Rational Quantum Secret Sharing

Manan Rai
Department of Computer Science
Stanford University
mananrai@stanford.edu

Jerry Zhilin Jiang
Department of Computer Science
Stanford University
zjiang23@stanford.edu

Andrew Tierno
Department of Computer Science
Stanford University
atierno@stanford.edu

1 Introduction

Secret sharing is a common cryptographic technique that involves partitioning information into parts with the unique property that each of these pieces by itself cannot be used to recover the original information. In this way, a secret holder may disseminate their data among a set of peers and know that unless they collaborate, the secret will remain unknowable to all parties involved. More formally, we define a (k, n)-threshold scheme as an encoding scheme under which a secret S (e.g. a bit string) is converted into n shares, with which the original message S can only be recovered if k parties collaborate, and any subset of $(k-1)$ shares does not reveal any information about the value distribution of the secret S. Quantum Secret Sharing attempts to extend this scheme to work with quantum secrets.

2 Background

There are several techniques by which we might extend this concept into the quantum domain. The Quantum Secret Sharing (QSS) protocol as described by Hillery et al. [1999]. and further by Cleve et al. [1999] operates by having three people (or more generally n) share GHZ state entangled qubits, and then measuring them in various bases according to the instructions of a secret keeper. More modern protocols have since been proposed, and the one in particular we investigate is the Rational Quantum Secret Sharing (RQSS) protocol proposed by Qin et al. [2018]. The RQSS algorithm proposes a slight tweak on the secret "game" where instead of "good" agents that always comply with the protocol and "bad" agents that always try and break it, all participants are "rational" agents in that they'd prefer to have the secret over not having the secret, but will not make any effort to pass the secret along once they have obtained it, and would further prefer nobody getting the secret to others getting the secret while they don't.

With this in mind, the RQSS algorithm is designed as a game where the secret holder only transmits the true secret with some probability p. The participants in the game must do work to decrypt the transmitted bit-string, and share their results (on punishment of the secret sharing ending early), prior to knowing whether the transmitted bit string was the true secret. In this way, the participants are sufficiently motivated to cooperate with one another while preventing "cheaters" that wish to keep the secret for themselves.

3 Methods

3.1 Entanglement

As a part of the RQSS scheme proposed by Qin et al. [2018], we utilize quantum entanglement to create n-qubit entanglements. Given a qubit ϕ storing the secret state after Inverse Quantum Fourier Transform (IQFT) and $(n-1)$ share qubits initialized to state $|1\rangle$, the entanglement algorithm applies Controlled-NOT (CNOT) gates between ϕ and each of the share qubits, so that all n qubits affect the state of one another when any one of them is measured or otherwise tampered with. After receiving all qubits of a n-qubit entanglement, the receiving party Bob then disentangles the set of qubits by applying the CNOT operation between ϕ and every share qubit again. This process is essential to the algorithm by enabling every receiving party (Bob) to detect cheating of other peers and abort accordingly.

3.2 Quantum Fourier Transform

Note that the Fourier transform is defined as

$$\hat{f}(t) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i x t} dx \tag{1}$$

and the generalized form of Discrete Fourier Transform, which is defined as follows

$$\tilde{x}_k = \sum_{n=0}^{N-1} x_n e^{\frac{2\pi i}{N} kn} \tag{2}$$

maps a sequence of complex numbers $x_0, \ldots, x_{N-1}$ to another sequence $\tilde{x}_0, \ldots, \tilde{x}_{N-1}$. Similarly, the Quantum Fourier Transform is defined as

$$\sum_j \alpha_j |j\rangle \to \sum_k \tilde{\alpha}_k |k\rangle \tag{3}$$

where

$$\tilde{\alpha}_k = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} \alpha_n e^{\frac{2\pi i}{N} kn} \tag{4}$$

Then, given the N^{th} root of unity, $\omega = \frac{2\pi i}{N}$, we can define the kn^{th} element of the transform as ω^{kn}. We can express this as a linear transform as

$$\tilde{F}_N = \begin{bmatrix} \omega^0 & \omega^0 & \omega^0 & \ldots & \omega^0 \\ \omega^0 & \omega^1 & \omega^2 & \ldots & \omega^{N-1} \\ \omega^0 & \omega^2 & \omega^4 & \ldots & \omega^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \omega^0 & \omega^{N-1} & \omega^{2(N-1)} & \ldots & \omega^{(N-1)^2} \end{bmatrix} \tag{5}$$

and conversely, the Inverse Fourier Transform is defined as

$$\tilde{F}_N^\dagger = \begin{bmatrix} \omega^0 & \omega^0 & \omega^0 & \ldots & \omega^0 \\ \omega^0 & \omega^{-1} & \omega^{-2} & \ldots & \omega^{-(N-1)} \\ \omega^0 & \omega^{-2} & \omega^{-4} & \ldots & \omega^{-2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \omega^0 & \omega^{-(N-1)} & \omega^{-2(N-1)} & \ldots & \omega^{-(N-1)^2} \end{bmatrix} \tag{6}$$

In our implementation, we define N-qubit PyQuil gates `QFT` and `IQFT` (representing the linear transforms as defined in Eqn. 5 and 6).

4 Protocol

The n-out-of-n RQSS protocol for the case of 1 secret sharer ("Alice") and n secret receivers ("Bob$_1$", ..., "Bob$_n$") can be broadly divided into the following steps:

- Alice decides with probability γ whether to reveal the true secret or to prepare a fake, testing secret
- Alice takes her d-dimensional (d=2 in our implementation, because we are operating on qubits instead of qutrits or other bases) secret φ, which may be the one she intends to share or simply a fake, and conducts Inverse Quantum Fourier Transform (IQFT) on the secret to get φ'
- Alice concatenates φ' and $n-1$ signal bits, each initialized to the pure state $|d-1\rangle$ (in our case, $|1\rangle$)
- Alice entangles all bits above to get the n-particle entangled state Φ using CNOT gates
- Alice conducts QFT on this state to get Φ'
- Alice creates n copies of Φ'
- Alice sequentially transmits one bit from every Φ' to a Bob
- The Bobs will then trade qubits with one another until they have a full set of particles Φ' that are entangled with one another and conducts IQFT on Φ' to get Φ
- Each Bob disentangles the state Φ using CNOT operations to get φ' and the signal bits
- Each Bob conducts IQFT on φ' to get the original secret φ
- Every Bob measures the signal bits, if they haven't received all the signal bits or if they do not all resolve to the state $|d-1\rangle$, they report the presence of cheating
- If cheating has been reported, the protocol terminates. Otherwise, Alice reveals if the Bobs now possess the true secret thus ending the protocol, or if it was fake and that now they must begin the protocol again

5 Experiments and Results

5.1 Overview

Our PyQuil implementation source code is publicly available at: `https://github.com/zjiang23/CS269Q-RQSS`. Both the Quantum Virtual Machine (QVM) and the Quil Compiler (QuilC) need to be running in the background (`qvm -S` and `quilc -S`, respectively) before executing the tests.

Since we implement the protocol modularly, we test that the entanglement and Fourier Transform procedures work as expected. In order to test the entire protocol, we run several experiments with 1 Alice and variable numbers of Bobs. We conduct experiments in the following situations:

- Pure Case: 1 Alice, n Bobs. Alice shares correct secret and all Bobs properly share their qubits with one another.
- Alice Sends Out Garbage: 1 Alice, n Bobs. Alice shares a garbage secret in order to test that the Bobs act rationally.
- Bob Refrains: 1 Alice, n Bobs, $m < n$ Bobs refrain. Alice shares a [pure/garbage] secret, but m Bobs do not share their secret.
- Bob Cheats: 1 Alice, n Bobs, $m < n$ Bobs cheat. Alice shares a [pure/garbage] secret, but m Bobs cheat by altering their qubits before sharing them with the others.

In the first two situations, we expect the protocol to succeed. In the latter two, the protocol breaks because the Bobs fail to act per expectations. With the aforementioned tests, we also calculate statistics of how the protocol fares in the event of cheating, including True Positives (protocol fails in the event of cheating), False Positives (protocol fails with no cheating), True Negatives (protocol succeeds with no cheating), and False Negatives (protocol succeeds in the event of cheating))

5.2 Pure Case

In the "pure case", Alice always uses the actual secret (setting $\gamma = 1$ using argument `-p 1`), and no Bob ever cheats. We see that in the pure case, all Bobs end up with n qubits. Note that the first qubit corresponds to the secret, and the remaining $n-1$ qubits are used by Alice to verify the veracity of the Bobs' motivations.

```
$ python tests.py -v -p 1
Running test with 1 Alice and 3 Bobs, and no cheating Bobs:
If no Bobs are cheating, all single particles should be 1.
Bob 1 received single particles: 1, 1
Bob 2 received single particles: 1, 1
Bob 3 received single particles: 1, 1
SECRET REVEALED [(0.7071067812+0j)|0> + (-0.7071067812+0j)|1>]
```

We observe that the protocol succeeds in this case, as expected.

5.3 Alice Sends Out Garbage

When Alice sends out a garbage secret, and no Bob ever cheats, the protocol will run till the end where the Bobs successfully retrieve a shared secret value, only to receive announcement from Alice that the secret is in fact garbage, and the protocol would restart again from the beginning.

When Alice always sends out garbage (setting $\gamma = 0$ using argument `-p 0`), the protocol always runs through and starts over repeatedly:

```
$ python tests.py -v -p 0
Running test with 1 Alice and 3 Bobs, and no cheating Bobs:
If no Bobs are cheating, all single particles should be 1.
Bob 1 received single particles: 1, 1
Bob 2 received single particles: 1, 1
Bob 3 received single particles: 1, 1
FAKE SECRET REVEALED, RETRYING...
If no Bobs are cheating, all single particles should be 1.
Bob 1 received single particles: 1, 1
Bob 2 received single particles: 1, 1
Bob 3 received single particles: 1, 1
FAKE SECRET REVEALED, RETRYING...
...
```

5.4 Bob Refrains

If any of the Bobs refuses to send out its shares, then at least one Bob will not receive its complete set of shares for secret retrieval. In our experiments, each Bob is directly given a collection of qubits it "receives," so Bob will abort if the collection of qubits does not include a complete set of shares. In practice, this will lead to the Bob reaching a time-out and aborting the process.

5.5 Bob Cheats

As mentioned by Qin et al. [2018], even if a Bob received a tampered qubit from its peer, there is an upper bound $\frac{1}{d}$ probability that the Bob fails to detect the tampering due to the nature of the algorithm. In our case, since $d = 2$, the probability of getting a false positive (i.e. accepting a tampered share) should therefore be $\frac{1}{d} = \frac{1}{2}$. As the number of rounds played before Alice reveals the true secret can be modeled as $X \sim \text{Geo}(\gamma)$, we see that the overall probability of a cheater not being discovered is bounded above as

$$p = \sum_{i=1}^{\infty} \left(\frac{1}{d}\right)^i \Pr(X = i) = \sum_{i=1}^{\infty} \left(\frac{1}{d}\right)^i \gamma(1-\gamma)^{i-1} \tag{7}$$

$$= \frac{\gamma}{d} \sum_{i=1}^{\infty} \left(\frac{1-\gamma}{d}\right)^{i-1}$$

$$= \frac{\gamma}{d} \left(\frac{1}{1-\frac{1-\gamma}{d}}\right) = \frac{\gamma}{\gamma + (d-1)} \tag{8}$$

γ	True Positives	False Positives	True Negatives	False Negatives
0.4	20	28	72	0
1	20	40	60	0

Table 1: Results for multiple runs with different γ values

With $\gamma = 0.4$ and $d = 2$, we should observe $p = \frac{0.4}{0.4+(2-1)} \approx 0.29$ of cheating cases being false positives. With $\gamma = 1$ and $d = 2$, we should observe $p = \frac{1}{1+(2-1)} \approx 0.50$ of cheating cases being false positives.

We run our tests with argument `-a` (equivalently `--run_all`), which sequentially runs a series of different test scenarios.

Example where cheating is detected:

```
Running test with 1 Alice and 3 Bobs, such that Bob 3 cheats
with a consistent program:
If no Bobs are cheating, all single particles should be 1.
Bob 1 received single particles: 1, 1
Bob 2 received single particles: 0, 0
Bob 3 received single particles: 1, 1
CHEATING DETECTED
```

Example where cheating is not detected:

```
Running test with 1 Alice and 2 Bobs, such that Bob 2 cheats
with a random program:
If no Bobs are cheating, all single particles should be 1.
Bob 1 received single particles: 1
Bob 2 received single particles: 1
SECRET REVEALED [(0.7071067812+0j)|0> + (-0.7071067812+0j)|1>]
```

Across multiple runs of all test cases, with $\gamma = 0.4$, we observe approximately a $2:5$ false positive versus true negative ratio, and 0.28 of cheating cases are not caught by the protocol. With $\gamma = 1$, we observe approximately a $2:3$ false positive versus true negative ratio, and 0.40 of cheating cases are not caught by the protocol. Furthermore, we observe no cases of false negatives at all. These results are summarized in Table 1. Note that these empirical results are consistent with our theoretical hypotheses above.

6 Future Work

We intend to extend this protocol further by developing a noise tolerant version of the procedure. While the protocol works successfully on a simulated quantum computer, it would not fare as well on an actual quantum processor. In particular, the state of the secret and single particles can get corrupted during transmission. Under the standard algorithm, this would prompt the Bobs to declare each other as cheating, thus disrupting the protocol. We envision a few solutions to this problem. For one, we could employ an error correcting scheme such as Shor's code to help correct any corruptions (though given current quantum hardware limitations this solution does not scale well). We could further use a majority vote among the Bob's to decide which, if any, Bob is lying.

7 Conclusion

We have successfully implemented the logic of the Rational Quantum Secret Sharing (RQSS) algorithm, and various test scenarios to prove the correctness of our implementation. This shows the feasibility of the algorithm on a simulated level. With future development of quantum computing hardware, it looks promising that the RQSS algorithm can be tested on actual quantum computers and put into practical applications in the future.

References

Richard Cleve, Daniel Gottesman, and Hoi-Kwong Lo. How to share a quantum secret. *Physical Review Letters*, 83(3):648, 1999. 1

Mark Hillery, Vladimír Bužek, and André Berthiaume. Quantum secret sharing. *Physical Review A*, 59(3):1829, 1999. 1

Huawang Qin, Wallace KS Tang, and Raylin Tso. Rational quantum secret sharing. *Scientific reports*, 8(1):11115, 2018. 1, 2, 4