Distributed Quantum Security: Extending the Security Proof of Quantum Key Distribution to Hold Across a Friendly Sink-Free Network

Alexis Goodfellow

November 30, 2019

Abstract

In the ideal case, Quantum Key Distribution (QKD) protocols such as the BB84 protocol and E91 Protocol ensure perfectly secure encryption across a point-to-point link. However, one of the key difficulties in the implementation of these protocols as they stand is the inability to use signal repeaters due to the fact that quantum information is destroyed on observation. This paper proposes an algorithm for bootstrapping the perfectly secure encryption provided by point-to-point QKD protocols to hold across a friendly, sink-free QKD network topology by concurrently distributing multiple encryption keys across the network.

1 Definitions

Before attempting to discuss the proof itself, it is critical to clearly define some key terminology used within it, particularly the precise meaning of the terms "friendly" and "sink-free".

In the context of this paper, a "friendly" intermediate node is one who is known to be under the maintenance and control of a good actor. Informally, this means a "friendly" node can be viewed as purely an informational thoroughfare. More formally, this means that all "friendly" nodes meet two fundamental criteria; the first is that they do not store any record of communicated messages that are known to have been successfully received, and the second is that they do not communicate any messages to anyone other than their intended recipients.

The next important term to define is the concept of a "sink node". A "sink node" exists if it is the case that for all possible traversals of the network, there exists an intermediate node that all transmissions must pass through on the path from a given source node s to a given receiving node r. In order for the following security proof to hold, the QKD network must not contain a "sink node", and thus must be "sink-free".

The last important concept to define is the concept of "path independence". Simply put, two paths are independent if and only if they share no intermediate nodes and only share the terminal nodes s and r.

Given a sink-free network of friendly nodes, the following security proof holds.

2 Example Networks

To get a better intuitive understanding of what exactly these friendly, sink-free networks look like, it's worthwhile to demonstrate these properties with some simple examples that illustrate some example networks satisfying zero, one, and all of these properties. The most simplistic example is one without any intermediate nodes at all, where the sender s is directly connected to receiver r[1].

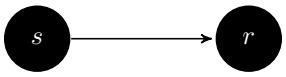

Figure 1: No intermediates at all

Though this case is the simplest possible example satisfying the two constraints, the security of this example is already assured by the point-to-point protocols that already exist since there are no intermediate nodes. For the purposes of this paper, it is an uninteresting example.

[1]The unidirectional arrows in these graphs are solely for illustrating the flow of information through the network. Each link itself is always bidirectional.

The simplest possible example of a graph with an intermediate node is also an example of a graph with a sink node. The next figure shows that graph, where s and r are the same sending and receiving nodes as before, and all nodes labeled with i are intermediate nodes.

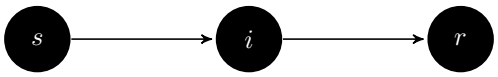

Figure 2: A sink node exists on a path from s to r

It is trivial to see that i meets the definition of a sink node. Let us consider the same example network topology in a different scenario, where the node marked with x is an extraneous node not on the path from s to r

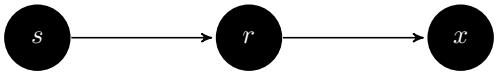

Figure 3: No sink node exists on a path from s to r

From this trivial example, we can see that for the same topology, a sink node can exist or not dependant upon the nodes chosen as s and r. This is important to the design of the network topology because it means that in order for any given node in the network to be able to securely communicate with any other given node in the network, there cannot be a single sink node between any two nodes in the network. This is equivalent to stating that *every* node pair on the network must be sink-free when chosen as s and r, which is a much stronger precondition than merely a *single* node pair being sink-free.

To see why the restriction of friendliness must hold for each intermediate node, consider the following network topology:

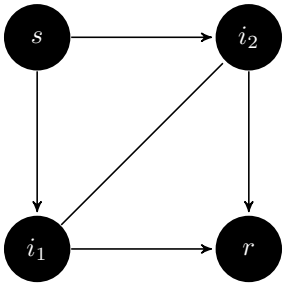

Figure 4: An unfriendly pair of intermediates

In this scenario, s knows that both i_1 and i_2 have connections to r, but s is unaware that i_1 and i_2 secretly have connections to each other. In this scenario, s could naively partition the secret message destined for r between two different transmissions, where the first transmission is routed through i_1 and the second

is routed through i_2. However, this partitioning would not ensure the security of the secret message due to the fact that i_1 and i_2 could communicate with each other across their secret link and synthesize the original message intended for r. This connection between i_1 and i_2 makes them both "unfriendly" nodes. This same topology can be made "friendly" by simply removing the connection between i_1 and i_2.

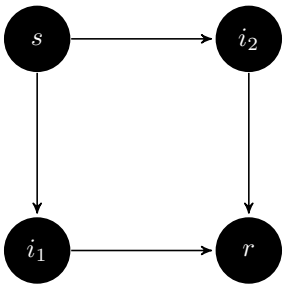

Figure 5: A friendly pair of intermediates

With these simple examples in mind to illustrate the definitions, the algorithm and security proof can now be presented on the assumption that the QKD network in question is both friendly and sink-free.

3 Algorithm

3.1 Part 1 - Communicating the Keys

1. Let l be the length of the message m that s wants to send to r.
2. Let $P = \{p_1, p_2, ..., p_n\}$ be the set of n independent paths between the two terminal nodes s and t.
3. For each path $p \in P$:
   (a) Let $C = \{c_1, c_2, ..., c_i\}$ be the set of point-to-point connections between each node in the path p.
   (b) Generate a unique random key r with length l
   (c) For each $c \in C$:
      i. Use a point-to-point QKD protocol to securely transmit the random key r to the next host.
      ii. Use a point-to-point QKD protocol to securely transmit the ordered list of nodes o traversed through the network so far.
      iii. Use a point-to-point QKD protocol to securely send the result of applying a mutually-agreed-upon cryptographic hash function h applied to o.

At the end of this process, both s and r are in possession of the same set of n unique encryption keys, but only r is in possession of the hashes and transmission history of each path taken through the network. This information needs to be communicated back to s in order for s to be informed that r is ready to receive the encrypted message.

3.2 Part 2 - Alerting the Sender

Though the friendly nodes do not store the *content* of the messages they transmit, they must see the source and destination addresses of the content, and they are allowed to *temporarily* store the triple (s, r, p), where s and r are the addresses of the sender and receiver pair and p is the address of the previous sender in the chain.

1. Let the triple (k_n, p_n, h_n) correspond to the key, path, and hash of a given path in P.
2. For each such triple with index n in P:
   (a) Let $m = (n-1)\ mod\ n$
   (b) Let x be the binary encoding of p_m in reverse order
   (c) For each point-to-point link in p_m until s is reached
      i. Use a point-to-point QKD protocol to securely transmit the hash h_n
      ii. Use a point-to-point QKD protocol to securely transmit the ordered list of nodes o traversed through the network so far on the path back from r.

After this process is complete and the authenticity of each path and connection is verified, the message itself must be encrypted by s, sent across the network from s to r, and decrypted by r.

3.3 Part 3 - Encrypting and Decrypting the message

1. Let K be the set of n random keys synthesized in Part 1
2. Let $\oplus$ be the bitwise exclusive or operation
3. Let the encrypted message $e = m \oplus k_1 \oplus k_2 \oplus ... \oplus k_n$
4. s sends e to r across the network on some path $p \in P$ from Part 1
5. r performs $x = e \oplus k_1 \oplus k_2 \oplus ... \oplus k_n$

Since it must be the case that $x = m$ due to the fact that the bitwise exclusive or operation is its own inverse operation, the last decryption step (5) in Part 3 is provably successful at undoing the encryption step (3) in Part 3.

4 Security Proof

4.1 Part 1 - Key Communication and Utilization

Due to the sink-free restriction, the maximum possible number of independent paths between the two terminal nodes s and t is *guaranteed* to be greater than two. A direct result of the existence of multiple paths is the existence of multiple keys.

Since each intermediate node in the network is friendly, no messages are ever stored or sent to a node other than the intended next-hop recipient. This friendliness restriction assures that the keys can't be communicated between any pair of nodes not on the same path, and path independence guarantees that no node exists in multiple paths. Taken together, these facts mean that any node besides s and r possess exactly one key, which is insufficient to fully decrypt the encoded e message that is guaranteed to use more than one key.

4.2 Part 2 - Extra Security Through Alerting the Sender

Sending back the hash of each path on the next path in the series assures two different conditions that have important security implications. Firstly, it assures accurate history preservation, since the hash of a path can be trivially checked against the binary encoding of that path for discrepancies. Secondly, assuring that the hashes are never communicated on the same path that generated them prevents against man-in-the-middle style attacks.

Regardless of algorithm, it is necessary for r to send some sort of message to s indicating that r is now ready to receive a new message. This proposed authentication algorithm has the advantage of preserving the important distinguishing property of QKD protocols that make them superior to classical encryption; the sender can determine the presence of eavesdroppers on the network during the key generation process, before any sensitive data is even sent across the network.

5 Conclusion

This security proof is sufficient to prove the impenetrable security of the proposed protocol across an entire friendly, sink-free QKD network across any number of intermediate nodes. However, the proof comes with some very strict restrictions on both the topology of the network and the behavior of the intermediate nodes. The sink-free restriction of the network topology is simply impossible to overcome due to the fact that a sink node will always possess every encryption key. Fortunately, it *may* be possible to relax the restrictions on friendliness in a future version of the algorithm.

Ultimately, since this protocol shows that the problem can be circumvented by simply ensuring the network topology is sink-free, this is a step forward on the path towards mitigating the quantum signal repeater problem preventing QKD networks from seeing wider use.