sends this string to Alice, and both parties compare the two strings to determine whether an eavesdropper is present.
7. If the error rate exceeds 0.36, then the parties conclude an eavesdropper is present, and the user may choose to manually exit or continue with the protocol anyway.

8. If there is no eavesdropper detected, Alice generates the final key by removing every fourth character (the security bits) from her original sifted key. Bob generates the key by taking the average of each set of three bits to minimize the effect of error rate in the channel, and throws out the sets of security bits in the same manner as Alice.

9. Now, Alice and Bob should have the same key to use for sending a secret message.

A few notable strengths were present in the protocol, namely that the exchanging of photons is the first action performed by either party, preventing an eavesdropper the ability to extrapolate any information which may help him or her decide on a measurement basis pattern which will yield results better than blindly guessing. Additionally, the fact that the security pattern is based on the positioning of bits after the keys are sifted prevent an eavesdropper from simply avoiding measuring every n-th bit, as it is impossible to know exactly which bits will be sifted before the bases are announced.

However, the BB84 implementation possessed several key weaknesses the eavesdropping implementation used to construct the key, usually undetected. Firstly, though the positioning of the security bits at the original exchange of photons is not certain, it is still predictable, especially at the beginning of the key. If the implementation had set a more stringent threshold for the error rate, the eavesdropping implementation would have had to put a greater deal of effort into deciding systematically which sets of bits to measure. However, another key weakness in the implementation is the fact that the acceptable error rate is so high, allowing much leeway in what bits an eavesdropper may measure (and potentially modify) and still escape detection.

The most critical error in the provided implementation, though, is the fact that each bit is sent more than once across the quantum channel. The entire BB84 protocol is reliant on the fact that an eavesdropper cannot clone photons to measure as to not disrupt the ones being sent through the channel. However, because the eavesdropper has multiple attempts to guess the correct basis, an eavesdropper is able to generate the key with 100% accuracy by measuring the different photons corresponding to the same bit in different bases.

2 The Eavesdropping Protocol

With the ability to exploit the issue of the repeated bits, we may use a systematic approach to intercepting the photons in order to guarantee that we do not break the security threshold. Since each bit is repeated thrice in the channel, and over 1/3 of the bits must be incorrect to cause a security warning, the eavesdropper (call her Eve) may do the following with no chance of detection:

1. Measure the first photon for each bit sent by Alice in the Horizontal/Vertical basis, and send the (possibly collapsed) photon to Bob.

2. Measure the second photon for each bit sent by Alice in the Diagonal/Anti-diagonal basis and send to Bob.

3. Allow the third photon for each bit to be sent to Bob without measuring.

This process guarantees Eve will disrupt exactly 1/3 of the photons, as she will chose the wrong basis to measure on exactly one of the two measurements for each photon. Because she does not measure the third photon, she guarantees that Bob will receive the same bit Alice sent (barring any channel error), and thus it is guaranteed that we have disrupted 1/3 of the bits. For each of the disrupted photons, there is a 50% chance that Bob will randomly generate the opposite polarization Alice sent (as Bob must have chosen the same basis as Alice, opposite to Eve's, for the bit to pass the sifting process). As a result, we expect to artificially introduce an error rate of 1/6, or 16.7% into the protocol, which, at the provided parameter of 120 bits, allowed Eve to generate the entire key, barring some channel error, and went undetected on 12 out of 15 trials.

To actually extrapolate the key after intercepting the photons, Eve looks at the bases Alice used to send the bits of the key. If she used the Horizontal/Vertical, basis, Eve appends the result from the first measurement of the corresponding photon set to her key. If Alice chose the Diagonal/Anti-diagonal basis, Eve appends the second photon's measurement. Next, Eve compares the bases used by Alice and Bob to determine which bits should be sifted from her key (as they will be thrown out by Alice and Bob). After sifting, Eve removes every fourth bit from her key, beginning with the first bit, to remove the bits Alice and Bob used for the security check. Eve now has a copy of Alice and Bob's exchanged key which is certainly correct at every position barring any channel error.

If we are able to assume Eve has a perfectly working photon detector, we are almost always able to generate the message, after error correction using three bits of the key for each bit of letters' ASCII codes, with few or no errors. In 15 trials using 500 bits (after throwing out trials where Eve was detected), we obtained an average error rate of roughly 10%, which is likely the result of channel noise represented by the less than perfect `sendPhoton()` method, which allows for a small chance of error when Alice sends a photon. Overall, we are satisfied with our Eve implementation, able to obtain a near-perfect, if not perfect result of an encrypted message with a very high likelihood of doing so undetected.