Entangled States and Bell's Inequality: A New Approach to Quantum Distance Bounding

Kevin Bogner
COSIC
Katholieke Universiteit Leuven
Leuven, Belgium
kevin.bogner@esat.kuleuven.be

Dave Singelée
COSIC
Katholieke Universiteit Leuven
Leuven, Belgium
dave.singelee@esat.kuleuven.be

Aysajan Abidin
COSIC
Katholieke Universiteit Leuven
Leuven, Belgium
aysajan.abidin@esat.kuleuven.be

***Abstract*—This paper introduces an entanglement-based Quantum Distance Bounding (QDB) protocol, whose security is derived from Bell's inequality violation. This protocol leverages the non-locality of quantum entanglement to enhance the integrity and security of quantum channels, providing device-independent assurances. We present both the theoretical framework and a practical implementation scenario using Qiskit. This study lays the groundwork for rigorous future analyses and the refinement of QDB protocols.**

***Index Terms*—quantum distance bounding, quantum communication, quantum entanglement, Bell's inequality, wireless security**

I. Introduction

Quantum entanglement and Bell's inequality are cornerstone concepts in quantum mechanics, crucial for the significant advancements in quantum information theory. This paper introduces a novel QDB protocol that capitalizes on the non-local properties of entangled states, substantially improving the security of the underlying protocol. The protocol presented here combines the fundamental concepts of existing QDB [1] protocols with the logic of the E91 [2] protocol.

Traditional Distance Bounding (DB) protocols rely on the rapid bit exchange to protect against complex relay attacks. Our method, in stark contrast, employs entangled qubits and relies on the observation of Bell's inequality as a stringent test to validate the security and integrity of quantum channels.

This protocol not only prevents undetected interceptions and replay attacks—common vulnerabilities in classical systems—but also uses the instantaneous state correlations of entangled qubits to establish secure communications. Our approach fully integrates the security advantages of quantum mechanics, particularly the reliance on Bell's inequality to verify quantum correlations, setting it apart from all existing quantum and classical DB protocols.

Following a review of existing technologies, the paper presents our innovative protocol and offers an informal security analysis, thereby providing a robust framework for future formal evaluations and optimisations.

II. Background and Preliminaries

This section lays out the foundational concepts necessary for understanding the QDB protocol presented in this paper. It begins with an overview of relay attacks— a primary security concern that DB protocols are designed to counteract. Then, the mechanics of classical DB protocols are explored, followed by an examination of the quantum aspects including qubits, entanglement, and Bell's inequality. Finally, we look deeper into the E91 protocol, from which our protocol is derived.

A. Relay Attacks

Relay attacks involve an attacker intercepting and relaying communication between two parties (commonly referred to as the verifier and the prover). The attacker intends to deceive the system into authorizing a transaction or access. In the context of DB, these attacks specifically aim to manipulate the timing of communication to falsify the apparent distance of the communicating parties. By doing so, an attacker can appear to be in a closer proximity than it actually is, potentially leading to unauthorized access. Classical and quantum DB protocols, such as the Hancke-Kuhn protocol [3] and the protocols displayed in [1], [4], [5], respectively, are designed to thwart such attacks by accurately measuring the maximum possible distance of the prover from the verifier and ensuring the integrity of the transmitted data.

B. Classical Distance Bounding Protocols

Classical DB protocols establish an upper limit on the physical distance between a verifier and a prover, while simultaneously authenticating the prover's identity, primarily to counteract relay attacks. One notable example is the Hancke-Kuhn protocol, which has influenced several subsequent QDB protocols [1]. Figure 1 displays the QDB protocol based on the Hancke-Kuhn protocol. This protocol employs a shared key (k_p) and a pseudo-random function (f_{k_p}) to generate two binary strings, each of length n.

During the distance measurement phase, the verifier transmits a sequence of challenges to the prover, who must then calculate and return a response based upon each received challenge. This exchange continues through n iterations, with the primary aim being the verification of each response against expected results and the timing of these responses. The integrity of the process relies on the assumption that the challenges are random and unforeseeable by the prover prior to dispatch, and that they propagate at the maximum feasible speed, namely the speed of light.

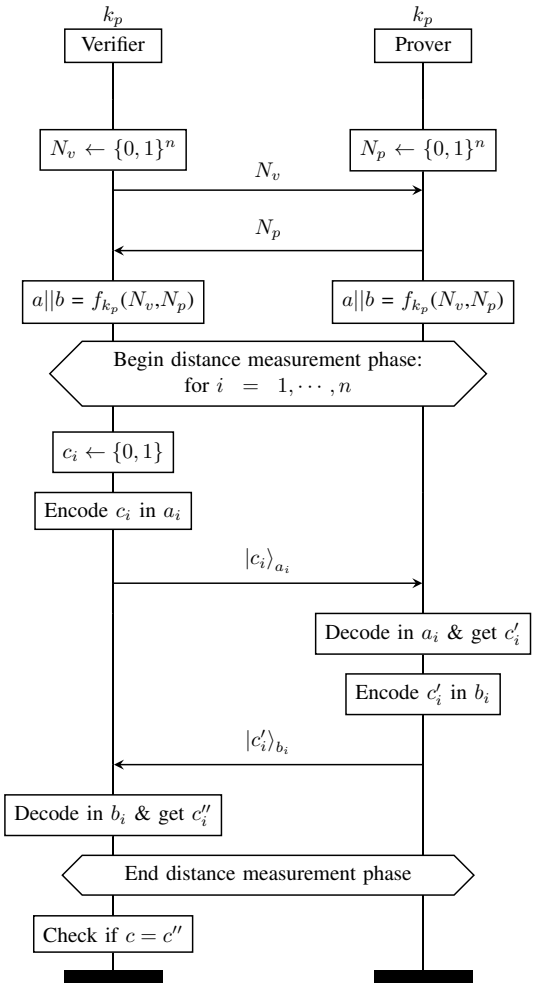

Fig. 1. QDB Protocol [1], based on Hancke-Kuhn.

C. Quantum Entanglement, Bell's Inequality, and the E91 Protocol

Quantum entanglement represents a profound phenomenon within quantum mechanics where the quantum state of each particle in a pair or group cannot be described independently of the state of the others, even when the particles are separated by large distances. A quintessential example of an entangled state is the singlet state [6]:

$$|\psi_s\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle),$$

which characterizes two qubits. In this state, if one qubit is measured to be in the state $|0\rangle$, the other is instantly known to be in the state $|1\rangle$, and vice versa, demonstrating a perfect anti-correlation that is pivotal for protocols such as E91.

In the analysis of quantum systems, observables related to measurements are denoted by Pauli matrices (X, Y, Z). For a two-qubit system involving parties named Alice and Bob, the observable $(\vec{\sigma} \cdot \vec{a}) \otimes (\vec{\sigma} \cdot \vec{b})$ is used, where $\vec{\sigma}$ represents the vector of Pauli matrices, and $\vec{a}$ and $\vec{b}$ are the vectors denoting chosen measurement directions on a unit sphere. The expected value of this observable in the singlet state is $-\vec{a} \cdot \vec{b}$, indicating perfect anti-correlation.

1) Bell's Inequality and Its Quantum Violation: Bell's inequality, specifically the Clauser-Horne-Shimony-Holt (CHSH) [7] variant, serves as a fundamental test to distinguish quantum entanglement from classical correlations. The CHSH inequality is formulated as:

$$|S| \leq 2,$$

where S is a parameter derived from the correlation measurements under four different settings. Quantum mechanics predicts that for entangled particles, $|S|$ can exceed 2, a result unattainable under classical theories constrained by local hidden variables.

To experimentally test the CHSH inequality using the singlet state, one measures the expectation values of the observables $X \otimes W$, $X \otimes V$, $Z \otimes W$, and $Z \otimes V$, where

$$W = \frac{1}{\sqrt{2}}(X + Z) \quad \text{and} \quad V = \frac{1}{\sqrt{2}}(-X + Z).$$

The CHSH correlation value, C, is then calculated as:

$$C = \langle X \otimes W \rangle - \langle X \otimes V \rangle + \langle Z \otimes W \rangle + \langle Z \otimes V \rangle,$$

with quantum entanglement predicting $C = 2\sqrt{2}$. This result, violating the CHSH inequality, confirms the non-classical nature of the correlations and underscores the essence of quantum mechanics and the presence of entanglement.

2) The E91 Protocol: Quantum Cryptography Using Entanglement: The E91 protocol, developed by Artur Ekert in 1991, leverages quantum entanglement to securely generate a shared random key between two parties (Alice and Bob) and allows for the detection of eavesdropping. This protocol is based on the principle that measurements on entangled particles are unpredictably random and strongly correlated.

In the E91 protocol, Alice and Bob repeatedly receive particles from entangled pairs, ensuring they share many such pairs. They independently choose one of three possible measurements for each particle received, either X, W or Z for Alice and W, Z or V for Bob, perform these measurements, and record the outcomes.

After many pairs have been measured, Alice and Bob publicly announce the measurement types they used for each pair, without revealing the measurement outcomes. They then use the results from the pairs where they chose corresponding observables ($W - W$ or $Z - Z$) to generate the key, as these should be correlated due to entanglement.

To detect eavesdropping, they compare a subset of their results to estimate the Quantum Bit Error Rate (QBER). A significant deviation from the expected correlation indicates potential eavesdropping. By applying the CHSH version of Bell's inequality to these results, they can further validate the security of their key; a violation of this inequality (i.e., a CHSH score exceeding 2) suggests that the results are

genuinely quantum and have not been tampered with by an eavesdropper. This protocol uses the fundamental properties of quantum mechanics to ensure secure communication.

III. Our QDB protocol based on E91

The QDB protocol operates through three key phases: the setup phase, the distance measurement phase, and the authentication phase, each essential for the secure and efficient functioning of the protocol. A detailed overview of the QDB protocol is depicted in Figure 2.

A. Setup Phase

Initially, the verifier and the prover agree on essential values critical for the protocol. Both parties generate unique values, a_v and a_p, which are crucial for the alignment and measurement of entangled particles as shown in Table I. They also share a common bitstring b, used for preparing and measuring the qubits during the exchange (Table II). This setup ensures that both the verifier and the prover are synchronized in their operations, facilitating accurate and secure communications.

TABLE I
Verifier and Prover Settings

Variable	Value	Setting
a_v	0	X
a_v	1	W
a_v	2	Z
a_p	0	W
a_p	1	Z
a_p	2	V

TABLE II
A Rule for Encoding Classical Bits as Qubits

Data	Computational (or $+$) Basis	Hadamard (or $\times$) Basis
0	$\|0\rangle$ (i.e., $\rightarrow$)	$\|+\rangle$ (i.e., $\nearrow$)
1	$\|1\rangle$ (i.e., $\uparrow$)	$\|-\rangle$ (i.e., $\nwarrow$)

B. Distance Measurement Phase

In this phase, the verifier creates a singlet pair of entangled particles, keeping one ($EP_{v,i}$) and sending the other ($EP_{p,i}$) to the prover. This step is critical as maintaining the entanglement over the distribution process is essential for the protocol's security. Upon receiving $EP_{p,i}$, both parties measure their respective particles using the predetermined settings $a_{v,i}$ and $a_{p,i}$, obtaining results m_i and m_i' respectively. The prover then encodes the measurement result m_i' onto a qubit in the state $|m'i\rangle\, b_i$ based on the shared bitstring b_i and sends it back to the verifier. A timer is started when $EP_{p,i}$ is sent and stopped when the encoded qubit is received and measured to obtain m_i'', providing the basis for calculating the distance bound.

C. Authentication Phase

The final phase validates the integrity of the exchanged values and the security of the quantum channel based on the combinations of measurement settings $a_{v,i}$ (verifier's settings) and $a_{p,i}$ (prover's settings). At the start of this phase, the prover sends its measurement settings $a_{p,i}$ to the verifier. Each round of measurements can require different checks depending on the settings used, as specified in Tables III and IV:

1) Value Matching Checks: For combinations listed in Table III, the verifier checks if the measured value m_i'' matches the expected m_i. This check confirms that the prover is at the claimed distance and that the quantum state has not been tampered with during transmission.

2) CHSH Checks: For combinations appearing in Table IV, the verifier calculates the CHSH value to assess the quality of entanglement of the particles involved. If the CHSH value exceeds the classical limit of $|S| \leq 2$, it indicates proper entanglement, verifying the integrity of the quantum states and ensuring that the operations were performed on the designated particles and not on decoy qubits.

3) No Check Rounds: If the combination of $a_{v,i}$ and $a_{p,i}$ does not correspond to any row in Tables III or IV, no specific check is performed. These rounds are used primarily for decoy or filler purposes to obscure the critical checks from potential eavesdroppers.

The successful completion of these checks allows the verifier to confidently establish an upper bound on the prover's distance, denoted as T_{max}. This comprehensive verification process ensures the security and accuracy of the distance measurement, crucial for preventing relay attacks and other forms of security breaches.

TABLE III
QDB Protocol Success Check

Combination	Verifier's Meas. (a_v)	Prover's Meas. (a_p)
1	W ($a_v = 1$)	W ($a_p = 0$)
2	Z ($a_v = 2$)	Z ($a_p = 1$)

TABLE IV
CHSH Measurement Combinations

Term in CHSH Expression	Measurement Settings	
	Verifier's Setting a_v	***Prover's Setting*** a_p
$\langle X \otimes W \rangle$	X ($a_v = 0$)	W ($a_p = 0$)
$\langle X \otimes V \rangle$	X ($a_v = 0$)	V ($a_p = 2$)
$\langle Z \otimes W \rangle$	Z ($a_v = 2$)	W ($a_p = 0$)
$\langle Z \otimes V \rangle$	Z ($a_v = 2$)	V ($a_p = 2$)

IV. Security Analysis

This section provides a preliminary, informal security analysis of our proposed QDB protocol, which integrates principles from the E91 protocol. While the security of the E91 protocol has been well-documented [8], previous analyses of QDB protocols have often lacked formal rigor [1], [4], [5], contrasting sharply with the more systematic approaches employed in

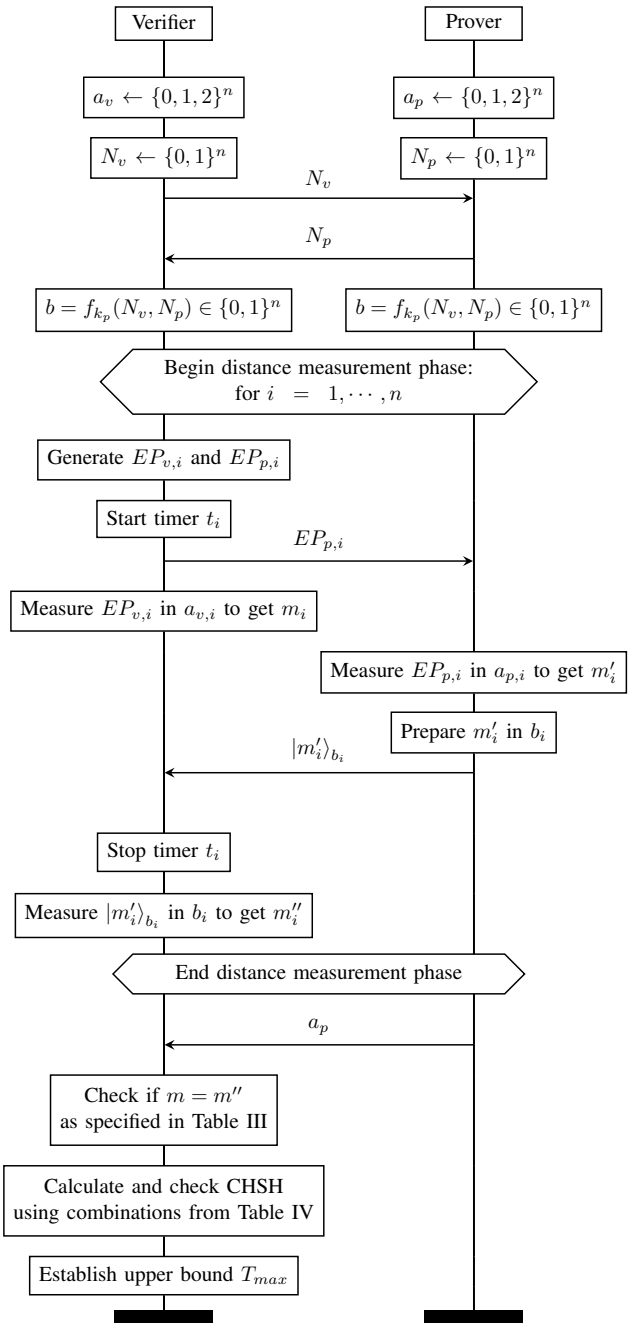

Fig. 2. QDB Protocol employing entanglement and Bell's inequality.

classical DB protocols [9]. The objective of this analysis is to identify the manner in which our protocol addresses a number of potential attack vectors.

A. Distance Fraud Attacks

Distance fraud involves a dishonest prover manipulating the communication process to appear closer to the verifier than they actually are. Our protocol combats this risk through the precise timing of quantum state responses. The integrity of these measurements is crucial, as their validity depends heavily on precise timing. Any attempt by the prover to falsify their distance would result in noticeable discrepancies in the timing of the quantum responses, effectively neutralizing the fraud attempt.

B. Relay and Mafia Fraud Attacks

Relay and Mafia fraud attacks involve an intermediary who manipulates the communication between the original parties. Our QDB protocol addresses these threats using quantum entanglement. Any attempt by an intermediary to measure or replicate the quantum states involved would disrupt their entanglement. The occurrence of this disturbance would be reflected in the outcomes expected under Bell's inequality, thereby providing clear evidence of tampering.

C. Reflection Attacks

A specific threat at the physical layer is the reflection attack, where an attacker tries to bypass security measures by mirroring the communication process back to the sender. Such actions are also detectable by our protocol, which leverages the no-cloning theorem [10] in quantum mechanics. This theorem posits that it is impossible to create an exact copy of an arbitrary unknown quantum state. Therefore, any attempt to clone the involved qubits would inevitably alter their state, revealing the attack through failed Bell's inequality tests, and thus ensuring the integrity of the quantum states.

D. Future Work

While this informal analysis highlights the robust security features of our QDB protocol, comprehensive and formal security proofs are necessary to fully validate its effectiveness under various attack scenarios. Future research should focus on developing these proofs and conducting empirical tests that more realistically simulate quantum interactions and potential attacks. Advancing the practical implementation and theoretical understanding of the protocol's security dynamics is crucial for its adoption and reliability in real-world applications.

V. Conclusion

This paper presented a QDB protocol that harnesses quantum entanglement and relies on the observation of Bell's inequality, setting a new benchmark in the security of QDB protocols. By integrating these quantum principles, our protocol not only fortifies communication channels against eavesdropping and other advanced threats but also introduces a device-independent method to verify channel integrity.

Our entanglement-based approach represents a pioneering approach to fully exploiting the security potential of quantum mechanics through the utilisation of entanglement in conjunction with the observation of Bell's inequality. It offers a comprehensive set of protective measures and resilience against sophisticated attacks.

Future directions will include:

1) Formal Security Analysis: Conducting a comprehensive, mathematical security analysis to affirm the protocol's theoretical robustness against diverse attack vectors.

2) Optimization and Practical Implementation: Refining the protocol to improve efficiency, reduce demands on resources, and adapt it for broad, practical application in various settings.

3) Experimental Validation: Implementing empirical tests to substantiate theoretical claims and adjust the protocol based on practical feedback and scenarios.

4) Integration with Existing Systems: Examining potential synergies with existing Quantum Key Distribution (QKD) systems and other cryptographic frameworks to enhance overall security infrastructure.

In conclusion, the proposed QDB protocol aims to enhance the security of quantum communications by integrating quantum entanglement and the principles of Bell's inequality. While our approach seeks to leverage the unique properties of quantum mechanics for improved security, it represents a step towards more robust quantum communication technologies rather than a comprehensive solution. Future work will be crucial in determining the practicality and effectiveness of this protocol within the broader landscape of secure communications. This approach holds potential for meaningful contributions to the field, particularly in enhancing the resilience of communications against evolving cyber threats.

ACKNOWLEDGMENT

This work was supported in part by CyberSecurity Research Flanders with reference number VR20192203, and by the European Commission through the Horizon Europe research and innovation programme under the project "Quantum Security Networks Partnership" (QSNP, grant agreement No 101114043).

REFERENCES

[1] A. Abidin, "Quantum distance bounding," in *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks*, 2019, pp. 233–238.

[2] A. K. Ekert, "Quantum cryptography based on bell's theorem," *Physical review letters*, vol. 67, no. 6, p. 661, 1991.

[3] G. P. Hancke and M. G. Kuhn, "An RFID distance bounding protocol," in *First international conference on security and privacy for emerging areas in communications networks (SECURECOMM'05)*. IEEE, 2005, pp. 67–73.

[4] A. Abidin, E. Marin, D. Singelée, and B. Preneel, "Towards quantum distance bounding protocols," in *Radio Frequency Identification and IoT Security: 12th International Workshop, RFIDSec 2016, Hong Kong, China, November 30–December 2, 2016, Revised Selected Papers 12*. Springer, 2017, pp. 151–162.

[5] A. Abidin, K. Eldefrawy, and D. Singelee, "Entanglement-based mutual quantum distance bounding," *arXiv preprint arXiv:2305.09905*, 2023.

[6] M. A. Nielsen and I. L. Chuang, *Quantum computation and quantum information*. Cambridge university press, 2010.

[7] J. F. Clauser, M. A. Horne, A. Shimony, and R. A. Holt, "Proposed experiment to test local hidden-variable theories," *Physical review letters*, vol. 23, no. 15, p. 880, 1969.

[8] J. Barrett, L. Hardy, and A. Kent, "No signaling and quantum key distribution," *Physical review letters*, vol. 95, no. 1, p. 010503, 2005.

[9] G. Avoine, M. A. Bingöl, I. Boureanu, S. Čapkun, G. Hancke, S. Kardaş, C. H. Kim, C. Lauradoux, B. Martin, J. Munilla *et al.*, "Security of distance-bounding: A survey," *ACM Computing Surveys (CSUR)*, vol. 51, no. 5, pp. 1–33, 2018.

[10] W. K. Wootters and W. H. Zurek, "A single quantum cannot be cloned," *Nature*, vol. 299, no. 5886, pp. 802–803, 1982.

[11] A. Kardashin, "E91 quantum key distribution protocol," GitHub, available: https://github.com/qiskit-community/qiskit-community-tutorials/blob/master/awards/teach_me_qiskit_2018/e91_qkd/e91_quantum_key_distribution_protocol.ipynb [Accessed: Apr. 29, 2024].

[12] K. Bogner, "qdb_e91," GitHub, available: https://github.com/kevinbogner/qdb_e91 [Accessed: May 20, 2024].

APPENDIX

This appendix includes the Python code for our protocol. The method for calculating the CHSH value is based on [11]. The code is available on GitHub [12].

1) Preparation Phase: Initializes the quantum registers and defines the singlet state preparation.

```
qr, cr = QuantumRegister(2), ClassicalRegister(2)
singlet = QuantumCircuit(qr, cr)
singlet.h(qr[0])
singlet.cx(qr[0], qr[1])
numberOfSinglets = 500
av = [random.randint(0, 2) for i in range(
    ↪ numberOfSinglets)]
ap = [random.randint(0, 2) for i in range(
    ↪ numberOfSinglets)]
b = [random.randint(0, 1) for i in range(
    ↪ numberOfSinglets)]
```

2) Measurement Configurations: Details the circuit configurations for different measurement bases a_v and a_p.

```
measureAV0 = QuantumCircuit(qr, cr)
measureAV0.h(qr[0])
measureAV0.measure(qr[0], cr[0])
measureAV1 = QuantumCircuit(qr, cr)
measureAV1.s(qr[0])
measureAV1.h(qr[0])
measureAV1.t(qr[0])
measureAV1.h(qr[0])
measureAV1.measure(qr[0], cr[0])
measureAV2 = QuantumCircuit(qr, cr)
measureAV2.measure(qr[0], cr[0])

measureAP0 = QuantumCircuit(qr, cr)
measureAP0.s(qr[1])
measureAP0.h(qr[1])
measureAP0.t(qr[1])
measureAP0.h(qr[1])
measureAP0.measure(qr[1], cr[1])
measureAP1 = QuantumCircuit(qr, cr)
measureAP1.measure(qr[1], cr[1])
measureAP2 = QuantumCircuit(qr, cr)
measureAP2.s(qr[1])
measureAP2.h(qr[1])
measureAP2.tdg(qr[1])
measureAP2.h(qr[1])
measureAP2.measure(qr[1], cr[1])

avBases = [measureAV0, measureAV1, measureAV2]
apBases = [measureAP0, measureAP1, measureAP2]
```

3) Circuit Execution and Pattern Matching: Executes the combined circuits and captures results for CHSH.

```
for i in range(numberOfSinglets):
    circuitName = str(i) + ":V" + str(av[i]) + "_P"
        ↪ + str(ap[i])
    combined_circuit = singlet.compose(avBases[av[i
        ↪ ]]).compose(apBases[ap[i]])
    combined_circuit.name = circuitName
    circuits.append(combined_circuit)

aer_sim = Aer.get_backend("aer_simulator")
result = aer_sim.run(transpile(circuits, aer_sim),
    ↪ shots=1, memory=True).result()

abPatterns = [re.compile("00$"), re.compile("01$"),
    ↪ re.compile("10$"), re.compile("11$")]

for i in range(numberOfSinglets):
    res = list(result.get_counts(circuits[i]).keys()
        ↪ )[0]
    if abPatterns[0].search(res):
        VerifierResults.append(-1)
        ProverResults.append(-1)
    if abPatterns[1].search(res):
        VerifierResults.append(1)
        ProverResults.append(-1)
    if abPatterns[2].search(res):
        VerifierResults.append(-1)
        ProverResults.append(1)
    if abPatterns[3].search(res):
        VerifierResults.append(1)
        ProverResults.append(1)
```

4) Measurement Results Processing: Retrieves the measurement results from the quantum memory.

```
for i in circuits:
    memory = result.get_memory(i)
    m_result = int(memory[0][0])
    m_prime_result = int(memory[0][1])
    m.append(m_result)
    m_prime.append(m_prime_result)
```

5) Encoding and Decoding Messages: Defines functions to encode and decode messages.

```
def encode_message(bits, bases, n):
    for i in range(n):
        qc = QuantumCircuit(1, 1)
        if bases[i] == 0:
            if bits[i] == 0:
                pass
            else:
                qc.x(0)
        else:
            if bits[i] == 0:
                qc.h(0)
            else:
                qc.x(0)
                qc.h(0)
        qc.barrier()
        message.append(qc)
    return message

def decode_message(message, bases, n, draw_circuit=
    ↪ False):
    backend = Aer.get_backend("aer_simulator")
    for q in range(n):
        if bases[q] == 1:
            message[q].h(0)
        message[q].measure(0, 0)
        if draw_circuit:
            print(f"Circuit {q}:")
            display(message[q].draw(output="mpl"))

        aer_sim = Aer.get_backend("aer_simulator")
        result = aer_sim.run(
            transpile(message[q], aer_sim), shots=1,
                ↪  memory=True
        ).result()
        measured_bit = int(result.get_memory()[0])
        measurements.append(measured_bit)
    return measurements

message = encode_message(m_prime, b,
    ↪ numberOfSinglets)
m_prime_two = decode_message(message, b,
    ↪ numberOfSinglets, False)
```

6) CHSH and Protocol Verification: Calculates the CHSH correlation value and compares the results for protocol validation.

```
def chsh_corr(result):
    countA0B0, countA0B2, countA2B0, countA2B2 =
        ↪ ([0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0,
        ↪ 0], [0, 0, 0, 0])
    for i in range(numberOfSinglets):
        res = list(result.get_counts(circuits[i]).
            ↪ keys())[0]
        index = int(res, 2)
        if av[i] == 0 and ap[i] == 0:
            countA0B0[index] += 1
        elif av[i] == 0 and ap[i] == 2:
            countA0B2[index] += 1
        elif av[i] == 2 and ap[i] == 0:
            countA2B0[index] += 1
        elif av[i] == 2 and ap[i] == 2:
            countA2B2[index] += 1

    def expectation(counts):
        total = sum(counts)
        return (counts[0] - counts[1] - counts[2] +
            ↪ counts[3]) / total if total != 0 else
            ↪  0

    expect00 = expectation(countA0B0)
    expect02 = expectation(countA0B2)
    expect20 = expectation(countA2B0)
    expect22 = expectation(countA2B2)
    corr = expect00 - expect02 + expect20 + expect22
    return corr

corr = chsh_corr(result)
print("CHSH correlation value: " + str(round(corr,
    ↪ 3)))

def compare_results(m, m_prime_two, av, ap):
    for i in range(len(av)):
        if (av[i] == 1 and ap[i] == 0) or (av[i] ==
            ↪ 2 and ap[i] == 1):
            if m[i] == m_prime_two[i]:
                matches += 1
            else:
                mismatches += 1
    return matches, mismatches

matches, mismatches = compare_results(m, m_prime_two
    ↪ , av, ap)
print("Number of matches:", matches)
print("Number of mismatches:", mismatches)
```