Comparing and Implementing Different Quantum Secret Sharing Schemes

Yousef Hindy, Pieter-Jan Stas
(Dated: June 8, 2019)

In this project, we investigate the idea of quantum secret sharing, in which a secret (usually a quantum state) is encoded into multiple qubits or "shares" which are then distributed to relevant parties. We implement a $(3, 5)$ scheme that has been around in literature for quite some time in `pyQuil`. We also contribute our own scheme that allows for any n and is a (n, n) scheme and implement it in `pyQuil` as well. Finally, we investigate the effects of noisy qubits on the fidelity of these designs.

INTRODUCTION

Recent advances in quantum information technology have nearly made quantum-secure communications possible. In a post-quantum world where classical encryption and communication systems are made vulnerable, there is a considerable need for systems that are robust and reliable for sharing information.

Consider the following problem: you are the owner of a bank vault, and you have eleven employees. You want to give your employees access to the vault, but you do not want to allow one or two or even three employees to conspire and unlock the vault together. In fact, you want to create a system of keys and locks such that you would need a majority of the employees (six) to come together to unlock the vault. Once you have created this, you have essentially created a $(6, 11)$ *threshold scheme*.

More generally, a (k, n) threshold scheme is one in which a secret is encoded into exactly n shares that are distributed. With at least k shares, you can perfectly reconstruct the secret. With $k - 1$ or fewer shares, however, you gain no information at all about the secret and are out of luck. The idea was first introduced by Adi Shamir in 1979 [1], where the shares were classical bits of information.

The idea can be extended into the quantum regime too, as shown by Cleve et. al in 1999 [2]. Instead of distributing n shares of classical information, we can disseminate n shares of quantum information in the form of qubits. In general, we take a quantum state $|\psi\rangle$ which lies in a Hilbert space $\mathcal{H}$ of dimension d and encode it into a new state $|phi\rangle$ or ρ (depending on whether you are using a pure state scheme or a mixed state scheme) which lies in a Hilbert space $\mathcal{H}'$ with dimension d^n. Given k out of n parts of the encoded state, you should be able to reproduce the original state $|\psi\rangle$. In [2], they show that due to the no-cloning theorem, $n < 2k$.

There is a deep connection between quantum secret sharing (QSS) schemes and quantum error correction (QEC) codes. In fact, every QSS is a QEC, but the relation does not necessarily hold the other way. QSS codes correct for $k-1$ erasure errors, and could be useful if implemented towards fault-tolerant quantum computation. The difference is that in a general QEC code, one can usually recover *some* information about the state from $k - 1$ shares, whereas in QSS codes, no information can be recovered.

In this paper, we present the implementation of two QSS schemes. The first is a $(3, 5)$ qubit sharing scheme that is mentioned in [2] but is actually described in [3] and [4]. We encountered several errors in the derivation and description in [4] and correct for them. The second scheme we present is an original one that we created that essentially encodes information in the parity of the product state of the qubits. It works in general as an (n, n) secret sharing scheme. We present examples and code that demonstrate our system in action.

WHAT WE IMPLEMENTED

We investigated three separate schemes, but were only able to implement two, as one would require more qubits than we can simulate (or run on the cloud at the moment) to do anything interesting. We present the schemes and the results of the simulations below. All test were run on the Rigetti Quantum Virtual Machine (QVM).

Scheme 1: $(k = 3, n = 5)$

The first scheme was mentioned in the original Cleve et. al paper [2], but originally produced as an error correcting scheme in Luhe et. al [4]. It encodes a secret state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ into a five qubit product state. With any three of the qubits, one can recover the original qubit's state. The scheme is shown in Figure 1. To decode from any of the three qubits, one must apply a Hadamard gate to one of the qubits and then do a Bell-state measurement on the rest. The outcome of the Bell-State measurement gives instructions for which Pauli transformation to apply to the first qubit to recover the original state. The approprimate transformations are given in Table I.

To test our scheme, we encoded a qubit in the $|1\rangle$ state and ran it through the scheme. At the end, we measured the target qubit and found how many times it was in the $|1\rangle$ state. In this case, we found that the qubit was in the $|1\rangle$ state 100% of the time, indicating that the scheme

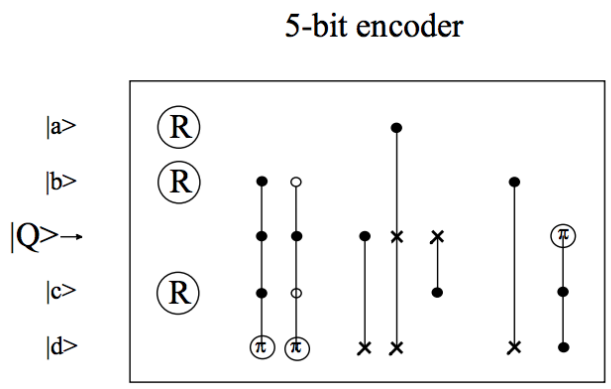

FIG. 1: $(3, 5)$ encoding scheme. The x's represent controlled nots, and the π's in circles represent multiplying by -1. Closed circles mean that the control is turned on when the qubit is in the state $|1\rangle$ and open circles mean that the control is turned on when the qubit is in state $|0\rangle$. Courtesy of [4]

Qubit A	Qubit B	Transformation on Qubit C
0	0	I
0	1	X
1	0	Z
1	1	XZ

TABLE I: Transformations Required After Bell-State Measurement For $(3, 5)$ Scheme

was able to recover the original qubit. We ran the same experiment for the $|0\rangle$ state, and achieved the same results. We extended this to an equal super position state: $|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, and found that when we measured the state, it was in the $|0\rangle$ state 49% of the time. Future work would include doing full state tomography on the final qubit to get the actual values of α and β.

The code for the implementation of this method can be found in the repository we included in the `quantum_secrets.ipynb` notebook. To change the original state that is encoded, one can adjust the program in the `prepare_state(qubit)` method.

Scheme 2: $(k = n, n)$

An interesting scheme we came up with is a $(k = n, n)$ which functions for any value of n. The principle is quite simple: we encode the qubit we want to transmit $|Q\rangle = \alpha|0\rangle + \beta|1\rangle$ into a new logical basis $\alpha|0\rangle_L + \beta|1\rangle_L$ of size 2^n using the method described in Fig. 2. $|0\rangle_L$ corresponds to an equal superposition of all states contain an even number of $|1\rangle$ states, whereas $|1\rangle_L$ corresponds to an equal superposition of all states containing an odd number of $|1\rangle$ states. The encoded states are shown in equation (1) for $n = 2$, $n = 3$, and $n = 4$ below (ignoring

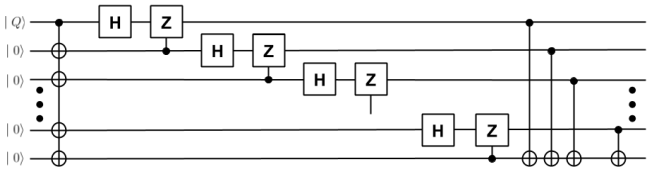

FIG. 2: $(k = n, n)$ encoding scheme. The solid dots correspond to the control qubit of a controlled gate, the clear circles with cross are NOT-gates, the H's are Hadamard gates and the Z's are Pauli-Z gates. Note that $|Q\rangle$ is the original qubit we want to encode, $\alpha|0\rangle + \beta|1\rangle$

.

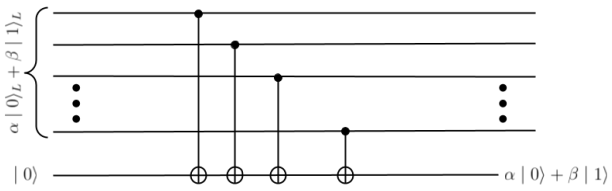

FIG. 3: $(k = n, n)$ deccoding scheme

normalization):

$$
\begin{aligned}
n = 2 : |0\rangle_L &= |00\rangle + |11\rangle \\
|1\rangle_L &= |01\rangle + |10\rangle \\
n = 3 : |0\rangle_L &= |000\rangle + |011\rangle + |101\rangle + |110\rangle \\
|1\rangle_L &= |001\rangle + |010\rangle + |100\rangle + |111\rangle \\
n = 4 : |0\rangle_L &= |0000\rangle + |0011\rangle + |0101\rangle + |0110\rangle \\
&\quad + |1010\rangle + |1001\rangle + |1100\rangle + |1111\rangle \\
|1\rangle_L &= |0001\rangle + |0010\rangle + |0100\rangle + |1000\rangle \\
&\quad + |0111\rangle + |1011\rangle + |1101\rangle + |1110\rangle
\end{aligned}
$$

Taking $n = 3$ as an example, we can see that if every qubit gets distributed to a different player (say Alice, Bob, and Charlie), each player individually has a qubit $\alpha(|0\rangle + |1\rangle) + \beta(|0\rangle + |1\rangle)$. This gives no information at all about the secret. Even if two players put their respective qubits together (whether it be Alice and Bob, Alice and Charlie, or Bob and Charlie), they get an equal superposition of all possible 2-qubit states: $\alpha(|00\rangle + |01\rangle + |10\rangle + |11\rangle) + \beta(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$, which still discloses no information about the original qubit $|Q\rangle$. Only when all three players put their qubits together can they recover the original qubit.

If one has access to all qubits, it is possible to recover the secret by measuring the parity of the state. There are undoubtedly several ways of doing this, but we have found a simple way to be the method described in Fig. 3.

Similarly to the $(k = 3, n = 5)$ scheme, we ran several

tests on the algorithm and found it to be reliable and functioning as expected. The code for the implementation of the $(k = n, n)$ method can be found in the repository we included in the `NN_sharing_scheme.ipynb` notebook.

Polynomial Codes

We also experimented with a class of codes that encode the original state into a polynomial combination of states. This code, based on CSS codes, was the basis of many of the proofs about these codes in the original Cleve et. al [2] paper. The basic idea is that the secret is encoded in the coefficients of a polynomial that is evaluated at n different points. Given the superposition of the polynomial over different coefficient values, one can recover the original state from k shares.

We were hoping to use this scheme to implement a general (k, n) code, but we found that in order to implement it, we would need at least 3 physical qubits to represent each "qubit" in the superposition, as we would be working in $\mathbb{Z}_8$ at a minimum. This means to implement an $n = 6$ scheme, we would need to have at least 18 logical qubits plus any additional ancillae used for intermediate computation. Once quantum computers are able to handle more qubits, we would be able to implement a scheme that would go beyond the schemes implemented above.

EFFECTS OF NOISE ON SCHEME 2: $(k = n, n)$

Interestingly, the $(n.n)$ QSS behaves quite badly in a noisy environment, as seen in Fig. 4. Here we added a bit flip probability of 0.1 for every single encoded qubit. This makes sense, as all qubits are needed for the decoding to get the original qubit state. If even one qubit experiences a bit flip, that is already enough to give a completely wrong result. Thus, the higher the number of encoded qubits is, the higher the chance for error. This error could of course be greatly improved if we used a quantum error correcting code for every shared qubit (but would greatly increase the number of required qubits).

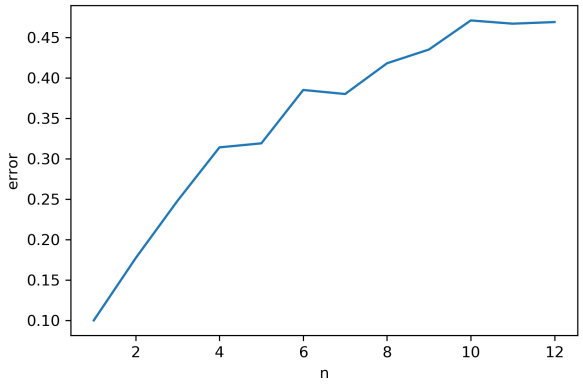

FIG. 4: Error of $(k = n, n)$ Quantum Secret Sharing scheme as n increases. Note that at error = 0.5, the state is completely depolarized and all information is lost.

FUTURE WORK

We originally wanted to implement a (k, n) scheme for any arbitrary k and n that obey the rule $n \leq 2k - 1$ given by [2]. Upon learning more about different quantum sharing schemes, this turned out to be way too ambitious, not to mention that such a scheme might render any other QSS obsolete. We did however come up with the less flexible (n, n) QSS, which is a first step in the right direction. One interesting future direction we might want to take would be trying to find, for example, a method $(n-1, n)$ or $(n-2, n)$ that works for any value of n.

Something we would want to try out in the future is, of course, to run our code on a real quantum computer. For this project we have only used the QVM provided by Rigetti, but the real deal is to test it out on an actual QC. This would also give us a better idea of how our quantum secret sharing schemes behave in real noisy environments, as opposed to the simulated noise we have used to test our code on the QVM.

Particularly for the $(n.n)$ QSS, it would be interesting to implement a Qauntum Error Correcting Code, as we have seen that introducing a noisy environment greatly diminishes its accuracy. Unfortunately, this would also significantly increase the required qubits for the On a real quantum computer, the currently limited number of accessible qubits would also make testing the limit of n difficult. As quantum chips increase their number of qubits in the future, this would be something interesting to test.

In addition, we had hoped to run noise experiments on the $(3, 5)$ scheme that we implemented, but since we used gate modifiers to create the triple controlled π rotations, we were not able to use the noisy QVM, as we got the error `QVMError: The noisy QVM doesn't support gate modifiers.` Hopefully, in future iterations of pyquil, this will be fixed so we can examine the effects of noise on the Cleve scheme.

ACKNOWLEDGEMENTS

We would like to thank Professor Boneh and Dr. Zeng for their advice in the early steps of our project. We would also like to thank the entire teaching team of CS 269Q for their help and support throughout the quarter.

[1] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, pp. 612–613, Nov. 1979.

[2] R. Cleve, D. Gottesman, and H.-K. Lo, "How to share a quantum secret," *Physical Review Letters*, vol. 83, no. 3, p. 648, 1999.

[3] R. Laflamme, C. Miquel, J. P. Paz, and W. H. Zurek, "Perfect quantum error correction code," 1996.

[4] H. Lu, Z. Zhang, L.-K. Chen, Z.-D. Li, C. Liu, L. Li, N.-L. Liu, X. Ma, Y.-A. Chen, and J.-W. Pan, "Secret sharing of a quantum state," 2016.