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

Alexis Goodfellow

December 12, 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 (1)(2). However, one of the key difficulties in the implementation of these protocols is the inability to use signal repeaters. This limitation exists because signal repeaters must observe a signal to be able to repeat it, but quantum information is immediately destroyed on observation. Research has been done on extending the range at which quantum information starts to decay, but the distance until the signal degrades is still short enough that connecting each host on the network to each other host with a single physical link remains impractical(3).

This paper proposes an algorithm for bootstrapping the perfectly secure encryption previously proven to be 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(4).

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].

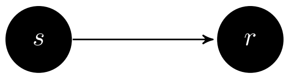

Figure 1: No intermediates at all

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.

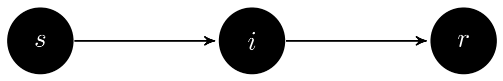

Figure 2: A sink node exists on a path from s to r

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

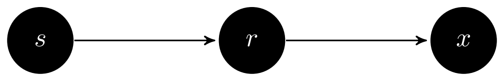

Figure 3: No sink node exists on a path from s to r

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:

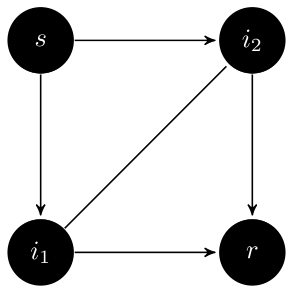

Figure 4: An unfriendly pair of intermediates

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.

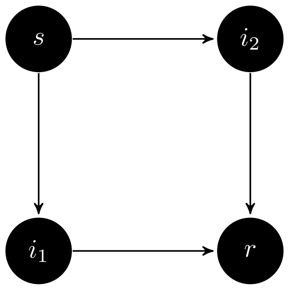
$$\begin{tikzcd} s \arrow[r] \arrow[d] & i_2 \arrow[d] \\ i_1 \arrow[r] & r \end{tikzcd}$$

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. 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. In order for this proof to be of any utility, it must be shown that it is possible to design an arbitrarily large network satisfying these conditions.

5 Arbitrarily Sized Friendly Sink-Free Networks

5.1 Construction Algorithm

1. Let N be a collection of $n_1, n_2, ..., n_i$ sets of network nodes where the $|n_k| > 1$ for any k.
2. Let G be a complete bipartite graph connecting the set of nodes n_1 with the set of nodes n_2.
3. For each remaining collection of nodes n_i in N:

(a) Extend the graph G by retaining all previous connections in graph G so far, and additionally connecting all nodes in the set n_{i-1} at the end of G with all nodes in the new set n_i, forming a complete bipartite sub-graph between n_{i-1} and n_i

5.2 Proof

Consider a complete bipartite graph G between two sets of nodes X and Y. If a node in X wishes to communicate with a node in Y, this is as simple as using a point-to-point protocol over the connection in the bipartite graph, which can be trivially shown to be secure.

Now consider the case where some node in X wishes to communicate with some other node in X. Since each node in X is connected to each node in Y and no nodes in X have direct connections to each other, it is necessary to use some node in Y as an intermediary in the communication.

If $|Y| = 1$, then it is trivial to demonstrate that the single node $y \in Y$ is a sink node. Moving to the general case, this is why it is necessary to impose the restriction that for a network N, each $n_k \in N$ subset of network nodes must satisfy $|n_k| > 1$ for any k. This condition guarantees the absence of any sink nodes from the graph, and this same reasoning applies inductively for removing any sink nodes from each complete bipartite sub-graph in G. This means that the whole network is guaranteed to be free of sink nodes.

Furthermore, the fact that each bipartite sub-graph is complete and the fact that each $n_k \in N$ set of network nodes must satisfy $|n_k| > 1$ for any k is sufficient to prove that any selection of two nodes $s, r \in G$ have at least two independent paths between them. This arises directly from the definition of a complete bipartite graph.

By the definition of a bipartite graph, it must be the case that no node in n_k can communicate between any other node in n_k for all subsets of nodes $n_k \in N$. Consequently, there can be no unfriendly nodes in the entire graph G.

6 Mitigating Failure Modes

Any networked system should be designed to be resilient against localized service outages, and so it's important to note some foreseeable failure modes of this system and strategies for mitigating them. The common theme of these failure modes is that some set of nodes in the network are rendered incapable of communicating with other nodes on the network, and these nodes fail in such a manner that either the friendly or sink-free conditions are violated.

The simplest example to conceptualize is where a single node on the network experiences an outage. Consider the repeated bipartite graph topology, and then consider the case where in that topology, there exists some $n_k \in N$ such that $|n_k| = 2$. Then assume that there is some node $f \in n_k$ that experiences a system failure and has to be taken down for maintenance. This one missing node would cause $|n_k| = 1$, which has been shown before to violate the sink-free constraint.

The resolution to this failure mode seems to be as simple as ensuring that there are multiple redundant nodes to avoid the possibility of one node failing compromising the sink-free constraint. However, a power outage is another possible failure mode which can affect multiple nodes *simultaneously.*

In a poorly designed network, it could be the case that every one of the nodes $n \in n_k$ is also subject to the condition $n \in d$ for some datacenter d. Assume that the datacenter d loses power. Consequently, each node $n \in n_k$ loses power, ultimately resulting in $|n_k| = 0$. This doesn't just represent a loss of the ability to securely communicate across the network, it represents a *bifurcation* of the network, resulting in some nodes being disabled from communicating with others entirely. In order to resolve this failure mode, it will be necessary to distribute the nodes of the network throughout multiple data centers, each of which has their own on premises redundant nodes in the case that a single node inside the datacenter must be removed from the network.

The failure modes already discussed arise due to network effects which would violate the sink-free precondition. The other precondition that could be violated is the friendliness condition. Friendliness is more or less assured in a network where all nodes are owned and operated by the same individual or organization. In order to expand a QKD network's size beyond the limits of a single organization, there must be a methodology for ensuring that one organization can communicate with another's network without compromising the friendliness condition in either subnetwork.

If this can be managed successfully, then by induction it will be possible to connect all QKD subnetworks together into one single interconnected QKD network without compromising the friendliness condition. Achieving this feat would be equivalent to creating a network that any actor with any motivations could trust to send their secret messages to no one else but their intended recipient.

7 Trusting the Network

An antagonistic model must be considered when designing a network with this degree of trust, because near-complete trust[2] in a security protocol can only be achieved when two actors on the network with diametrically opposed interests could not possibly forward information about each other outside of the intended communication path without being detected. Establishing such a degree of trust in any technology is incredibly difficult, as it is an extremely strict requirement. This hard requirement can be satisfied, but the argument demonstrating this fact requires understand some important concepts which must first be defined. These concepts are that of the *Quantum Information Forwarder* and *Exclusively Volatile Memory.*

[2]"near-complete" is the best assurance of trust that can exist for a security scheme as every computer system is theoretically susceptible to a generalized version of the Trusting Trust Attack(5)

7.1 Quantum Information Forwarders

The Quantum Information Forwarder (QIF) performs the same function for the QKD network as the Internet Service Provider (ISP) provides for a classical network. In both cases, there is a contract between an end-user and an organization: the end-user pays money to an organization, and in return the user gains access to the network along with the assurance of properly maintained communication across that network.

An end-user only need have a contract with a single ISP, but an end-user with only a single QIF would have to send every secure transmission through a network of nodes all owned by the same entity, and this is a security vulnerability. Thus, end users must use multiple QIFs to be assured that a single entity cannot acquire all keys associated with an encrypted message. This economic exchange of money for a secure communication service means that any given QIF has a clear economic incentive to avoid peering with every other QIF, as doing so would undermine not just the security model, but also undermine the business model of both peered organizations. Fundamentally, this means that each QIF has an *economically* imposed requirement to ensure the absence of messages sent outside of the intended path, which ultimately satisfies one of the key preconditions of friendliness.

7.2 Exclusively Volatile Memory

The other precondition of friendliness is that a node cannot permanently store the contents of any key or message it receives. To accomplish this, it must be the case that every node on the network be restricted to using only a short-lived memory needed for successfully sending a single transmission.

A satisfactory preexisting technology satisfying this requirement already exists, and it is called dynamic random access memory (DRAM). The reason this type of memory is called ”dynamic” is that the electrical charges representing the current state of the machine decay over time, and need to be recharged periodically to preserve the state of the machine. The result of removing this recharge step is the creation of a memory architecture that rapidly decays into a disordered state.

While removing this recharge step is a horrible idea for most applications, it is worthwhile to use in a cryptographic system [3]. The result of using this technology is a node which is implemented using exclusively volatile memory, and thus the node cannot permanently store information. In order for an actor to confirm that every one of the nodes on another network are friendly, said actor needs to be able to audit the nodes on the network they are about to join to confirm that they use exclusively volatile memory. Assuming that such a network audit can be performed, this satisfies the second condition of friendliness.

[3]this is subject to the constraint that the well-ordered state can be preserved long enough for any intended communications to be sent without experiencing data corruption

8 Conclusion

This paper proves the security of a distributed QKD algorithm through a sink-free friendly network, and also provides a construction of a network topology satisfying both the sink-free and friendly conditions. 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, and the friendliness restriction is impossible to overcome due to the security risks inherent with intermediate nodes being able to share information with each other.

Still, this protocol ultimately demonstrates that the signal repeater problem in QKD networks can be circumvented by simply ensuring the network topology is both friendly and sink-free. This network has been demonstrated to be possible predicated on the existence of at least two disjoint QIFs whose subnetworks share a similar sink-free construction as the series of consecutively conjoined bipartite graphs. As such, a working implementation of this proposed protocol would be an important step on the path to mitigating the quantum signal repeater problem, which one of the core problems preventing QKD networks from seeing wider commercial use.

References

[1] *Proceedings of IEEE International Conference on Computers, Systems, and Signal Processing, Bangalore, India*, 1984.

[2] A. K. Ekert, “Quantum cryptography based on bell’s theorem,” *Physical Review Letters*, vol. 67, no. 6, p. 661–663, May 1991.

[3] A. Zeilinger, “Long-distance quantum cryptography with entangled photons,” *Quantum Communications Realized*, Sep 2007.

[4] P. W. Shor and J. Preskill, “Simple proof of security of the bb84 quantum key distribution protocol,” *Physical Review Letters*, vol. 85, no. 2, p. 441–444, July 2000.

[5] K. Thompson, “Reflections on trusting trust,” *Communications of the ACM*, vol. 67, no. 8, p. 761–763, August 1994.