Quantum Secret Sharing: A Discussion and Implementation

Benjamin Fearon, Alex Rickman, Frank Zheng

June 2019

1 Introduction and Classical Secret Sharing

Secret sharing describes a protocol in which information (a secret) is encoded into n shares, with k shares able to construct the secret and k-1 shares carrying no relevant information whatsoever. This sort of procedure is referred to as a ((k,n)) threshold protocol, and is useful in situations where individual actors can't be trusted to act honestly, but they can be trusted to hold each other accountable. In a ((2,2)) threshold scheme, two agents would have a share of some secret, but each of their shares would individually provide no information about the secret. Thus, the agents would be forced to collaborate[1][2].

In an example of classical secret sharing, Alice might have a mission that she wishes to be carried out by agents Bob and Charlie. Alice knows that at least one of the two agents is trustworthy, but she isn't sure which. Thus, she requires a way to ensure that they complete the mission together, so that the honest agent can keep the dishonest agent in check. To encode the mission instructions (a binary string), Alice can simply add a second, random binary string. She can then send the random string to Bob, and the encoded mission instructions to Charlie (or vice versa). Now Bob and Charlie must work together in order to derive any meaning from their respective shares of the instructions.

However, if the dishonest agent (let's say Bob) is able to get hold of Charlie's share, he can recreate the mission instructions by himself and sabotage the mission. Thus, classical secret sharing is weak to eavesdropping, which is one of the principle motivators for quantum secret sharing methods. In this paper, we will discuss and quantum ((k,n)) schemes that uses quantum mechanisms to more securely share both quantum and classical information between multiple parties. The bulk of our implementation is regarding a ((2,2)) quantum secret-sharing scheme that encodes and decodes quantum information (rather than classical keys). Our implementation is two-fold: firstly in a local version which produces the correct qubit up to an overall sign, and a networked version which has Alice, Bob, Charlie, and the quantum machine all on different servers which correctly encodes and decodes a quantum secret (though it may differ by a sign).

2 Classical Key Algorithm

This section will discuss an algorithm, proposed by Hillery et al. that uses a GHZ triplet to share a classical secret between two agents. Let's again use Alice as the secret owner, and Bob and Charlie as the agents[1].

2.1 Encoding and Decoding

Initially, Alice, Bob, and Charlie each have access to a qubit, the three of which are entangled in a GHZ state:

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|1_a 1_b 1_c\rangle + |0_a 0_b 0_c\rangle) \tag{1}$$

Bob \ Alice	$+x$	$-x$	$+y$	$-y$
$+x$	$\|0\rangle + \|1\rangle$	$\|0\rangle - \|1\rangle$	$\|0\rangle - i\|1\rangle$	$\|0\rangle + i\|1\rangle$
$-x$	$\|0\rangle - \|1\rangle$	$\|0\rangle + \|1\rangle$	$\|0\rangle + i\|1\rangle$	$\|0\rangle - i\|1\rangle$
$+y$	$\|0\rangle - i\|1\rangle$	$\|0\rangle + i\|1\rangle$	$\|0\rangle - \|1\rangle$	$\|0\rangle + \|1\rangle$
$-y$	$\|0\rangle + i\|1\rangle$	$\|0\rangle - i\|1\rangle$	$\|0\rangle + \|1\rangle$	$\|0\rangle - \|1\rangle$

Figure 1: The state of Charlie's qubit as a function of Alice and Bob's respective measurement bases.

All three parties will measure their qubits, and announce the direction of the measurement (either x or y), but not the result. The entangled nature of the GHZ triplet effectively encodes the result of Alice's measurement (the secret) in the results of Bob and Charlies measurements (the shares).

If Alice and Bob both measure their qubits in the x direction, the result on the GHZ triplet wave-function can be expressed as follows:

$$|\psi\rangle = \frac{1}{2\sqrt{2}}[(|+x\rangle_a |+x\rangle_b + |-x\rangle_a |-x\rangle_b)(|0\rangle_c + |1\rangle_c) + (|-x\rangle_a |+x\rangle_b + |+x\rangle_a |-x\rangle_b)(|0\rangle_c - |1\rangle_c)] \quad (2)$$

Where the x and y eigenstates are given by:

$$|\pm x\rangle_i = \frac{1}{\sqrt{2}}(|0\rangle_i \pm |1\rangle_i) \quad (3)$$

$$|\pm y\rangle_i = \frac{1}{\sqrt{2}}(|0\rangle_i \pm i\,|1\rangle_i) \quad (4)$$

From equation (2), one can see that if Charlie also measures his particle in the x direction, he will know whether Alice and Bob's measurements are the same ($|+x\rangle_c$), or opposite ($|-x\rangle_c$). However, since Charlie doesn't know the result of Bob's measurement, he knows nothing about Alice's measurement (except that it was in the x basis). Similarly, Bob knows the result of his own measurement, but doesn't know whether Alice's measurement is the same or the opposite of his. The only way for Charlie or Bob to learn Alice's measurement is to combine their results. This protocol works the same way if every player measures in the y basis. The state of Charlie's qubit as a function of Alice and Bob's respective measurement choices is shown in Figure 2.2.

2.2 Eavesdropping and Dishonesty

Now, what happens if one or both of Bob and Charlie measure in a different basis than Alice? In these situations, neither Bob nor Charlie can determine Alice's result. For this reason, if each player determines their measurement basis randomly, some constant (time-averaged) percentage of the trials will not provide usable information. However, if Bob or Charlie attempts to cheat by measuring the other player's qubit, this cheating attempt will upset the percentage of unusable trials, which the group can detect. For more information, see the paper by Hillery et al. [1]

3 Quantum Key Algorithm

The previous section discussed a method of sharing classical information between multiple parties. Quantum methods can also be used to share quantum information. If Alice wants to share an arbitrary quantum state $|\psi_A\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$ between Bob and Charlie, she can do so with a GHZ

triplet, much like in the classical secret case. Alice first combines her secret particle and her GHZ particle, and then measures the pair in the Bell basis (equations (5) and (6)).

$$|\Psi\pm\rangle_{Aa} = \frac{1}{\sqrt{2}}(|00\rangle_{Aa} \pm |11\rangle_{Aa}) \tag{5}$$

$$|\Phi \pm Aa\rangle = \frac{1}{\sqrt{2}}(|01\rangle_{Aa} \pm |10\rangle_{Aa}) \tag{6}$$

The four particle state can then be exprressed as follows:

$$\begin{aligned} |\psi_4\rangle =& \frac{1}{2}[|\Psi_+\rangle_{Aa}(\alpha|00\rangle_{bc} + \beta|11\rangle_{bc}) + |\Psi_-\rangle_{Aa}(\alpha|00\rangle_{bc} - \beta|11\rangle_{bc}) \\ &+ |\Phi_+\rangle_{Aa}(\alpha|00\rangle_{bc} + \beta|11\rangle_{bc}) + |\Phi_-\rangle_{Aa}(\alpha|00\rangle_{bc} - \beta|11\rangle_{bc})] \end{aligned} \tag{7}$$

Alice then chooses either Bob or Charlie at random (let's say she chooses Bob), and has him measure his particle along the x axis.

If Alice tells Charlie what Bell state she measures, Charlie now only requires the result of Bob's measurement in order to reconstruct Alice's measurement via the following gate applications:

Alice's Result	Measurer's Result	Reconstructing Operators
$\|\Psi_+\rangle$	$\|+x\rangle$	I
$\|\Psi_+\rangle$	$\|-x\rangle$	Z
$\|\Psi_-\rangle$	$\|+x\rangle$	Z
$\|\Psi_-\rangle$	$\|-x\rangle$	I
$\|\Phi_+\rangle$	$\|+x\rangle$	X
$\|\Phi_+\rangle$	$\|-x\rangle$	XZ
$\|\Phi_-\rangle$	$\|+x\rangle$	XZ
$\|\Phi_-\rangle$	$\|-x\rangle$	X

This will bring Charlie's qubit into Alice's original state, as desired. By comparing a subset of desired inputs to reconstructed outputs periodically, Alice can detect dishonestly amongst Bob and Charlie as well as any eavesdroppers who have entangled an ancilla qubit onto the shared state. All of these cheating schemes inherently introduce errors into the key-sharing, by design of the protocol. For more details of this process we refer the reader to the Hillery paper[1].

4 Implementation

In this section we discuss an efficient implementation of the Quantum Key Algorithm in PyQuil[3]. Though this procedure is well-known, we aim to implement it in a more elegant and modular way than traditional methods. Here we translate the high-level features of the algorithm into pseudocode, and direct the reader to our public codebase on github[4] for more details about the PyQuil code itelf. We note that while the steps themselves are fairly straightforward to implement in PyQuil, it's more complicated to separate Alice, Bob, Charlie, and the Quantum machine onto to three different servers which send commands over a network. Our implementation allows Alice to share a single qubit by specifying an α and a β. Bob and Charlie will each receive a quantum share of Alice's qubit, and by combining their information, one of them will be able to recreate her quantum state.

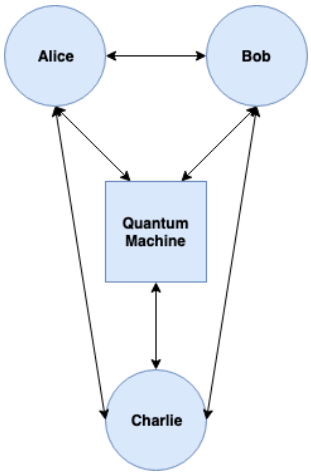

Figure 2: This shows the diagram of the network between Alice, Bob, Charlie, and the quantum machine.

1. Generate an arbitrary state: Once the quantum machine receives the alpha and beta from Alice, we use the Grove package from the PyQuil Library[3] to generate an arbitrary quantum state
$$\alpha |0\rangle + \beta |1\rangle . \tag{8}$$

2. Initialize a GHZ state: The quantum machine then entangles the qubits into a GHZ states with the standard Hadamard and CNOT gates from PyQuil.

3. Alice measures in the Bell Basis: Alice tells the quantum machine to measure in the Bell Basis. Since PyQuil only provides direct measurement in the Z-basis, we perform a change of basis procedure whereby we apply a CNOT and a Hadamard in sequence before measuring in Z. The four bit results of this measurement will tell us which of the four Bell states Alice Measured.

4. Alice chooses Bob or Charlie at random to measure in x-basis: This can be performed with any random number generating techniques such as the random package from the Standard Python Library. It then sends a command over the network to Bob or Charlie to measure the corresponding qubit in the x-basis.

5. Alice reports Bell Basis measurement to other: Alice sends her Bell Basis measurement to the other person over the network.

6. Chosen one reports x measurement to the other: Whomever was told by Alice to do the x-measurement sends their measurement over the network to the other person.

7. Other reconstructs the original arbitrary state by applying appropriate operators: Lastly, the reconstructing agent tells the quantum computer to apply the standard PyQuil gates according to the table above.

   This will bring their qubit into Alice's original state, as desired, and the quantum computer sends the wavefunction back.

   In the local implementation, we use a branching program to apply the appropriate operator, which we only measure once. We return a wavefunction requiring one run of the program (from the appropriate branch) and obtain correct quantum information up to a global sign. We note that in the proof-of-concept network implementation, Bob or Charlie does not have access to

shared program and thus cannot construct a branching program in the same way. We therefore cannot with PyQuil prove via print statements, etc, that they obtained the correct qubit, since this requires running the program again, thus collapsing the wavefunction into a potentially different state than Alice and Bob initially measured. On a real quantum computer, the reconstructing agent would simply have access to the qubit carrying the quantum information Alice sent. In our case, on a qvm, we use the wavefunctionSimulator class[3] to return the wavefunction with each amplitude correct up to a sign.

5 Conclusion

We implement a straightforward, efficient, and modular quantum key sharing algorithm using the PyQuil Library from Rigetti Computing[3]. In addition to the basic implementation, we included a proof-of-concept in which we set up four different servers that send commands to each other to simulate a real-life situation. Alice, Bob, and Charlie are connected to each other and are also networked to a quantum machine on which they can all operate.

As this is a proof-of-concept, we did not take time to provide security on our network communication. For instance, when sending commands, we simply ask the servers to include their name (either Alice, Bob, or Charlie), and the quantum machine takes these names at face value without authenticating if the commands are actually coming from the right person. Thus, the next step would be to include encryption techniques within our server-client scheme such that each entity would be able to authenticate itself so that the server knows for certain that the person claiming themselves to be Alice is telling the truth. This would be necessary in any real-world security protocol of this form. We also note that while we added functionality for Alice to send a string of arbitrary qubits in our standard implementation, the proof-of-concept server-client implementation does not yet include this. This would be another clear next step. Lastly, we note that papers on this topic [2][1] discuss protocols for sharing between more than three parties, and we could continue this work by including functionality for several more quantum key sharing parties.

References

[1] M. Hillery, V. Bužek, and A. Berthiaume, "Quantum secret sharing," , vol. 59, pp. 1829–1834, Mar 1999.

[2] R. Cleve, D. Gottesman, and H.-K. Lo, "How to Share a Quantum Secret," , vol. 83, pp. 648–651, Jul 1999.

[3] R. S. Smith, M. J. Curtis, and W. J. Zeng, "A practical quantum instruction set architecture," 2016.

[4] F. Z. Benjamin Fearon, Alex Rickman, "Cs269q final project: Quantum secret sharing," 2019.