Economic Keyless Semi-Quantum Point-to-Point Communication

(EKSQPC)

Kripa Adhikari, Sara Shajari

Carleton University

1 Introduction

1.1 Context

The following report analyzes the implementation and simulation of the Economic Keyless Semi-Quantum Point-to-Point Communication (EKSQPC) protocol in MATLAB. With the rapid growth in technology, especially in the field of quantum computers, our normal day-to-day portable devices such as mobile phones and laptops face a major threat. The classical encryption methods used in these portable devices are at risk because of quantum computers and their power to intervene these encryption methods. Previous solutions to these problems such as Semi-Quantum Computers or Quantum Key Distribution (QKD) are not ideal as QKD requires advanced quantum components. Hence, in this report we will be testing and reporting on a new proposed solution, EKSQPC. EKSQPC allows a sender (Alice) and a receiver (Bob) to securely communicate using entangled EPR pairs and Bell measurement-based attack detection. The system was implemented using object-oriented MATLAB code. Key components include EPR pair generation, Tele-Fetch technology, and the Measure-and-Replay Attack Detection (MRAD) concept.

1.2 Problem Statement

The main goal of this project is to simulate and analyze the EKSQPC protocol using MATLAB. This protocol aims to provide a solution for secure communication between classical and semi-quantum devices, without having to rely on costly quantum hardware. We aim to demonstrate the effectiveness of the EKSQPC protocol through real-life simulations.

1.3 Result

We implemented a fully modular simulation of EKSQPC using object-oriented MATLAB code. The system simulates entangled qubit transmission, Tele-Fetch-based bit inference, and MRAD-based attack detection. The implementation successfully detects attacks and infers transmitted bits, and scales from 10 to 1,000 qubits with consistent performance. A visualization using bar graphs is also incorporated to help interpret the results.

1.4 Outline

The rest of the report is structured as follows:

Section 2 presents background information

Section 3 presents methods and implementation of the project

Section 4 presents the evaluation

Section 5 presents the results

Section 6 presents the conclusion

2 Background Information

The EKSQPC protocol builds upon core principles of Quantum Computing this includes, quantum entanglement, Bell state measurements and semi-quantum communication models. One of the main focuses of the protocol is the use of entangled EPR pairs (Einstein-Podolsky-Rosen). EPR pairs allow the qubits to be correlated even if they are far apart. This is the main property that is utilized which helps Alice to infer Bob's actions and to detect if any attack or interference has occurred. The EKSQPC protocol integrates three other important mechanisms to support its solution:

1. OBP (One Bit Protocol): Alice sends one qubit from an entangled pair to Bob. Bob either measures it or reflects it back. Alice uses Bell measurements to infer Bob's action.
2. MRAD (Measure and Replay Attack Detection): The portion of qubits that Bob reflects back are probes. If Eve performs a measurement and re-sends a fake qubit, the entanglement is disrupted. Bell measurements on these probes reveal such attacks.
3. Tele-Fetch: A technique used by Alice to infer Bob's measurement result based on her knowledge of the initial EPR pair and the result of her Bell measurement. This allows for the secure retrieval of Bob's bit without direct transmission of the measurement result.

Together, OBP, MRAD, and Tele-Fetch form the EKSQPC protocol. EKSQPC ensures high qubit efficiency, low entanglement preservation time, and minimal quantum hardware requirements.

3 Methods and Implementation

The EKSQPC protocol solution was implemented in MATLAB using a modular and class-based architecture. This implementation only required a workable version of MATLAB along with the MATLAB support package for quantum computing (QIT). Hence, this simulation is accessible and reproducible in standard MATLAB environments. The structure of the project includes 3 classes:

- Alice Class (alice.m) : This class is responsible for generating EPR pairs, performing Bell measurements, implementing the Tele-Fetch technology and applying the Measure and Replay Attack Detection (MRAD). To do this, it uses basic quantum gates like Hadamard and CNOT, and reads the results from the simulated quantum state.

- Bob Class (bob.m) : This class simulates Bob's behaviour, where he is responsible for randomly choosing to reflect or measure each received qubit. (We do all of the randomized choosing using the rand() function).
- Main Script (main.m) : This file is responsible for coordinating the simulation. It sets everything up, loops through 10/1000 qubits, decides if Eve attacks, and tells Alice what to do based on Bob's choice. It also counts how many qubits were probes or data, how many attacks were caught, and then shows the results in a simple bar graph.

This setup reflects how the actual EKSQPC protocol is designed and keeps the code organized and easy to follow.

Initially, we decided to test each run based on an initial number of 10 qubits. This simulates the transmission of the 10 qubits. For each of the 10 qubits, Bob randomly decides to reflect (probe) or measure (data). If Bob reflects the qubit, Alice applied MRAD to detect an attack. If Bob measures the qubit, Alice uses Tele-Fetch logic to infer Bob's measurement result.

```
if action == 0
% For probes: MRAD is used to detect eavesdropping
probeCount = probeCount + 1;
attackDetected(k) = alice.checkMRAD(k, eveAttacks);  % Check for Eve's interference
fprintf("[Q%d - Probe] Attack Detected: %d\n", k, attackDetected(k));
else
% For data: Bob measures, Alice uses Tele-Fetch to infer the bit
dataCount = dataCount + 1;
[e1, e2] = alice.bellMeasurement(alice.eprStates{k}); % Alice performs Bell measurement
uB = alice.teleFetch(alice.ik_values(k), e1, e2);   % Alice infers Bob's measured bit
fprintf("[Q%d - Data] Bob measured. Alice inferred bit: %d\n", k, uB);
end
```

For further testing we decided to increase the number of qubits to 1000 to analyze how the EKSQPC protocol would execute in terms of larger scalability.

The main script also produces a graph for visual aid:

- The graph is a bar plot that summarizes the number of qubits, whether they are probes or data and also shows the number of detected attacks. This visual representation of the outcome makes it easier to analyze and interpret the results.

3.1 Program Execution

When the program starts, it first runs the main.m script, which sets the number of qubits and attack probability. It then creates an instance of the alice class and calls the generateEPRPairs(n) method to create entangled qubit pairs using Hadamard and CNOT gates. Next, it creates an instance of the bob class, which will later use the reflectOrMeasure() method to randomly decide whether each qubit will be reflected (for MRAD) or measured (for data). For each qubit, the program checks if Eve is attacking using rand() < p_attack. If the qubit is reflected, Alice runs checkMRAD(k, eveAttacks) to detect interference. If Bob measures it instead, Alice calls bellMeasurement(state) followed by teleFetch(ik, e1, e2) to infer the bit Bob saw. Throughout the loop, the program keeps count of probes, data, and attacks. Finally, it uses the built-in bar() function to generate a summary graph at the end of the run.

4 Evaluation

We evaluated the EKSQPC simulation at two scales (results are based on a specific test run):

- Small Scale (10 qubits): Detected 4 attacks; successfully inferred all data bits.
- Large Scale (1,000 qubits): Detected 237 attacks with a 60% attack probability, demonstrating reliable attack detection and protocol scalability.

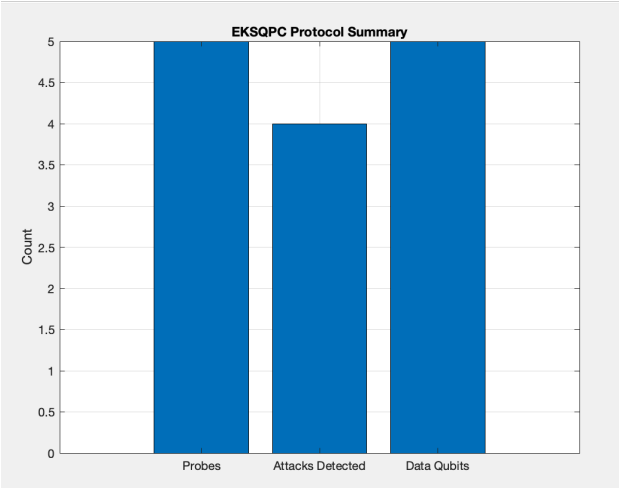
To ensure reliable and realistic testing, the simulation was designed to include randomized decision-making and attack behavior. We used MATLAB's rand() function to randomly

determine Bob's action for each qubit (either reflecting or measuring it) mimicking real-time uncertainty in communication. Additionally, we introduced an adjustable variable p_attack, which sets the probability that Eve will attempt to interfere with any given qubit. This variable is initialized to 0.6 or 60% to make sure our program can mimic real-life scenarios as closely as possible. This probabilistic attack model made the results more representative of real-world quantum communication conditions.

5 Results

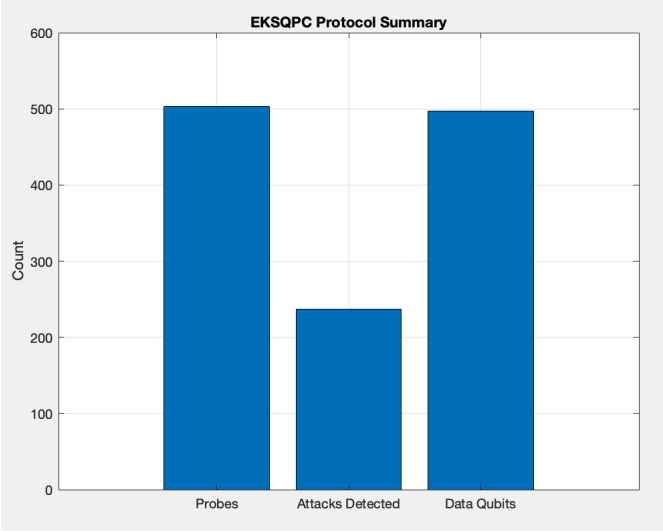
To evaluate the performance of our EKSQPC implementation, we conducted two simulations, one with 10 qubits and another with 1,000 qubits. For this specific test run, in the smaller-scale test, the protocol successfully identified 4 attacks out of 10 qubits, with Alice correctly inferring Bob's bit for each measured data qubit. This provided a clear demonstration of the protocol's core logic and flow.

```
=== EKSQPC DEMO ===
[Q1 - Probe] Attack Detected: 1
[Q2 - Data] Bob measured. Alice inferred bit: 1
[Q3 - Probe] Attack Detected: 0
[Q4 - Data] Bob measured. Alice inferred bit: 1
[Q5 - Data] Bob measured. Alice inferred bit: 1
[Q6 - Data] Bob measured. Alice inferred bit: 0
[Q7 - Data] Bob measured. Alice inferred bit: 0
[Q8 - Probe] Attack Detected: 1
[Q9 - Probe] Attack Detected: 1
[Q10 - Probe] Attack Detected: 1

=== SUMMARY ===
Detected 4 attacks out of 10 qubits
>>
```



For the larger-scale test, the system processed 1,000 qubits under a consistent 60% attack probability (p_attack = 0.6). Out of the total, 237 attacks were detected, closely aligning with expected probabilistic outcomes.

```
=== EKSQPC DEMO ===
[Q1 – Data] Bob measured. Alice inferred bit: 1
[Q2 – Data] Bob measured. Alice inferred bit: 1
[Q3 – Probe] Attack Detected: 0
[Q4 – Probe] Attack Detected: 1
[Q5 – Data] Bob measured. Alice inferred bit: 0
[Q6 – Probe] Attack Detected: 0
[Q7 – Data] Bob measured. Alice inferred bit: 0
[Q8 – Probe] Attack Detected: 1
[Q9 – Data] Bob measured. Alice inferred bit: 1
[Q10 – Data] Bob measured. Alice inferred bit: 0
[Q11 – Probe] Attack Detected: 0
[Q12 – Probe] Attack Detected: 0
[Q13 – Data] Bob measured. Alice inferred bit: 1
[Q14 – Probe] Attack Detected: 1
[Q15 – Data] Bob measured. Alice inferred bit: 1
[Q16 – Data] Bob measured. Alice inferred bit: 0
[Q17 – Probe] Attack Detected: 1
[Q18 – Data] Bob measured. Alice inferred bit: 1
[Q19 – Data] Bob measured. Alice inferred bit: 0
[Q20 – Data] Bob measured. Alice inferred bit: 0
[Q21 – Probe] Attack Detected: 0
[Q22 – Data] Bob measured. Alice inferred bit: 1
[Q23 – Probe] Attack Detected: 0
[Q24 – Probe] Attack Detected: 1
[Q25 – Probe] Attack Detected: 0
[Q26 – Probe] Attack Detected: 1
[Q27 – Data] Bob measured. Alice inferred bit: 0
[Q28 – Data] Bob measured. Alice inferred bit: 1
[Q29 – Probe] Attack Detected: 1
[Q30 – Data] Bob measured. Alice inferred bit: 1
[Q971 – Probe] Attack Detected: 1
[Q972 – Probe] Attack Detected: 0
[Q973 – Data] Bob measured. Alice inferred bit: 0
[Q974 – Data] Bob measured. Alice inferred bit: 0
[Q975 – Probe] Attack Detected: 0
[Q976 – Probe] Attack Detected: 0
[Q977 – Probe] Attack Detected: 1
[Q978 – Probe] Attack Detected: 0
[Q979 – Data] Bob measured. Alice inferred bit: 1
[Q980 – Probe] Attack Detected: 1
[Q981 – Probe] Attack Detected: 1
[Q982 – Probe] Attack Detected: 0
[Q983 – Probe] Attack Detected: 1
[Q984 – Data] Bob measured. Alice inferred bit: 0
[Q985 – Data] Bob measured. Alice inferred bit: 1
[Q986 – Data] Bob measured. Alice inferred bit: 0
[Q987 – Data] Bob measured. Alice inferred bit: 0
[Q988 – Data] Bob measured. Alice inferred bit: 0
[Q989 – Probe] Attack Detected: 1
[Q990 – Data] Bob measured. Alice inferred bit: 0
[Q991 – Data] Bob measured. Alice inferred bit: 0
[Q992 – Data] Bob measured. Alice inferred bit: 1
[Q993 – Probe] Attack Detected: 0
[Q994 – Probe] Attack Detected: 1
[Q995 – Probe] Attack Detected: 0
[Q996 – Probe] Attack Detected: 1
[Q997 – Data] Bob measured. Alice inferred bit: 0
[Q998 – Data] Bob measured. Alice inferred bit: 1
[Q999 – Probe] Attack Detected: 1
[Q1000 – Probe] Attack Detected: 1

=== SUMMARY ===
Detected 237 attacks out of 1000 qubits
>>
```



The corresponding bar graphs visually highlighted the split between probes, detected attacks, and data qubits. These results confirmed that the scalability of our simulation is effective and accurate.

As we executed the program multiple times with increasing qubit counts, we observed that the accuracy of attack detection improved with the scale. In the smaller test with 10 qubits, the number of detected attacks varied due to the limited sample size. However, as the number of qubits increased to 1,000, the detection rate began to stabilize and become more accurate with the probability defined by p_attack. This outcome supports our hypothesis that the EKSQPC protocol becomes more accurate with larger datasets.

6 Conclusion

6.1 Constraints & Limitations

While the EKSQPC protocol simulation worked as intended, there were a few limitations. First, the simulation assumes ideal quantum behavior without any noise or hardware imperfections, which is not always realistic in actual quantum environments. If we were able to simulate real-world noise then this protocol simulation would have been more accurate. Additionally, since the simulation is done entirely in MATLAB, it doesn't fully capture the physical behavior of actual quantum systems.

6.2 Future Work

For future work, we plan to expand this project by implementing the Rate Estimation Keyless Semi-Quantum Point-to-Point Communication (REKSQPC) protocol, which estimates the rate of attacks and enhances security. Another potential future improvement is integrating this project with quantum hardware simulators like IBM Qiskit to test performance in more realistic

environments. We also hope to create a better GUI to make the project more interactive and visually appealing.

6.3 Relevance

This project is directly relevant to COMP 4900 as it applies quantum communication theory to practical simulation, bridging the gap between classical and semi-quantum systems. It also utilizes some major quantum concepts that we have been learning throughout the course such as EPR pairs, Bell Measurement, QKD, Quantum Circuits, Teleportation etc.

6.4 Summary

In conclusion, this project successfully implemented and demonstrated the EKSQPC protocol using MATLAB. Through modular class-based design, we were able to simulate entangled qubit transmission, attack detection using MRAD, and secure bit inference with Tele-Fetch. Our results confirmed that the protocol works effectively, especially when scaled to larger qubit counts.

Team Contributions

Kripa Adhikari :

- Designed the project structure
- Wrote the project report
- Implemented the alice.m file for the MATLAB simulation

Sara Shajari :

- Designed the project structure
- Edited and proof-read the project report

- ❖ Implemented the main.m and bob.m files for the MATLAB simulation

References

[1] Lu, H., Barbeau, M., & Nayak, A. (2019a, January 11). *Keyless semi-quantum point-to-point communication protocol with low resource requirements*. Nature News. https://www.nature.com/articles/s41598-018-37045-0

[2] https://github.com/kripa01/EKSQPC_project