Complete Fairness in Secure Two-party Computation from Quantum One-Wayness

Arpita Maitra[1], Goutam Paul[2], Asim K. Pal[3], Asmita Samanta[2], and Hridam Basu[4]

[1] Institute for Advancing Intelligence, TCG Centres for Research and Education in Science and Technology (TCG CREST), India
arpita.maitra@tcgcrest.org
[2] Indian Statistical Institute, Kolkata, India
goutam.paul@isical.ac.in, asmitasamanta6@gmail.com
[3] Management Information Systems Group, Indian Institute of Management Calcutta, India
asim@iimcal.ac.in
[4] Rump Labs
hridam.basu@gmail.com

Abstract. In 1997, Mayers, and Lo and Chau, in their papers proved that if one of the parties is malicious, it is not possible to achieve unconditional security in quantum bit-commitment (Phy. Rev. Lett, 1997). Moreover, Lo showed that it is not possible to achieve unconditional security in two-party quantum computation (Phy. Rev. A, 1997). However, restricting the computational power of the adversary, via assumptions like the existence of one-wayness, it has been shown that secure quantum bit-commitment is indeed possible by Dumais, Mayers, and Salvail (EUROCRYPT 2000). In their paper, they showed that with the assumption of a one-way permutation that is secure against any polynomial-time quantum adversary, one can achieve a secure quantum bit-commitment scheme. Although security is guaranteed, complete fairness remains an issue in quantum two-party computation for the real-world framework.

In this paper, we show for the first time that secure two-party quantum computation with complete fairness is achievable for the same class of functions as studied by Gordon, Hazay, Katz, and Lindell (STOC 2008) in the classical domain. This seems to stand in sharp contrast to the impossibility result of Ben-Or, Crepeau, Gottesman, Hassidim, and Smith (FOCS 2006). However, they considered a malicious quantum adversary with unconditional computational power and used a broadcast channel. On the other hand, we use the hybrid model idea of Gordon, Hazay, Katz, and Lindell with a non-simultaneous channel and the idea of composition by Ran Canetti (Journal of Cryptology, 2000) to achieve complete fairness in the quantum domain, under the assumption of a computationally bounded adversary.

At first, we design secure two-party computation protocols in a hybrid model, using a trusted third party. Then we prove how these protocols achieve complete fairness in the hybrid model. Finally, we construct a secure two-party communication protocol and show how we can use this to remove the trusted third party of the hybrid model and achieve both security and complete fairness in the real-world model. In this context, we devise a zero-knowledge proof and use it as an important tool in our protocol.

Keywords: Embedded XOR · Fairness · Millionaires' problem · Secure quantum computation.

1 Introduction and Related Work

In a secure two-party computation, two parties or players want to compute a particular function of their inputs, keeping the inputs secret from each other. They are only allowed to obtain the function's output, preserving some security notions under certain adversarial models.

Yao's millionaires' problem [53], or the 'greater than' function, is one of the examples of the secure two-party computation (2PC). It deals with two millionaires, Alice and Bob, who are interested in finding who amongst them is richer, without revealing their actual wealth to each other. Much effort has been given to solve this problem in the quantum domain [23, 25, 44, 52, 55], all of which analyzed the security issues against several eavesdropping strategies. [25] dealt with the problem with a semi-honest party. In [55], the millionaires' problem is studied considering a continuous variable. [23] exploited the idea of quantum key distribution to solve the problem. [44] proposed the use of Bell states to solve this problem. Their protocol

also exploits a third party to assist the players. [52] showed the vulnerability of their protocol if the third party is disloyal. However, none of these works [10, 23, 25, 29, 44, 51, 52, 55] analyze the security issues considering malicious players.

In the classical domain, the subsequent work of [20, 21] showed that any function over polynomial-size domains that does not contain an "embedded XOR" can be converted into the greater than function, or more specifically, into the millionaires' problem. Hence, the millionaires' problem covers all functions without embedded XOR. [20, 21] also studied a function that has an embedded XOR, namely, a function that simply checks whether the inputs chosen by two players (from a specified domain) are equal or not.

Exploiting the idea of [20], we, for the first time, design two quantum protocols for these two distinct sets of functions. We also analyze security issues when players are malicious, unlike existing quantum protocols [10, 23, 25, 29, 44, 51, 52, 55].

1.1 Motivation for a Quantum Solution

There are primarily two reasons for why we move to a quantum solution.

- In the classical domain, to obtain a real-world protocol from the hybrid protocol of [20], one needs a secure 2PC with abort functionality, where OT is necessary. Classically, OT from one-way functions alone is black-box impossible and requires key agreement [24]. Unlike classical domain, OT in the quantum domain can be achieved from one-wayness alone [2, 22].
- For the classical MAC used in [20], both parties need a shared secret key which requires a public key encryption. However, most of the well-known public key encryptions are vulnerable due to Shor's algorithm [42]. We solve this problem by suitable use of quantum entanglement along with a keyless Algebraic Manipulation Detection (AMD) Code [13].

1.2 Overview of the Quantum Solution and Its Design Challenges

We achieve the secure quantum two-party computation protocols in three steps: (1) designing a hybrid model with complete fairness, (2) constructing the quantum two-party sub-protocol, and (3) securely composing these two steps. In the following, we state the challenges to each of the three steps and how our contributions circumvent those challenges.

Hybrid Model with Complete Fairness. The two main challenges in designing such a model are how to detect the forgery and how to achieve fairness under early abort. [20]'s classical hybrid model relies on MAC signatures for forgery detection and the structure of secret sharing for fairness under early abort. In the quantum domain, we replace MAC-based forgery detection with entanglement-based security along with Algebraic Manipulation Detection (AMD) Code [13]. Shares are distributed as halves of Bell states, and any tampering by a Byzantine adversary is either reflected in the measued Bell states or in the verification process of AMD code.

Constructing the Quantum Two-Party Sub-Protocol. There are several impossibility results in the quantum domain in this context. In 1997, [35] and [31] showed that the existence of an unconditionally secure quantum bit commitment scheme is impossible in the presence of a dishonest party. Also in 1997, [30] demonstrated that it is not possible to achieve quantum secure two-party computation (2PC) without any computational hardness assumptions in the presence of a dishonest party. In 2006, [4] showed that one can achieve secure quantum MPC if the number of dishonest parties in an n-party MPC protocol is strictly less than $\frac{n}{2}$, thus making it impossible to establish a secure protocol with one dishonest out of two parties. Many researchers have tried to exploit the quantum mechanical effect [43] to solve the problems of secure MPC [10, 23, 25, 29, 44, 52, 55]. In recent years, there have been many works in this area, which have shown that with the assumption of the existence of a quantum one-way function or puzzle, i.e., restricting the computational power of the quantum adversary, one can get a secure quantum bit-commitment [15,17,26,27], hence a secure quantum oblivious transfer (OT) protocol [14,15]. Using these ideas [15], we designed a protocol for solving the millionaires' problem for a polynomial-time bounded quantum adversary.

For our real-world protocol from the hybrid one, simply having a secure quantum 2PC is insufficient. We also need a sub-protocol that must reproduce all dealer functionalities, such as: computing f, embedding

the output at the correct round, and filling remaining rounds with $\perp$. We give an explicit construction (Section 4.3) that achieves this using the quantum OT as a sub-protocol, additional modifications to the share generation process, and some checking conditions in the reconstruction phase. For this, we have also used a custom zero-knowledge proof (ZKP (Section 4.3.3, Section 6), to ensure the consistency of inputs across rounds.

Composing the Protocols in Steps 1 and 2. The main technical challenge here is that verifying the simulator construction (i.e., building a hybrid-model adversary A_π from a real-world adversary A via the ideal-world simulator S_ρ) remains valid, when all parties and adversaries are quantum polynomial-time, auxiliary inputs may be quantum states, and indistinguishability is against quantum distinguishers. One may ask what is new here. Our proof follows Canetti's structure [8] but instantiates each step with quantum objects (Theorems 4, 5 and 6). We acknowledge that a fully general quantum composition theorem (e.g., in the quantum UC framework of Unruh [45]) would subsume this; we provide a self-contained proof for our specific sequential-composition setting to keep the paper self-contained.

2 Preliminaries

2.1 Functionality

In the classical domain and in a two-party setting, a functionality $\mathcal{F} = \{f_\lambda\}_{\lambda\in\mathbb{N}}$ is a sequence of randomized processes, where λ is the security parameter and f_λ maps pairs of inputs to pairs of outputs (one for each party). Explicitly, we can write $f_\lambda = (f_\lambda^1, f_\lambda^2)$, where f_λ^1 (resp. f_λ^2) represents the output of the first party, say P_1 (resp. output of the second party, say P_2). The domain of f_λ is $X_\lambda \times Y_\lambda$, where X_λ (resp. Y_λ) denotes the possible inputs of the first (resp. second) party. If the domain sizes $|X_\lambda|$ and $|Y_\lambda|$ are polynomial in λ, then we say that $\mathcal{F}$ is defined over polynomial-size domains. If each f_λ is deterministic, we say that each f_λ as well as the collection $\mathcal{F}$ is a function.

2.2 Two-Party Computation

In classical domain, the two-party computation of a functionality $\mathcal{F} = \{f_\lambda^1, f_\lambda^2\}$ is defined as follows. P_1 is holding 1^λ and a input $x \in X_\lambda$ and a party P_2 is holding 1^λ and a input $y \in Y_\lambda$, then the joint distribution of the outputs of the parties is statistically close to $(f_\lambda^1(x, y), f_\lambda^2(x, y))$.

2.3 Ideal vs. Real World model

In classical multi-party computation (MPC), the security of a protocol depends on the comparison between two extreme models, called *ideal world model* and *real world model*. In the ideal world model, there is an incorruptible trusted party who computes the function after obtaining the inputs from the players and announces the output to each of the players. Whereas, in the real-world model, the players execute a protocol for computing the functional value, along with keeping their inputs secret [53]. However, such a scheme requires garbled circuit [54] or homomorphic encryption [38] that are hard to implement in practice. On the other hand, by relaxing the constraint, i.e., by hybridizing the ideal and the real world model, one can achieve the desired security proof of an MPC protocol. In [20, 21], the authors showed how one can compute some functions with complete fairness exploiting such a hybridized model, where the incorruptible party becomes offline after distributing the shares of the functional value among the players, like a *dealer* in secret sharing [41]. After this part, the protocol is executed like real world model. This approach is actually taken to bypass a garbled circuit or homomorphic encryption.

In the quantum domain, to the best of our knowledge, there does not exist this type of idea. Thus, one may feel difficulty in accepting it as a secure MPC, as the computation has been done through an honest party. One may argue that this honest party itself can announce the outcome to the players. However, one should note that if there is no trusted party, then no function is securely computable in the context of MPC - be it in the classical or in the quantum domain. Too strict an assumption, such as that there does not exist any trusted party, makes the security proof fail. In [30], the author showed the impossibility of computing some functions in two sided two party setting in the quantum domain when one of the parties is malicious.

Thus, an interesting open problem of practical interest is whether one can find any function that can be securely computed by relaxing the assumption slightly in the context of MPC. In this paper, analogous to the classical domain [20, 21], we show that with the idea of a hybrid model, one could securely compute two classes of functions - the millionaire's problem and the embedded XOR problem.

2.4 Adversarial Model and Security Notions

There are two types of adversaries: Fail-Stop and Byzantine. In the fail-stop setting, each party follows the protocol as directed, except that it may choose to abort at any time, and a party is assumed not to change its input when running the protocol. On the other hand, in a Byzantine setting, a deviating party may behave arbitrarily. It may change the inputs or choose to abort. Since a Byzantine adversary covers all the characteristics of a fail-stop adversary, it is very natural to consider only the Byzantine setting. If a protocol is secure against a Byzantine adversary, it must be secure against a fail-stop adversary. Hence, throughout the paper, we analyze the security issues against a Byzantine adversary only.

For fairness, we need to introduce some terminology. Let us assume that P_1 begins by holding an input $x \in X$ and P_2 begins by holding an input $y \in Y$, $z \in \{0,1\}^*$ is the auxiliary input of the adversary, and λ is the security parameter.
Let $\{IDEAL_{\mathcal{F},\mathcal{S}(z),\lambda}(x,y)\}_{(x,y)\in X\times Y, z\in\{0,1\}^*}$ represent a pair of two random variables denoted by $VIEW$ and OUT, where $VIEW_{ideal}(x,y)$ represents the output of the party who is corrupted by the adversary $\mathcal{S}$ and $OUT_{ideal}(x,y)$ represents the output of the honest party in the ideal world. Thus, we can write

$$\{IDEAL_{\mathcal{F},\mathcal{S}(z),\lambda}(x,y)\}_{(x,y)\in X\times Y, z\in\{0,1\}^*} = (VIEW_{ideal}(x,y), OUT_{ideal}(x,y)).$$

Similarly, let $\{REAL_{\mathbf{\Pi},\mathcal{A}(\mathbf{z}),\lambda}(x,y)\}_{(x,y)\in X\times Y, z\in\{0,1\}^*}$ represents a pair of two random variables, namely $VIEW_{real}(x,y)$ and $OUT_{real}(x,y)$, where $VIEW_{real}(x,y)$ denotes the random variable in real world consisting of the view of the player corrupted by the adversary $\mathcal{A}$ and $OUT_{real}(x,y)$ represents the random variable consisting of the output of the honest party in the real world [28].

Definition 1. *A protocol $\mathbf{\Pi}$ is said to securely compute a functionality $\mathcal{F}$ with complete fairness if for every adversary $\mathcal{A}$, there exists an adversary, $\mathcal{S}$, with same computational complexity in the ideal model such that*

$$\{IDEAL_{\mathcal{F},\mathcal{S}(z),\lambda}(x,y)\}_{(x,y)\in X\times Y, z\in\{0,1\}^*} \equiv \{REAL_{\mathbf{\Pi},\mathcal{A}(\mathbf{z}),\lambda}(x,y)\}_{(x,y)\in X\times Y, z\in\{0,1\}^*}.$$

Here, in our case, we are only considering the quantum adversaries having polynomial-time bounded computational power.

In our hybrid model, the fairness condition is as follows.

Definition 2. *A protocol $\mathbf{\Pi}$ is said to securely compute a functionality $\mathcal{F}$ with complete fairness if for every adversary $\mathcal{A}$, there exists an adversary, $\mathcal{S}$, with same computational complexity in the ideal model such that*

$$\{IDEAL_{\mathcal{F},\mathcal{S}(z),\lambda}(x,y)\}_{(x,y)\in X\times Y, z\in\{0,1\}^*} \equiv \{HYBRID_{\mathbf{\Pi},\mathcal{A}(\mathbf{z}),\lambda}(x,y)\}_{(x,y)\in X\times Y, z\in\{0,1\}^*}.$$

Here, $REAL$ is replaced by $HYBRID$, which is the random variable consisting of the view ($VIEW$) of the adversary and the output (OUT) of the honest party in the hybrid world in the same manner as above.

2.5 Revisiting the Protocol for Solving Millionaires' Problem with Complete Fairness [20]

In this section, we first revisit the protocol to solve the millionaires' problem, or more precisely, the greater than function, proposed by [20, 21]. Note that this protocol covers all functions that do not contain an embedded XOR. Let us denote two players by P_1 and P_2. As we deal with a hybrid model, there is a trusted party whom we call the dealer. Suppose P_1 has the secret i and P_2 has the secret j, $1 \le i \le N$, $1 \le j \le N$, where $N = N(\lambda)$ is the domain size of each input. The dealer gives an ordered list $X = \{x_1, x_2, \ldots, x_N\}$ to P_1 and another ordered list $Y = \{y_1, y_2, \ldots, y_N\}$ to P_2. Then P_1 sends x_i to the dealer, and P_2 sends y_j to the dealer. Let f be a deterministic function that maps $X \times Y \to \{0,1\} \times \{0,1\}$. The function $f(x_i, y_j)$ can be defined as a pair of outputs, i.e., $f(x_i, y_j) = (f_1(x_i, y_j), f_2(x_i, y_j))$, where $f_1(x_i, y_j)$ is the output of the

first party P_1 and $f_2(x_i, y_j)$ is the output of the second party P_2. For the millionaires' problem, the function is defined as follows [20, 21]. For $w = 1, 2$,

$$f_w(x_i, y_j) = \begin{cases} 1 & \text{if } i > j; \\ 0 & \text{if } i \leq j. \end{cases} \tag{1}$$

The protocol proceeds in a series of N iterations. The dealer creates two sequences $\{a_l\}$ and $\{b_l\}$, $l = 1, 2, \ldots, N$, as follows.

$$a_i = b_j = f_1(x_i, y_j) = f_2(x_i, y_j).$$

For $l \neq i$, $a_l = \perp$ and for $l \neq j$, $b_l = \perp$.

Next, the dealer splits the secret a_l into the shares a_l^1 and a_l^2, and the secret b_l into the shares b_l^1 and b_l^2, so that $a_l = a_l^1 \oplus a_l^2$ and $b_l = b_l^1 \oplus b_l^2$, and gives the shares $\{(a_l^1, b_l^1)\}$ to P_1 and the shares $\{(a_l^2, b_l^2)\}$ to P_2. In each round l, P_2 sends a_l^2 to P_1, who, in turn sends b_l^1 to P_2. P_1 learns the output value $f_1(x_i, y_j)$ in iteration i, and P_2 learns the output value $f_2(x_i, y_j)$ in iteration j. In a round $l \neq i$ P_1 outputs $\perp$ and in a round $l \neq j$ P_2 output $\perp$. As we require three elements, 0, 1 and $\perp$, we define 0 by 00, 1 by 11 and $\perp$ by 01. Note that the dealer who will distribute the shares is honest and can compute the function described in Equation (1).

The algorithms in the Byzantine setting are the same as the fail-stop setting, except for some additional steps. In a Byzantine setting, the shares are signed by the dealer. As explained in [20, 21], exploiting the MAC signature, we can prevent the players from sending a false share.

2.6 Revisiting the Protocol involving Embedded XOR with Complete Fairness [20]

In this section, we first describe the solution of the embedded XOR problem with complete fairness proposed by [20]. Let us denote two players by P_1 and P_2. Player P_1 is given an ordered list $\{x_1, x_2, x_3\}$ and P_2 is given an ordered list $\{y_1, y_2\}$. P_1 randomly chooses the input from the ordered list and sends it to the dealer. P_2 also randomly chooses the input from his list and delivers it to the dealer. The dealer calculates the function. For convenience, we here recall the table for f given in [20].

	y_1	y_2
x_1	0	1
x_2	1	0
x_3	1	1

The function can be described as

$$f_w(x_i, y_j) = \begin{cases} 1 & \text{if } i \neq j; \\ 0 & \text{if } i = j. \end{cases} \tag{2}$$

where, x_i and y_j denote the inputs from P_1 and P_2 respectively and $w \in \{1, 2\}$. From now, we denote x_i by x and y_j by y. The protocol proceeds in a series of N iterations, where $N = \omega(\log \lambda)$, λ is the security parameter. The dealer chooses the revelation round r according to a geometric distribution with parameter γ. The dealer then creates two sequences $\{a_l\}$ and $\{b_l\}$, $l = 1, 2, \ldots, N$, as follows.

$$\begin{aligned} &\text{For } l \geq r, \;\; a_l = f_1(x, y) = b_l = f_2(x, y). \\ &\text{For } l < r, \; a_l = f_1(x, \hat{y}), \quad\;\; b_l = f_2(\hat{x}, y), \end{aligned}$$

where $\hat{x}$ (or $\hat{y}$) is a random value of x (or y) chosen by the dealer.

Next, the dealer splits the secret a_l into the shares a_l^1 and a_l^2, and the secret b_l into the shares b_l^1 and b_l^2, so that $a_l = a_l^1 \oplus a_l^2$ and $b_l = b_l^1 \oplus b_l^2$, and gives the shares $\{(a_l^1, b_l^1)\}$ to P_1 and the shares $\{(a_l^2, b_l^2)\}$ to P_2. In each round l, P_2 sends a_l^2 to P_1, who, in turn sends b_l^1 to P_2. P_1 (res. P_2) learns the output value $f_1(x, y)$ (res. $f_2(x, y)$) in iteration r. Here we assume that the dealer who will distribute the shares is honest and can compute the function described in Equation (2).

2.7 Quantum Bit Commitment

For the quantum bit commitment, it has been assumed that there exists a secure quantum one-way permutation. Let $\Phi = \{\phi_n : \{0,1\}^n \to \{0,1\}^n | n > 0\}$ be a family of one-way permutations, where n is the security parameter. Before running the protocol, the parties choose and agree on a $\phi_n \in \Phi$.

In the bit commitment protocol, there are two phases: commit and open. Suppose there are two parties, P and Q. In the commit phase, P chooses a value $w \in \{0,1\}$ to commit, and selects a random $x \in \{0,1\}^n$. Then constructs a quantum state $\rho = |\phi_n(x)\rangle_{\theta(w)^n}$, where $\theta(w)^n$ is the basis used to polarize $\phi_n(x)$. Then P sends it to Q. Q receives ρ and stores it until the open phase. In the open phase, P announces the pair (w, x) for verification. Q first measures the quantum state ρ in $\theta(w)^n$ basis and checks whether it matches with $\phi_n(x)$ or not.

More detailed analysis of the protocol and its security has been described in the paper [17]. But as one can understand that its perfect concealing property comes from the indistinguishability of non-orthogonal states (a law of quantum physics), and the computationally binding property coming from the one-wayness of ϕ_n.

Recently, there have been many other papers with different approaches for secure quantum bit commitment [15, 26, 27] and one can consider those schemes as building blocks in their protocols also.

2.8 Quantum Oblivious Transfer

As we all know, with the help of secure bit-commitment, we can design a secure OT protocol. Here we are going to present a secure OT protocol which was first presented by Claude Crépeau in [14] and then reused in [15].In this protocol, we need a quantum secure bit commitment scheme. As we have already shown that the assumption of the existence of one-way permutation implies the existence of a secure bit-commitment scheme, we can say that the existence of one-way permutation also implies the existence of a secure OT protocol. The protocol is as follows:

Suppose P has two message bits m_0 and m_1, and Q wants to get the message m_b, where $b \in \{0,1\}$. The protocol is as follows:

1. At first, P chooses $2n$-many random bits $s_i \in \{0,1\}$ and $2n$-many random bits $\beta_i \in \{0,1\}$, where $1 \leq i \leq 2n$.
2. If $\beta_i = 0$, P encodes s_i in $\{|0\rangle, |1\rangle\}$ basis. Otherwise, P encodes s_i in $\{|+\rangle, |-\rangle\}$ basis. Let the encoded quantum states be $\rho_i = |s_i\rangle_{\beta_i}, \forall i \in [2n]$.
3. Then P sends all the ρ_is to Q.
4. Q first chooses $2n$-many random bits $\hat{\beta}_i \in \{0,1\}$, where $1 \leq i \leq 2n$.
5. Q measures the state ρ_is in $\{|0\rangle, |1\rangle\}$ basis if $\hat{\beta}_i = 0$. Otherwise Q measures the state ρ_is in $\{|+\rangle, |-\rangle\}$ basis. Let us assume the measured bit at Q's end is r_i, where $r_i \in \{0,1\}$ $\forall i \in [2n]$.
6. Now Q uses the previously described quantum Bit-Commitment protocol (BC) to commit to the values $(\hat{\beta}_i, r_i)$ and $(\hat{\beta}_{n+i}, r_{n+i})$, for $1 \leq i \leq n$, to P. (The commited values are actually strings of qubits as described in the above quantum bit-commitment protocol, but for the ease of understanding here we call them commited values loosely.)
7. Then P chooses n-many random bits $f_i \in \{0,1\}$ and sends them to Q.
8. After that, Q opens the committed values $(\hat{\beta}_{nf_i+i}, r_{nf_i+i})$, for $1 \leq i \leq n$, to P.
9. Then P checks whether $r_{nf_i+i} = s_{nf_i+i}$ holds or not, whenever $\beta_{nf_i+i} = \hat{\beta}_{nf_i+i}$ holds. If it holds, then they proceed further. Otherwise, P terminates the protocol.
10. After the successful verification, whenever $f_i = 0$, P sets $\beta_i \leftarrow \beta_{n+i}$ and Q sets $\hat{\beta}_i \leftarrow \hat{\beta}_{n+i}$.
11. Then P announces the values of $\beta_i, \forall i \in [n]$.
12. After that, Q chooses two sets $N_0, N_1 \subset \{1, 2, ..., n\}$, such that $|N_0| = |N_1| = \frac{n}{3}$, $N_0 \cap N_1 = \emptyset$, and $\beta_i = \hat{\beta}_i, \forall i \in N_b$, and sends N_0, N_1 to P.
13. P then sets the following values:

$$\hat{m}_0 = m_0 \bigoplus_{i \in N_0} s_i, \quad \hat{m}_1 = m_1 \bigoplus_{i \in N_1} s_i$$

After that, P sends the values $\hat{m}_0$ and $\hat{m}_1$ to Q.

14. Upon receiving the values, Q computes

$$m_b = \hat{m}_b \bigoplus_{i \in N_b} r_i.$$

Now, the security of P against a dishonest Q comes from the computationally binding property of the BC protocol, and the security of Q against a dishonest P comes from the perfectly concealing property of the BC protocol. As we already know, though the concealing property of BC comes naturally from the law of physics, for the binding property, we need the assumption of the existence of a quantum one-way permutation. This implies that to achieve a secure OT protocol, we need the assumption of the existence of a quantum one-way permutation.

3 Quantum Solution of Millionaires' Problem with Complete Fairness in Hybrid Model

In this section, we propose a quantum solution to the millionaire's problem in the hybrid model. It is the quantum analogue of the protocol of [20, 21] in the classical domain. However, their security proof is based on some computational hardness in the classical domain. Whereas we exploit the property of entanglement to provide security of the protocol in the quantum domain. After that, in Section 4 we describe how we can achieve complete fairness in the real world from our hybrid model.

Here, we exploit four Bell state basis [37]. The maximally entangled two particle state is $|g_0\rangle = \frac{1}{\sqrt{2}}\Big[\,|0\rangle_1 |0\rangle_2 + |1\rangle_1 |1\rangle_2\,\Big]$. This state is called Einstein, Podolsky, Rosen Pair, in short EPR pair or Bell state. There are four independent Bell states. They are

$$|g_0\rangle = \frac{1}{\sqrt{2}}\Big[\,|0\rangle_1 |0\rangle_2 + |1\rangle_1 |1\rangle_2\,\Big], |g_1\rangle = \frac{1}{\sqrt{2}}\Big[\,|0\rangle_1 |0\rangle_2 - |1\rangle_1 |1\rangle_2\,\Big],$$
$$|g_2\rangle = \frac{1}{\sqrt{2}}\Big[\,|0\rangle_1 |1\rangle_2 + |1\rangle_1 |0\rangle_2\,\Big], |g_3\rangle = \frac{1}{\sqrt{2}}\Big[\,|0\rangle_1 |1\rangle_2 - |1\rangle_1 |0\rangle_2\,\Big].$$

Here, subscript 1 stands for P_1's qubit and subscript 2 stands for P_2's qubit. We need any three of these orthogonal states. In this work, without loss of generality, we consider $|g_0\rangle$, $|g_1\rangle$ and $|g_2\rangle$.

Like classical case, the secret of P_1 is i and the secret of P_2 is j, $1 \leq i \leq N$, $1 \leq j \leq N$ where N is any integer. They want to know whether $i > j$ or $i \leq j$. The dealer supplies them two ordered lists, $X = \{x_1, x_2, \ldots, x_N\}$ to P_1 and $Y = \{y_1, y_2, \ldots, y_N\}$ to P_2. P_1 chooses x_i and P_2 chooses y_j from their respective lists and sends those to the dealer. The dealer will compute the function and will distribute the shares (here, qubits) in such a way that P_1 will get the value of the function, i.e, $f_1(x_i, y_j)$ in iteration i and P_2 will get the value of the function, i.e, $f_2(x_i, y_j)$ in iteration j. The protocol proceeds in a series of N iterations. In a round $l \neq i$ P_1 outputs $\perp$ and in a round $l \neq j$ P_2 outputs $\perp$.

3.1 Protocol Description

This protocol has two phases: the Share Generation phase and the Reconstruction phase.

In the Share Generation phase, the dealer gets the inputs from both the parties, and then checks whether the inputs are valid or not. If both the inputs are valid then it computes $f_1(x_i, y_j) = f_2(x_i, y_j) = b$, where $b \in \{0, 1\}$. Then for each round $l (\leq N)$, depending on the value of b and l it selects two entangled states from the Bell states $|g_0\rangle$, $|g_1\rangle$, $|g_2\rangle$, and sends the 1st qubit of both the chosen entangled states to P_1 and remaining 2nd qubit of both the chosen entangled states to P_2. For extra protection against any cheating, like applying any gate operation on the shared qubit before reconstruction by the dishonest party to cheat the honest party, we have used an extra encoding. For this encoding we have used Algebraic Manipulation Detection (AMD) code [13]. For each entangled states $|g_0\rangle$, $|g_1\rangle$, $|g_2\rangle$ the trusted third party chooses three element X_0, X_1, X_2 from $\mathbb{F}_q^d$. In each round, for each shared Bell state $|g_s\rangle$ it encodes X_s using an uniformly chosen random number r_s, and generates a tag t_s. It shares the r_s and t_s appropriately to check any tampering on the shared qubits. Detailed description of this phase is given in Algorithm 1.

Algorithm 1: QShareGen

Input: Input x_i from P_1 and input y_j from P_2, where indices $i, j \in \{1, \ldots, N\}$, and domain $\mathcal{D}$ with $|\mathcal{D}| = N$, a field $\mathbb{F}_q$.

Output: Share lists $list_1, list_2$ of $2N$ qubits and $2N$ classical bits each to P_1, P_2 (or $\perp$ if x_i or y_j does not belong to $\mathcal{D}$).

if $x_i \notin \mathcal{D}$ ***or*** $y_j \notin \mathcal{D}$ **then return** $\perp$ to both the parties

$b \leftarrow f_1(x_i, y_j) = f_2(x_i, y_j) \in \{0, 1\}$

// Notation: $\delta_{a,b} = 1$ iff $a = b$; $|g_b\rangle = \frac{1}{\sqrt{2}}(|00\rangle + (-1)^b |11\rangle)$; $|g_2\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$;

Choose three elements X_0, X_1, X_2 from $\mathbb{F}_q^d$ and make them publicly available

for $l \leftarrow 1$ **to** N **do**

  for $m \in \{1, 2\}$ **do**

    // Set $|g_b\rangle$ at $(m = 1, l = i)$ and $(m = 2, l = j)$; $|g_2\rangle$ elsewhere;

    $|\psi_{m,l}\rangle \leftarrow \left(\delta_{m,1}\delta_{l,i} + \delta_{m,2}\delta_{l,j}\right)|g_b\rangle + \left(1 - \delta_{m,1}\delta_{l,i} - \delta_{m,2}\delta_{l,j}\right)|g_2\rangle$

    $r_{m,l} \xleftarrow{\$} \mathbb{F}_q$

    // Depending on $|\psi_{m,l}\rangle$ (whether it is $|g_s\rangle$, for $s \in \{0, 1, 2\}$), encode X_s with $r_{m,l}$ using AMD code and get a tag $t_{m,l}$;

    for $s \leftarrow 0$ **to** 2 **do**

      if $|g_s\rangle$ *has been assigned to* $|\psi_{m,l}\rangle$ **then**

        $t_{m,l} \leftarrow {r_{m,l}}^{d+2} + \sum_{ind=1}^{d} X_{s,ind} \cdot {r_{m,l}}^{ind}$

$list_1[l] \leftarrow$ (1st half of the prepared entangled state $|\psi_{1,l}\rangle$, 1st half of the prepared entangled state $|\psi_{2,l}\rangle$, $r_{2,l}$, $t_{1,l}$)

$list_2[l] \leftarrow$ (2nd half of the prepared entangled state $|\psi_{1,l}\rangle$, 2nd half of the prepared entangled state $|\psi_{2,l}\rangle$, $r_{1,l}$, $t_{2,l}$)

return $list_1$ to P_1, and $list_2$ to P_2

In the Reconstruction phase, in each round, at first P_2 sends the first qubit of its list for that round to P_1, and after that P_1 sends the second qubit of its list for that round to P_2. Upon receiving the shares, they measure the quantum states in the Bell basis. P_1 gets the output b in i-th round and P_2 gets the output in j-th round. For the detailed description of the reconstruction phase, one may refer to Algorithm 2.

3.2 Security Analysis

A Byzantine player can behave arbitrarily. He can manipulate the shares (here, qubits) which he has obtained from the dealer, or may abort early. In this subsection, we will show how entanglement provides security against such manipulation. The aborting case will be discussed next.

3.2.1 Security Against Forgery

Without loss of generality, let us assume that P_1 tries to manipulate the qubits obtained from the dealer in the motivation to convey the wrong message to P_2. Here, manipulation means sending an arbitrary qubit or swapping the qubits of his list or applying any gate operation on the qubit. This forgery is detected with significant probability, i.e. the probability of successful cheating is negligible. Like classical MAC signature, in quantum domain, entanglement along with AMD code [13] provides security against such forgery.

Theorem 1. *The failure probability of P_2 to detect cheating by P_1 who is corrupted by a Byzantine adversary in an arbitrary round $l \leq j$ is $\frac{(d+1)(j+1)}{4qN}$.*

Algorithm 2: $\Pi_{\mathsf{Fair}}^{\mathsf{QMP}}$

Input: Lists $list_1, list_2$ of $2N$ qubits and $2N$ classical bits each, private indices $i, j \in \{1, ..., N\}$, $X_0, X_1, X_2 \in \mathbb{F}_q^d$.

Output: P_1 outputs evaluated $out_1 \in \{0, 1\}$; P_2 outputs evaluated $out_2 \in \{0, 1\}$.

for $l \leftarrow 1$ **to** N **do**

  `// Symmetric exchange:` $(a = \texttt{sender}, b = \texttt{receiver}, t = \texttt{target}, \textit{def} = \texttt{default})$

  for $(a, b, t, def) \in \{(2, 1, i, 1), (1, 2, j, 0)\}$ **do**

    if P_a *aborts or fails to send qubit* **then**

      return P_b outputs *def* if $l \leq t$, else outputs out_b

    P_a transmits $list_a[l, b]$ and $list_a[l, 3]$ to P_b

    P_b measures $|\psi\rangle$ (constructed by $list_1[l, b]$ and $list_2[l, b]$) in Bell basis $\{|g_0\rangle, |g_1\rangle, |g_2\rangle, |g_3\rangle\}$

    if *Measured state is* $|g_3\rangle$ **then**

      abort and report forgery by P_a

    `// For measured state` $|g_c\rangle$`, check whether it is genuine or tampered through AMD code verification`

    $|g_c\rangle$ is the measured state for $c \in \{0, 1, 2\}$

    $r' \leftarrow$ 3rd part of $list_a[l]$

    $t' \leftarrow (r')^{d+2} + \Sigma_{ind=1}^{d} X_{c,ind} \cdot (r')^{ind}$

    $t'' \leftarrow$ 4th part of $list_b[l]$

    if $t' \neq t''$ **then**

      abort and report forgery by P_a

    else if $l = t$ ***and*** *measured state is* $|g_c\rangle$ *with* $c \in \{0, 1\}$ **then**

      $out_b \leftarrow c$

    else if $l \neq t$ ***and*** *measured state is* $|g_2\rangle$ **then**

      continue

    else

      abort and report forgery by P_a

return P_1 outputs out_1, P_2 outputs out_2

Proof. According to the protocol, in round $l \neq j$, if no cheating occurs, then P_2 will get $|g_2\rangle = \frac{1}{\sqrt{2}}\Big[|0\rangle_1 |1\rangle_2 + |1\rangle_1 |0\rangle_2 \Big]$. In terms of density matrix it can be written as

$$\rho = \frac{1}{2}\Big(|0\rangle_1 |1\rangle_2 + |1\rangle_1 |0\rangle_2 \Big)\Big(\langle 0|_1 \langle 1|_2 + \langle 1|_1 \langle 0|_2 \Big).$$

Now, let us assume that P_1 sends an arbitrary qubit which is $|\phi\rangle = [\alpha |0\rangle_3 + \beta |1\rangle_3]$, instead of the correct one. In terms of density matrix, the arbitrary state can be written as

$$\rho_3 = |\phi\rangle \langle\phi| = \Big[|\alpha|^2 |0\rangle_3 \langle 0| + \alpha^* \beta |1\rangle_3 \langle 0| + \alpha\beta^* |0\rangle_3 \langle 1| + |\beta|^2 |1\rangle_3 \langle 1| \Big].$$

Thus, the state at the end of P_2 would be

$$\begin{aligned}
\rho_2 &= [tr_{P_1}(\rho)](\rho_3) \\
&= \frac{1}{2}\Big[|1\rangle_2 \langle 1| \Big(|\alpha|^2 |0\rangle_3 \langle 0| + \alpha^* \beta |1\rangle_3 \langle 0| + \alpha\beta^* |0\rangle_3 \langle 1| + |\beta|^2 |1\rangle_3 \langle 1| \Big) \\
&\quad + |0\rangle_2 \langle 0| \Big(|\alpha|^2 |0\rangle_3 \langle 0| + \alpha^* \beta |1\rangle_3 \langle 0| + \alpha\beta^* |0\rangle_3 \langle 1| + |\beta|^2 |1\rangle_3 \langle 1| \Big) \Big].
\end{aligned}$$

In this case, when P_2 will measure qubit 2 and qubit 3 in Bell basis, after measurement, P_2 will get $|g_0\rangle$ or $|g_1\rangle$ or $|g_2\rangle$ or $|g_3\rangle$ with probability $\frac{1}{4}$ instead of $|g_2\rangle$ only. The detailed calculations are given here.

Let us assume that P_2 obtains $|g_0\rangle$ after measurement. Thus, the probability that P_2 obtains $|g_0\rangle$ is given by

$$\begin{aligned}
&|g_0\rangle_{23}\langle g_0|(\rho_2) = \langle g_0|\rho_2|g_0\rangle_{23} \\
&= \frac{1}{4}\Big[\Big(\langle 00|_{23} + \langle 11|_{23}\Big)\Big[|1\rangle_2\langle 1|\Big(|\alpha|^2|0\rangle_3\langle 0| + \alpha^*\beta|1\rangle_3\langle 0| + \alpha\beta^*|0\rangle_3\langle 1| + \\
&|\beta|^2|1\rangle_3\langle 1|\Big) + |0\rangle_2\langle 0|\Big(|\alpha|^2|0\rangle_3\langle 0| + \alpha^*\beta|1\rangle_3\langle 0| + \alpha\beta^*|0\rangle_3\langle 1| + \\
&|\beta|^2|1\rangle_3\langle 1|\Big)\Big]\Big(|00\rangle_{23} + |11\rangle_{23}\Big)\Big] \\
&= \frac{1}{4}\Big[|\alpha|^2 + |\beta|^2\Big] = \frac{1}{4}.
\end{aligned}$$

If $l \neq j$, according to our protocol, P_2 should get $|g_2\rangle$ only. But as P_1 sends an arbitrary qubit to P_2, when measured, P_2 gets any one of the four Bell states with probability $\frac{1}{4}$. Thus, if $l \neq j$ and P_2 gets $|g_0\rangle$ or $|g_1\rangle$ or $|g_3\rangle$, he immediately concludes that P_1 is cheating. The success probability of detecting such cheating for a round $l \neq j$ is $\frac{3}{4}$ just by the property of entanglement. But we have also used AMD code for extra verification. In [13], it is shown that the probability of undetected cheating is $\frac{d+1}{q}$. So the probability of undetected cheating in spite of having entanglement and the AMD code is $(\frac{1}{4}\cdot\frac{d+1}{q})$.

Similarly, when $l = j$, if P_1 does not cheat, P_2 would get either $|g_0\rangle$ or $|g_1\rangle$ depending on the value of $f_2(x_i, y_j)$. However, if P_1 cheats, when measured, P_2 will get any one Bell state. In case of $|g_0\rangle$ and $|g_1\rangle$, he can not detect the cheating just by the entanglement property because he does not know the value of $f_2(x_i, y_j)$ a priori. However, if he gets $|g_2\rangle$ or $|g_3\rangle$, he immediately detects the cheating with certainty. But as we have used AMD code, for the measured states $|g_0\rangle$ or $|g_1\rangle$, we have extra protection from undetected cheating. Since the probability of undetected cheating in AMD code is $\frac{d+1}{q}$, the actual probability of an undetected cheating in round $l = j$ is $(\frac{1}{2}\cdot\frac{d+1}{q})$. As, P_1 have no idea about the value of j, the average probability of undetected cheating is

$$\begin{aligned}
&(\frac{1}{4}\cdot\frac{d+1}{q})\Pr(l \neq j) + (\frac{1}{2}\cdot\frac{d+1}{q})\Pr(l = j) \\
&= (\frac{1}{4}\cdot\frac{d+1}{q})[\Pr(l < j) + \Pr(l > j)] + (\frac{1}{2}\cdot\frac{d+1}{q})\Pr(l = j).
\end{aligned}$$

We do not bother about $\Pr(l > j)$ because, P_2 should have no incentive to detect the cheating when $l > j$, as he has already got his output value in round j. Thus the total success probability of P_2 to detect such cheating conditioning on the event that $l \leq j$ is

$$\begin{aligned}
&(\frac{1}{4}\cdot\frac{d+1}{q})\Pr(l < j) + (\frac{1}{2}\cdot\frac{d+1}{q})\Pr(l = j) \\
&= (\frac{1}{4}\cdot\frac{d+1}{q}).\frac{j-1}{N} + (\frac{1}{2}\cdot\frac{d+1}{q}).\frac{1}{N} = \frac{(d+1)(j+1)}{4qN}.
\end{aligned}$$

□

For appropriate choices of d and q we can achieve desired negligible failing rate. The same conclusion can be drawn when we assume P_2 is corrupted. In this case, we modify the theorem in the following way.

Theorem 2. *The failure probability of P_1 to detect cheating by P_2 who is corrupted by a Byzantine adversary in an arbitrary round $l \leq i$ is $\frac{(d+1)(i+1)}{4qN}$.*

3.2.2 Fairness Against Early Abort

As P_1 is always computing its output first followed by P_2, the aborting of P_1 plays an important role to achieve the fairness of the protocol. The early abort of P_2 will terminate the protocol up to that round in

which P_2 aborts. In that case, either both get the output or none gets the output. Thus, early abort of P_2 does not affect the fairness condition. We now concentrate on the early abort of P_1.

Theorem 3. *The protocol* $\mathbf{\Pi_{Fair}^{QMP}}$ *achieves fairness.*

Proof. Let us assume that P_1 aborts in round l. There are two cases: $i \leq j$ and $i > j$. We analyze each case one by one.

Case 1: $i \leq j$.
Subcase 1(a): $l < i$. In this case, P_1 outputs $\perp$ and P_2 outputs 0. In *ideal world model*, the trusted party sends $f_1(x_i, y_j)$ to P_1 in iteration i and $f_2(x_i, y_j)$ to P_2 in iteration j. In all other rounds trusted party sends $\perp$ to both P_1 and P_2. If a party P_1 (res. P_2) aborts the protocol in an arbitrary round l after getting the output, the trusted party sends the honest party P_2 (res. P_1) the value of $f_2(x_l, y_j)$ (res. $f_1(x_i, y_l)$). Thus when P_1 aborts in round $l < i$, P_1 outputs $\perp$ whereas P_2 outputs $f_2(x_l, y_j)$. As $i \leq j$ and $l < i$, then $l < j$. So $f_2(x_l, y_j) = 0$ (refer to Equation 1). Hence,

$$\begin{aligned}&\Pr\left[\Big(VIEW_{ideal}(x,y), OUT_{ideal}(x,y)\Big) = (\perp, 0)\Big| l < i \bigwedge i \leq j\right]\\ &= \Pr\left[\Big(VIEW_{hybrid}(x,y), OUT_{hybrid}(x,y)\Big) = (\perp, 0)\Big| l < i \bigwedge i \leq j\right].\end{aligned}$$

Subcase 1(b): $l = i$. In this case, P_1 obtains the correct output i.e. 0 and P_2 outputs 0. In *ideal model*, when P_1 aborts in round $l = i$, trusted party has already sent 0 to P_1 and $f_2(x_i, y_j)$ to P_2. As $i \leq j$, $f_2(x_i, y_j) = 0$ (Equation 1). Hence,

$$\begin{aligned}&\Pr\left[\Big(VIEW_{ideal}(x,y), OUT_{ideal}(x,y)\Big) = (0, 0)\Big| l = i \bigwedge i \leq j\right]\\ &= \Pr\left[\Big(VIEW_{hybrid}(x,y), OUT_{hybrid}(x,y)\Big) = (0, 0)\Big| l = i \bigwedge i \leq j\right].\end{aligned}$$

Subcase 1(c): $l > i$. Here two cases can arise. i) $i < l \leq j$, in this case, P_1 obtains correct output and P_2 outputs 0. ii) $i = j < l$, in this case, both P_1 and P_2 have already obtained 0. In *ideal model*, if P_1 aborts in round $l > i$, P_1 has already got its output value whereas trusted party sends $f_2(x_l, y_j)$ to P_2. When $i < l \leq j$, then $f_2(x_l, y_j) = 0$ (Equation 1) whereas for $i = j$, P_2 has already got the correct output i.e 0. Hence,

$$\begin{aligned}&\Pr\left[\Big(VIEW_{ideal}(x,y), OUT_{ideal}(x,y)\Big) = (0, 0)\Big| l > i \bigwedge i \leq j\right]\\ &= \Pr\left[\Big(VIEW_{hybrid}(x,y), OUT_{hybrid}(x,y)\Big) = (0, 0)\Big| l > i \bigwedge i \leq j\right].\end{aligned}$$

Case 2: $i > j$.
Subcase 2(a): $l \leq j$. In this case, P_1 outputs $\perp$ and P_2 outputs 0. In *ideal model*, if P_1 aborts in round $l \leq j$, P_1 outputs $\perp$ and trusted party sends $f_2(x_l, y_j)$ to P_2. As $l \leq j$, $f_2(x_l, y_j) = 0$ (Equation 1). Hence,

$$\begin{aligned}&\Pr\left[\Big(VIEW_{ideal}(x,y), OUT_{ideal}(x,y)\Big) = (\perp, 0)\Big| l \leq j \bigwedge i > j\right]\\ &= \Pr\left[\Big(VIEW_{hybrid}(x,y), OUT_{hybrid}(x,y)\Big) = (\perp, 0)\Big| l \leq j \bigwedge i > j\right].\end{aligned}$$

Subcase 2(b): $j < l < i$. In this case, P_1 obtains $\perp$ and P_2 gets the correct output i.e. 1. In *ideal model*, if P_1 aborts in round $j < l < i$, P_1 is given $\perp$ whereas the trusted party sends $f_2(x_l, y_j)$ to P_2. As $j < l$, then $f_2(x_l, y_j) = 1$ (Equation 1). Hence,

$$\begin{aligned}&\Pr\left[\Big(VIEW_{ideal}(x,y), OUT_{ideal}(x,y)\Big) = (\perp, 1)\Big| j < l < i \bigwedge i > j\right]\\ &= \Pr\left[\Big(VIEW_{hybrid}(x,y), OUT_{hybrid}(x,y)\Big) = (\perp, 1)\Big| j < l < i \bigwedge i > j\right].\end{aligned}$$

Subcase 2(c): $j < l = i$. In this case, P_1 and P_2 both obtain the correct output i.e. 1. In *ideal model*, if P_1 aborts in round $j < l = i$, P_1 is given 1 whereas the trusted party sends $f_2(x_l, y_j)$ to P_2. As $j < l = i$, then

$f_2(x_l, y_j) = f_2(x_i, y_j) = 1$ (Equation 1). Hence,

$$\Pr\left[\Big(VIEW_{ideal}(x,y), OUT_{ideal}(x,y)\Big) = (1,1)\Big| j < l = i \bigwedge i > j\right]$$
$$= \Pr\left[\Big(VIEW_{hybrid}(x,y), OUT_{hybrid}(x,y)\Big) = (1,1)\Big| j < l = i \bigwedge i > j\right].$$

When $i < l < N$, P_1 has no incentive to abort as in this case both P_1 and P_2 have already obtain their respective outputs.

Hence, from the above analysis, we can conclude that in the hybrid model, the adversary does no more harm than the ideal scenario. Thus our protocol achieve fairness. □

As discussed in the paper [20], if we want to compute a functionality $\mathcal{F}$ in the real world, then at first we can design a secure protocol to compute it in a $\mathcal{G}$-Hybrid model (say, π, which calls $\mathcal{G}$ to compute $\mathcal{F}$) and then can replace $\mathcal{G}$ by a real-world protocol ρ. We name the final real-world protocol as π^ρ. Also in that paper, we have seen that if the protocol ρ can securely implement $\mathcal{G}$ and if the protocol π achieves complete fairness to compute $\mathcal{F}$ in $\mathcal{G}$-hybrid model, then the protocol π^ρ computes $\mathcal{F}$ with complete fairness in real world.

In this section, we have already shown that our hybrid model achieves complete fairness. Using the above fact, we can also construct a real-world protocol from this hybrid model. We have discussed how we can do so in detail in Section 4.

4 Quantum 2PC Protocol for Millionaires' Problem in Real-World Model

In this section, we present the step-by-step process for obtaining a real-world, secure quantum 2PC protocol without a trusted third party from our hybrid model.

4.1 The Main Duties of the Trusted Third Party

In Gordon's Paper and also in the hybrid model of our paper, the third party is doing the following things:

1. It is actually computing the function $f(x_i, y_j)$ with the secret inputs x_i corresponding to i and y_j corresponding to j, and then generating the shares of the computed value of $f(x_i, y_j)$ for several rounds in a special manner. (Here $f(x_i, y_j) = 1$ iff $i > j$, otherwise $f(x_i, y_j) = 0$.)
2. It is setting two functions $f_1(x_l, y_l) = a_l$, where only $a_i = f(x_i, y_j)$ and for any $l \neq i$, $a_l = \perp$. Similarly $f_2(x_l, y_l) = b_l$ such that only $b_j = f(x_i, y_j)$ and for any $l \neq j$, $b_l = \perp$. Then generates shares of a_l, b_l for each round l.
3. It is assuring that P_1 is getting the actual share of the output only in round i and nowhere else. It is ensuring this by sharing the shares of $\perp$ in all the round l, where $l \neq i$.
4. Similarly it is assuring that P_2 is getting the actual share of the output only in round j and nowhere else. It is ensuring this by sharing the shares of $\perp$ in all the round l, where $l \neq j$.

4.2 Idea of Removing Third Party Using Ran Canetti's Composition Theorem [8]

We have already shown that our proposed protocols achieve complete fairness in the hybrid model. Now we want to show that it is possible to replace our trusted third party in the hybrid model if we can get a secure 2PC protocol to do the third party's work properly. We want to show that if ρ can securely compute f and if the protocol π achieves complete fairness in f-hybrid model (a hybrid model, where the trusted third party has been used to compute the function f securely), then π^ρ (it is the real-world protocol, where the trusted third party of the protocol π has been replaced by ρ) can compute f with complete fairness in the real-world, using Ran Canetti's idea [8]. To show this we will prove the following theorem.

Theorem 4. *Suppose f is a 2-party function, π is a 2-party protocol in f-hybrid model to achieve complete fairness, ρ is a secure 2-party protocol to compute f in real world, and π^ρ is the composed protocol in the real-world to compute f. Then for any nonadaptive real-life adversary $\mathcal{A}$, there exists a nonadaptive adversary $\mathcal{A}_\pi$ in the f-hybrid model, whose running time is polynomial in the running time of $\mathcal{A}$, and such that*

$$\{HYBRID_{\pi,\mathcal{A}_\pi(z),\lambda}(x,y)\}_{(x,y)\in X\times Y, z\in\{0,1\}^*} \equiv \{REAL_{\pi^\rho,\mathcal{A}(z),\lambda}(x,y)\}_{(x,y)\in X\times Y, z\in\{0,1\}^*}.$$

Proof. To prove the theorem, we first construct $\mathcal{A}_\rho$ using $\mathcal{A}$, and then, using this, we construct $\mathcal{A}_\pi$, which satisfies the desired condition.

Construction of $\mathcal{A}_\rho$:
Here, parties P_1 and P_2 run the protocol ρ. One of them is corrupted. Adversary $\mathcal{A}_\rho$ starts with a set C containing the corrupted party, input of the party in C, and the auxiliary input z. The adversary $\mathcal{A}_\rho$ works as follows:

1. Let l be the round where protocol π^ρ evokes the protocol ρ for the first time. $\mathcal{A}_\rho$ verifies whether auxiliary input z is a valid internal state of $\mathcal{A}$ at round $(l-1)$ or not. If not, then it halts with no output. Otherwise,
   (a) Run $\mathcal{A}$ from the internal set z. Let P_1', P_2' be the parties (imaginary parties who are running π_ρ) which interact with $\mathcal{A}$.
   (b) It also considers a C', which contained the corrupted party for $\mathcal{A}$, where $P_j' \in C'$ iff $P_j \in C$. And the input of the corrupted party P_j' is the same as the input of P_j.
   (c) Whenever the uncorrupted party P_i sends a message m to the corrupted party P_j, $\mathcal{A}_\rho$ simulates $\mathcal{A}$ as if P_i' has sent the message m to P_j'.
   (d) Whenever $\mathcal{A}$ instruct the corrupted party P_j' to sent a message m' to the uncorrupted party P_i', $\mathcal{A}_\rho$ instruct P_j to send the same message m' to P_i.
2. Once $\mathcal{A}$ halts, $\mathcal{A}_\rho$ outputs the current internal state of $\mathcal{A}$ and halts.

Now from the security of ρ, we can say that there exists an adversary $\mathcal{S}_\rho$ such that

$$\{REAL_{\rho,\mathcal{A}_\rho(z),\lambda}(x,y)\}_{(x,y)\in X\times Y, z\in\{0,1\}^*} \equiv \{IDEAL_{f,\mathcal{S}_\rho(z),\lambda}(x,y)\}_{(x,y)\in X\times Y, z\in\{0,1\}^*}. \tag{3}$$

Now we use this $\mathcal{S}_\rho$ along with $\mathcal{A}$ to build an adversary $\mathcal{A}_\pi$ for the protocol π in f-hybrid model.

Construction of $\mathcal{A}_\pi$:
Here, the adversary $\mathcal{A}_\pi$ interacts with the parties P_1 and P_2, who are running the protocol π and have access to a trusted third-party to compute f. $\mathcal{A}_\pi$ has access to the set C, contains the corrupted party, the input of the corrupted party, and the auxiliary input Z. The adversary $\mathcal{A}_\pi$ works as follows:

1. Let l be the round where protocol π^ρ evokes the protocol ρ for the first time.
2. It runs $\mathcal{A}$ on C, input of the corrupted party, and z up to round $(l-1)$. (As we can recall that up to this round π^ρ and π work exactly in the same manner.)
3. In the l-th round, $\mathcal{A}$ expects to start interacting with ρ and get an output from ρ. But in protocol π, there is no ρ, instead of it here the protocol π has access to a trusted third party to compute f. So, in order to continue an uninterrupted run of $\mathcal{A}$, $\mathcal{A}_\pi$ invokes $\mathcal{S}_\rho$ as follows:
   (a) $\mathcal{S}_\rho$ is given the the set C, and the input of the party in C. It gets the current internal state of $\mathcal{A}$ as its auxiliary input z.
   (b) As we know that $\mathcal{S}_\rho$ is an adversary in the ideal model of computing f, then $\mathcal{S}_\rho$ can output a string such that the following holds:

$$(VIEW_f(x,y), OUT_f(x,y)) \equiv (VIEW_{\mathcal{S}_\rho}(x,y), OUT_{\mathcal{S}_\rho}(x,y)). \tag{4}$$

4. Once it gets the output from $\mathcal{S}_\rho$, it uses this as the internal state of $\mathcal{A}$, as if it is an output from ρ.
5. It then runs $\mathcal{A}$ on that internal state until the protocol π ends.
6. Once the protocol π is done, it outputs the output of $\mathcal{A}$ and halts.

Now here until $(l-1)$-th round the internal states of $\mathcal{A}$ and $\mathcal{A}_\pi$ are same. From equation (2), we get

$$REAL_{\rho,\mathcal{A}_\rho(z),\lambda}(x,y)\}_{(x,y)\in X\times Y, z\in\{0,1\}^*} \equiv \{IDEAL_{f,\mathcal{S}_\rho(z),\lambda}(x,y)\}_{(x,y)\in X\times Y, z\in\{0,1\}^*}$$

i.e.,

$$(VIEW_\rho(x,y), OUT_\rho(x,y)) \equiv (VIEW_{\mathcal{S}_\rho}(x,y), OUT_{\mathcal{S}_\rho}(x,y)). \tag{5}$$

So, from equations (3) and (4), we can conclude that

$$(VIEW_\rho(x,y), OUT_\rho(x,y)) \equiv (VIEW_f(x,y), OUT_f(x,y)) \tag{6}$$

Again, after getting the output from $\mathcal{S}_\rho$, $\mathcal{A}_\pi$ works exactly as how $\mathcal{A}$ works after getting output from ρ. Let k be the round where $\mathcal{A}$ gets the output from ρ and set the output as its internal state for $(k+1)$.
As we have already shown in equation (3), the output of $\mathcal{S}_\rho$ and the trusted third-party in π are equivalent, and as we have also shown in equation (5), the output of the trusted third party in π and the output of ρ in π^ρ are equivalent, we can conclude that the internal state of $\mathcal{A}_\pi$ against π is equivalent to the internal state of $\mathcal{A}$ against π^ρ in the $(k+1)$-th round. Since after $(k+1)$-th round $\mathcal{A}$ and $\mathcal{A}_\pi$ works identically, we can finally conclude that

$$\{REAL_{\pi^\rho,\mathcal{A}(z),\lambda}(x,y)\}_{(x,y)\in X\times Y,z\in\{0,1\}^*} \equiv \{HYBRID_{\pi,\mathcal{A}_\pi(z),\lambda}(x,y)\}_{(x,y)\in X\times Y,z\in\{0,1\}^*}.$$

This concludes the theorem. □

So, clearly we can conclude that, if ρ can securely compute f and can handle all the duties of the third party and if the protocol π achieves complete fairness in f-hybrid model, then π^ρ (it is the real-world protocol, where the trusted third party of the protocol π has been replaced by ρ) can compute f with complete fairness in the real-world.

4.3 Existence of such a Quantum Secure 2PC to Handle all the Functionalities of the Trusted Third Party

If we want to use the above theorem to achieve a 2PC protocol in the real world to achieve complete fairness, then we have to find a secure 2PC protocol that can manage all the functionalities of the trusted third party efficiently. Now to find the existence of such a secure 2PC, we first focus on the first functionality of the trusted third party, and then we will discuss the other functionalities later. To construct such a secure 2PC, which compute $f(x,y)$ from secret x,y without sharing information about them, we need some cryptographic tools like quantum bit commitment and quantum oblivious transfer. We have given a detailed description of that real world secure quantum 2PC protocol, which is not fair, in Section 4.3.1.

4.3.1 Quantum 2PC for Millionaires' Problem in Real World

In 2PC, a very well-known problem is the Millionaires' problem. In this problem, there are two parties with their corresponding secret inputs x and y, respectively. They want to compute whether $x > y$ or not without revealing their secret to the other party. Now the question is, why have we chosen this particular problem? In the paper [20], the authors have shown that if we can achieve a solution for the 'greater-than' problem, then we can claim that the protocol can be applied to any function defined over polynomial-size domains that does not contain an embedded XOR. This is the main reason for choosing this particular problem. Now we are going to present our secure quantum 2PC protocol to compute the greater-than problem (which has been used, as a sub-protocol, in our real-world protocol for computing Millionaires' problem with complete Fairness).

Circuit Generation: Suppose x and y are two n-bit integers. Let us assume that $x = x_{n-1}x_{n-2}...x_1x_0$ and $y = y_{n-1}y_{n-2}...y_1y_0$. Suppose $f : \{0,1\}^{2n} \to \{0,1\}$ is a function such that $f(x,y) = 1$ iff $x > y$, otherwise $f(x,y) = 0$.

Then we can write the boolean expression of f as

$$f(x,y) = \bigvee_{i=0}^{n-1} \left((x_i \wedge \neg y_i) \wedge \bigwedge_{j=i+1}^{n-1} (x_j \odot y_j) \right).$$

Since we can write $(a \vee b)$ as $\neg(\neg a \wedge \neg b)$, $(a \odot b)$ as $\neg(a \oplus b)$, and we can compute NOT gate ($\neg$), XOR gate ($\oplus$), and AND gate ($\wedge$) using X-gate, CNOT gate, and Toffoli gate, respectively, we can construct the quantum circuit for f.

Share Generation: In 2PC, Alice and Bob use secret sharing schemes to generate secret shares of their own inputs and send one share of their inputs to the other. Now to compute $f(x,y)$, the party actually runs

the circuit on their own shares such that their corresponding circuit outputs are actually secret shares of $f(x, y)$, and finally reconstructs the actual output from the output shares. So, if we want to successfully compute $f(x, y)$ using 2PC protocols, then we have to use a secret sharing scheme that is gate invariant.

Now, clearly, additive secret sharing is not itself AND-gate invariant, but if we can use OT, then we can use additive secret sharing to handle the computation of f.

Suppose Alice's input is $x = x_0 \oplus x_1$ and Bob's input is $y = y_0 \oplus y_1$. Then Alice sends x_1 to Bob, and Bob sends y_0 to Alice. (Here the shared values are actually $|x_1\rangle$ and $|y_0\rangle$, but for the ease of understanding we have written them as just values throughout the protocol. Actually at the end of the protocol description, one can observe that we have applied quantum gates on these shared inputs, so all the input values and intermediate values are actually qubits.)

Now, suppose at any given time they have to compute an operation on two intermediate values $a = a_0 \oplus a_1$ and $b = b_0 \oplus b_1$ in the actual circuit, and Alice has knowledge of only a_0, b_0 and Bob has knowledge of only a_1, b_1. Alice wants to compute c_0 and Bob wants to compute c_1, such that $c = c_0 \oplus c_1$ and the output of the operation on a, b is c.

Process of generating c_0 and c_1, shares of c, using a_0, a_1, b_0, b_1:

1. If the operation is '$\oplus$', then
   (a) Alice computes $c_0 = a_0 \oplus b_0$. (b) Bob computes $c_1 = a_1 \oplus b_1$.
   We can observe that c_0 and c_1 are the proper shares of $c = a \oplus b$.
2. If the operation is '$\neg$', then
   (a) Alice computes $c_0 = \neg a_0 \oplus 1$. (b) Bob computes $c_1 = \neg a_1$.
   We can observe that c_0 and c_1 are the proper shares of $c = \neg a$.
3. If the operation is '$\wedge$', then
   (a) To compute the share c_0 of $c = a \wedge b$ Alice does the followings:
   - Alice computes $(a_0 \wedge b_0)$ and then asks Bob to participate in an OT protocol to compute $a_1 \wedge b_0$ as a_1 is only known to Bob.
   - Bob knows the value a_1. It sets OT messages $m_i = a_1 \wedge i$ for $i \in \{0, 1\}$
   - Then Alice and Bob run OT with (m_0, m_1) as Bob's messages and b_0 as Alice's query input.
   - After the completion of the protocol, Alice gets $(a_1 \wedge b_0)$.
   - Then it computes $c_0 = (a_0 \wedge b_0) \oplus (a_1 \wedge b_0)$.

   (b) Similarly to compute share c_1 of $c = a \wedge b$, Bob does the followings.
   - Bob computes $(a_1 \wedge b_1)$ and then asks Alice to participate in an OT protocol to compute $a_0 \wedge b_1$ as a_0 is only known to Alice.
   - Alice sets OT messages $m_i = a_0 \wedge i$ for $i \in \{0, 1\}$
   - Then Alice and Bob run OT with (m_0, m_1) as Alice's messages and b_1 as Bob's query input.
   - After the completion of the protocol, Bob gets $(a_0 \wedge b_1)$.
   - Then it computes $c_1 = (a_1 \wedge b_1) \oplus (a_0 \wedge b_1)$.

   We can observe that c_0 and c_1 are the proper shares of $c = a \wedge b$.

Computation: As the quantum circuit of f is already known publicly, and Alice and Bob know what are there corresponding circuits are for each gate presented in the actual circuit (If there is a CNOT gate in the actual circuit, they will use a CNOT gate in their corresponding circuits, with setting the control bit as $|1\rangle$ (as they are constructing these circuits locally for themselves only, they can set any control bits as of their choices without any vulnerability). If there is an X gate in the actual circuit, then Bob uses an X gate only in that position of his circuit, but Alice uses an X gate followed by a CNOT gate on that qubit outcome of the X gate, controlled by $|1\rangle$. If there is a Toffoli gate in the actual circuit, then they use one Toffoli gate on their own shares and use two OT circuits. Alice sets two messages, using her 1st share (using two Toffoli gates), as two inputs in the 1st OT circuit and uses her 2nd share as the query input to the 2nd OT circuit. Then she applies a CNOT gate on the outcome of the 1st Toffoli gate and the 2nd OT. In Bob's case, he uses his 2nd share as the query input to the 1st OT circuit, and he sets two messages, using his 1st share (using two Toffoli gates), as two inputs in the 2nd OT circuit. Then he applies a CNOT gate on the outcome of the 1st Toffoli gate and the 1st OT.), so they can build their corresponding full circuits beforehand.

After running their corresponding full circuits on their individual shares, Alice gets some $g(x_0, x_y)$, and Bob gets some $h(x_1, y_1)$, such that $f(x, y) = g(x_0, y_0) \oplus h(x_1, y_1)$. Now they send their outputs to each other. Then both of them just compute the XOR of those two outputs and reconstruct the actual value of $f(x, y)$.

Correctness of the protocol: As we have shown that our secret generation process for any intermediate value is correct, it follows that the final output of the 2PC protocol is also correct if both the party follow the protocol correctly.

Security of the protocol: For security, we have to assure that any party does not get any extra information regarding others' secret input from the interactions during the protocol, other than the information that can be achieved from the final output and its own share. Now, in this protocol, the parties interact with each other only when they have to run the OT protocol. Now, if the OT protocol assures security against each other, then the 2PC protocol also assures security against each other. Now, for the security of OT, we need the assumption of the existence of a quantum one-way permutation. So, we can conclude that our 2PC protocol is also secure with the assumption of the existence of a quantum one-way permutation.

4.3.2 Secure and Fair Q2PC Protocol in Real World

We name the quantum 2PC protocol of Section 4.3.1 as GF. Here, the main concern is to determine whether we can guarantee all the functionalities of the trusted third party using the described quantum-secure 2PC protocol GF or not.

- Clearly, it can guarantee that without revealing the whole secret to any third party, the parties can generate the valid shares of actual output just by sharing the shares of the secret input among each other.
- But the trusted third party also ensures that only in the i-th round, P_1 gets the share of actual output through f_1, and only in the j-th round P_2 gets the share of actual output through f_2.
- Also, in all other rounds, parties receive shares of $\perp$ rather than the actual output.

So, as we can see, just having a quantum 2PC to generate output shares from two inputs is not enough to replace the third party of our protocol of the hybrid model. Now, the most important question is whether it is possible to achieve the required functionalities of the trusted third party using the given quantum-secure 2PC protocol. The answer is yes, if we make some modifications to the share generation and output reconstruction phases of the protocol. We have found a way out of this issue. For this, we present a real-world Q2PC protocol that uses only the protocol GF (explained in Section 4.3.1) as a sub-protocol and achieves complete fairness using the Algorithm 3 for share generation and the Algorithm 4 for reconstruction.

Note: Before going to the proofs of security and complete fairness, let's clarify some notations first. As shown in the above protocol, we have designated the protocol GF of Section 4.3.1 as a sub-protocol in step 6 of the share generation algorithm. Here in the real world, this sub-protocol GF is basically our ρ, and the whole protocol, combining the algorithms for share generation and reconstruction, is our π^ρ. So, in our case, we consider the hybrid model protocol π as exactly the same as π^ρ, except for step 6 of the share generation algorithm. In π, in step 6, the parties call a trusted third party, provide their inputs, and receive the corresponding output shares (in π^ρ, this trusted third party has been replaced by protocol ρ). Now to prove the fairness and security of π^ρ, we are first going to prove that protocol π achieves complete fairness.

Theorem 5. *Protocol π achieves complete fairness.*

Proof. We have to show that if the dishonest party gets the correct output, then the honest party also must get the correct output. The reconstruction phase of π and π^ρ is exactly same. (Also, one can observe that other than the checking portion the whole reconstruction process is basically similar as our hybrid model protocol given in Section 3.) Because of the checking condition 1, in the reconstruct phase we can be assured that the parties have used their inputs correctly as per the given protocol; otherwise they can not pass these checking conditions. Now, there are the following three possibilities:

Algorithm 3: Share Generation Phase

Input: Inputs $z_i \in P_1, z_j \in P_2$ from $[z_1, z_N]$; test sets S_1, S_2 with $|S_1| = |S_2| = N$, $s < z_1, \forall s \in S_1$ and $s > z_N, \forall s \in S_2$.

Output: Locally encoded states $E_1^{(k)}, E_2^{(k)}$ for $k \in \{1, \ldots, N\}$.

// Notations: $\delta_{a,b} = 1$ if $a = b$ and 0 if $a \neq b$; $Z^s = Z$ if $s = 1$ and Identity if $s = 0$.

for $k \leftarrow 1$ **to** N **do**

  // $x_a^{(k)}$ is P_1's input for the function f_a in the k-th round, and $y_a^{(k)}$ is P_2's input for the function f_a in the k-th round; s_{wk} is the k-th element of the set S_w, where $w \in \{1, 2\}$

  $(x_1^{(k)}, x_2^{(k)}) \leftarrow (\delta_{k,i} z_i + (1 - \delta_{k,i}) s_k,\ z_i)$, with $s_k \in \{s_{1k}, s_{2k}\}$

  $(y_1^{(k)}, y_2^{(k)}) \leftarrow (z_j,\ \delta_{k,j} z_j + (1 - \delta_{k,j}) s_k)$, with $s_k \in \{s_{1k}, s_{2k}\}$

// Distribute Bell pairs and encode private 2PC shares

for $k \leftarrow 1$ **to** N **do**

  for $m \in \{1, 2\}$ **do**

    Distribute $|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ as $E_m^{(k)}$ (P_1 receives qubit 1, P_2 receives qubit 2)

    // P_1,P_2 runs the protocol GF on their private inputs $x_m^{(k)}$ and $y_m^{(k)}$ to generate shares of the output $f(x_m^{(k)}, y_m^{(k)})$ in their ends

    $(o_1, o_2) \leftarrow GF(x_m^{(k)}, y_m^{(k)})$ // private binary shares, P_i gets o_i

    Party P_1 applies Z^{o_1} to its qubit of $E_m^{(k)}$ Party P_2 applies Z^{o_2} to its qubit of $E_m^{(k)}$

1. No one has aborted before the end of the protocol. Then we have no problem. Both parties get the correct output.
2. Honest party has aborted the protocol because of an inconsistency in the revealed output value in a certain round of the reconstruction phase. That means the dishonest party has already got its desired output. Now, if the honest party has already gotten its output, then there is no problem. Otherwise, if the honest party is P_1, then it outputs 1, and if the honest party is P_2, then it outputs 0.
3. Dishonest party has aborted the protocol. That means that, here too, the dishonest party has already obtained its desired output. Here too, if the honest party has already received its output, there is no problem. Otherwise, if the honest party is P_1, then it outputs 1, and if the honest party is P_2, then it outputs 0.

So, protocol π achieves complete fairness. □

Theorem 6. *Our proposed real-world Q2PC protocol π^ρ achieves both security and complete fairness.*

Proof. We have already explained, in Section 4.3.1, that real world protocol ρ achieves security. And in Theorem 5, we have shown that π achieves complete fairness. As in π^ρ, the trusted third party of π has been replaced by a real-world protocol ρ, then using Theorem 4, we can conclude that our prescribed real-world Q2PC protocol (π^ρ) achieves both security and complete fairness. □

All the checks mentioned here, which are to be proven by zero-knowledge proofs, can be done in polynomial time very easily, so it is a problem in the P class. We know that every problem in NP class has a zero-knowledge proof [18]. Since P $\subseteq$ NP, there exists a corresponding zero-knowledge proof to solve our problem. We have designed a dedicated ZKP protocol to efficiently handle our problem. Here, we provide an overview of our ZKP protocol in the latter part of this section.

4.3.3 Overview of the ZKP Protocol

Party P_1 (and symmetrically P_2) proves that its committed per-round inputs $u_k = x_1^{(k)}$ to its own function and $v_k = x_2^{(k)}$ to the other function satisfy $\mathcal{R}_{\mathsf{cons}}$ (Definition 3). That is, there are a one-hot vector $e \in \{0, 1\}^N$

Algorithm 4: Reconstruction Phase

Input: Locally held halves of the entangled states $E_1^{(k)}, E_2^{(k)}$ for $k \in \{1, \ldots, N\}$; Private target round indices $i, j \in \{1, \ldots, N\}$; Input tuples $T_{a,b}$ $(a, b \in \{1, 2\})$, where $T_{a,b}$ denotes party P_a's inputs for function f_b

Output: P_1 outputs $f_1(z_i, z_j)$; P_2 outputs $f_2(z_i, z_j)$ (or aborts on detected cheating).

`// Phase 1: Quantum Zero-Knowledge Verification (`Π_{QZKP}`)`

Execute Π_{QZKP} where each party P_a $(a \in \{1, 2\})$ proves:
(i) $|T_{a,a} \cap [z_1, z_N]| = 1$ and $|T_{a,a} \cap (S_1 \cup S_2)| = N - 1$ with distinct values
(ii) $T_{a,3-a}$ is constant across rounds with a single value in $[z_1, z_N]$
(iii) Index consistency: $T_{a,1} \cap T_{a,2} \cap [z_1, z_N]$ is a singleton set

if Π_{QZKP} *verification fails for either party* **then abort** protocol

`// Phase 2: Round-by-Round Reconstruction and Output Verification`

for $k \leftarrow 1$ **to** N **do**

  `// Iterate over symmetrical transfers:`
  $(a =$ `sender`$, b =$ `receiver`$, m =$ `state index`$, t =$ `target round`$)$

  for $(a, b, m, t) \in \{(2, 1, 1, i),\ (1, 2, 2, j)\}$ **do**

    P_a transmits its qubit of $E_m^{(k)}$ to P_b

    P_b measures $E_m^{(k)}$ in Bell basis, and set $v_m^{(k)} = c$ for measurement result $\frac{|00\rangle + (-1)^c |11\rangle}{\sqrt{2}}$

    if $k = t$ **then**

      $out_b \leftarrow v_m^{(k)}$

    else if $v_m^{(k)}$ *does not match expected output on garbage input* **then**

      abort and report cheating by party P_a `// as` P_b`'s input is from` S_1 `or` S_2

return P_1 outputs out_1, P_2 outputs out_2

and a value $z \in [z_1, z_N]$ with $v_k = z$ for all k, $u_k = z$ when $e_k = 1$, and $u_k \notin [z_1, z_N]$ otherwise. Without this, a cheating party could submit in-range values in several rounds of its own function and binary-search the other party's input, or use different values for the two functions.

How is the goal achieved? The witness vectors are encoded by multilinear extensions over $\{0, 1\}^{\log N}$. All pointwise constraints (booleanity, selection, and range via bit decomposition) are batched into one zero-check. The global constraints ($\sum_k e_k = 1$ and constancy of v) are folded into the same sumcheck. Zero-knowledge comes from a masked sumcheck [11, 50] with randomized extensions, and the polynomials are committed with a post-quantum polynomial commitment (Section 6.4). The resulting argument is perfectly complete, knowledge-sound against quantum polynomial-time provers, and computationally zero-knowledge (Theorem 9). The commitments are the ones through which inputs enter GF (Section 6.7), so the argument constrains the actual execution.

5 Quantum Protocol for Embedded XOR

We suitably modify the classical protocol by [20] to propose a quantum solution to the embedded XOR problem. As in the quantum protocol to solve the millionaires' problem, here also we exploit entangled states to obtain security.

Now we describe the protocol. Let P_1 is given an ordered list $\{x_1, x_2, x_3\}$ and P_2 is given an ordered list $\{y_1, y_2\}$. P_1 randomly chooses an input x from his ordered list and sends it to the dealer. Similarly, P_2 also chooses an input y randomly from his ordered list and sends it to the dealer. Dealer computes the function and creates two sequences $\{a_l\}$ and $\{b_l\}$, $l = 1, 2, \ldots, N$, where m is the total number of the rounds.

For $l \geq r$, $a_l = f_1(x, y) = b_l = f_2(x, y)$, and for $l < r$, $a_l = f_1(x, \hat{y})$, $b_l = f_2(\hat{x}, y)$,
where $\hat{x}$ (or $\hat{y}$) is a random value of x (or y) chosen by the dealer. In the quantum domain, the two sequences $\{a_l\}$ and $\{b_l\}$ are distributed by exploiting the qubits of entangled states. The mechanism is described in Algorithm 5 ($QEShareGen$) and Algorithm 6 ($\mathbf{\Pi^{QEP}_{Fair}}$).

5.1 Quantum Protocol in Hybrid Model

This protocol also has two phases: the Share Generation phase and the Reconstruction phase. At first, the dealer chooses r and d following a geometric distribution, and sets $N = r + d$ (N is the total number of rounds).

In the Share Generation phase, the dealer gets the inputs from both the parties, and then checks whether the inputs are valid or not. Then for each round $l(< r)$, it computes $f_1(x, \hat{y})$ and $f_2(\hat{x}, y)$, and for each round $l(\geq r)$ it computes $f(x, y)$. Depending on the computed values, for each round it selects two entangled states from the Bell state $|g_0\rangle$, $|g_1\rangle$. Then it sends the 1st qubit of both the chosen entangled states to P_1 and the remaining 2nd qubit of both the chosen entangled states to P_2. So in each round, both of them receive a list of two qubits. A detailed description of the share generation phase is given in Algorithm 5.

Algorithm 5: QEShareGen

Input: Inputs $x \in P_1, y \in P_2$ from domain $\mathcal{D}$, parameter $\gamma \in (0, 1)$.
Output: Share lists $list_1, list_2$ of $2N$ qubits and $2N$ classical bits each (or $\perp$ if $x, y \notin \mathcal{D}$).

if $x \notin \mathcal{D}$ ***or*** $y \notin \mathcal{D}$ **then return** $\perp$ to both the parties

Sample $r, d \sim \mathcal{G}(\gamma)$; set $N \leftarrow r + d$
Choose X_0, X_1 from $\mathbb{F}_q^d$, and make them public

`// Notation:` $\mathbb{I}_{[l \geq r]} = 1$ `if` $l \geq r$`, else 0;` $|g_c\rangle = \frac{1}{\sqrt{2}}(|00\rangle + (-1)^c |11\rangle)$ `for` $c \in \{0, 1\}$
for $l \leftarrow 1$ **to** N **do**
  Sample dummy inputs $\hat{x}$ and $\hat{y}$ from ordered lists of P_1 and P_2
  $(a_l, b_l) \leftarrow \Big(f_1\big(x,\, \mathbb{I}_{[l \geq r]} y + (1 - \mathbb{I}_{[l \geq r]})\hat{y}\big),\, f_2\big(\mathbb{I}_{[l \geq r]} x + (1 - \mathbb{I}_{[l \geq r]})\hat{x},\, y\big)\Big)$
  `// Use of AMD code for authentication`
  $rand_1, rand_2 \xleftarrow{\$} \mathbb{F}_q$
  $t_1 \leftarrow {rand_1}^{d+2} + \Sigma_{ind=1}^{d} X_{a_l, ind} \cdot {rand_1}^{ind}$
  $t_2 \leftarrow {rand_2}^{d+2} + \Sigma_{ind=1}^{d} X_{b_l, ind} \cdot {rand_2}^{ind}$
  `// Distribute Bell states` $|g_{a_l}\rangle, |g_{b_l}\rangle$ `to lists at round` l
  $(list_1[l, 1], list_2[l, 1]) \leftarrow$ halves of prepared $|g_{a_l}\rangle$
  $(list_1[l, 2], list_2[l, 2]) \leftarrow$ halves of prepared $|g_{b_l}\rangle$
  $(list_1[l, 3], list_1[l, 4]) \leftarrow (rand_2, t_1)$
  $(list_2[l, 3], list_2[l, 4]) \leftarrow (rand_1, t_2)$

return $(list_1, list_2)$ to P_1 and P_2

In the Reconstruction phase, in each round, at first P_2 sends the first qubit of its list for that round to P_1, and after that P_1 sends the second qubit of its list for that round to P_2. Upon receiving the shares, they measure the quantum states in the Bell basis. P_1 and P_2 both gets the actual output $f(x, y)$ in each round l with $l \geq r$. For the detailed description of the reconstruction phase, check Algorithm 6.

5.2 Security Analysis Against Forgery and Early Abort

In this subsection, we discuss the security issues against a Byzantine adversary. First, we analyze the sensitivity of our protocol to detect cheating by a Byzantine player. Then we analyze the fairness issue when a player aborts early.

Theorem 7. *In our setting, in an arbitrary round $l \leq r$, the failure probability of an honest player to detect cheating by a player who is corrupted by a Byzantine adversary is $\frac{d+1}{2q}$.*

Algorithm 6: $\Pi_{\mathsf{Fair}}^{\mathsf{QEP}}$

Input: Share lists $list_1, list_2$ of $2N$ qubits and $2N$ classical bits each, secret revelation round r, $X_0, X_1 \in \mathbb{F}_q^d$.

Output: P_1 outputs evaluation $a \in \{0,1\}$; P_2 outputs evaluation $b \in \{0,1\}$.

for $l \leftarrow 1$ **to** N **do**

  `// Symmetrical transfer loop:` $(a = \texttt{sender}, b = \texttt{receiver})$

  for $(a,b) \in \{(2,1), (1,2)\}$ **do**

    if P_a *aborts or fails to send qubit* **then**

      return P_b returns the outcome of round $l-1$

    P_a sends $list_a[l,b]$ and $list_a[l,3]$ to P_b

    P_b measures $(list_1[l,b], list_2[l,b])$ in Bell basis

    if *Measured state is not* $|g_0\rangle$ *or* $|g_1\rangle$ **then**

      abort and report forgery by P_a

    `// For measured state` $|g_c\rangle$`, check whether it is genuine or tampered through AMD code verification`

    $|g_c\rangle$ is the measured state for $c \in \{0,1\}$

    $rand \leftarrow list_a[l,3]$

    $t \leftarrow list_b[l,4]$

    $t' \leftarrow rand^{d+2} + \Sigma_{ind=1}^{d} X_{c,ind} \cdot rand^{ind}$

    if $t \neq t'$ **then**

      abort and report forgery by P_a

    else

      Record c as outcome for round l

return P_1 and P_2 both outputs the the outcome of round l

Proof. Without loss of generality, we assume that P_1 is corrupted by the Byzantine adversary and tries to manipulate the qubits. According to our protocol, in any round $l \leq r$, if P_1 does not cheat, P_2 will measure either $|g_0\rangle$ or $|g_1\rangle$ depending on the value of b_l. However, when P_1 cheats, the case will be different. Let us assume that in round $l \leq r$, P_1 sends an arbitrary qubit $|\phi\rangle = \alpha |0\rangle_3 + \beta |1\rangle_3$ to P_2. *Here, we assume that if P_1 would not cheat at the round l, P_2 would receive $|g_0\rangle$. The same thing happens if we assume that P_2 will receive $|g_1\rangle$.* Thus the final state at the end of P_2 would be

$$\begin{aligned}\rho_2 &= [tr_{P_1}(\rho)](\rho_3) \\ &= \frac{1}{2}\Big[|1\rangle_2 \langle 1| \Big(|\alpha|^2 |0\rangle_3 \langle 0| + \alpha^*\beta |1\rangle_3 \langle 0| + \alpha\beta^* |0\rangle_3 \langle 1| + |\beta|^2 |1\rangle_3 \langle 1| \Big) \\ &\quad + |0\rangle_2 \langle 0| \Big(|\alpha|^2 |0\rangle_3 \langle 0| + \alpha^*\beta |1\rangle_3 \langle 0| + \alpha\beta^* |0\rangle_3 \langle 1| + |\beta|^2 |1\rangle_3 \langle 1| \Big)\Big],\end{aligned}$$

where $\rho = \frac{1}{2}(|0\rangle_1 |1\rangle_2 + |1\rangle_1 |0\rangle_2)(\langle 0|_1 \langle 1|_2 + \langle 1|_1 \langle 0|_2)$ and $\rho_3 = [|\alpha|^2 |0\rangle_3 \langle 0| + \alpha^*\beta |1\rangle_3 \langle 0| + \alpha\beta^* |0\rangle_3 \langle 1| + |\beta|^2 |1\rangle_3 \langle 1|]$.

P_2 will measure qubit 2 and qubit 3. Thus, after measurement, P_2 will get $|g_0\rangle$ or $|g_1\rangle$ or $|g_2\rangle$ or $|g_3\rangle$ with probability $\frac{1}{4}$ instead of $|g_0\rangle$ only (see Section 3.2.1). As in round $l \leq r$, P_2 will measure either $|g_0\rangle$ or $|g_1\rangle$ without any cheating, when he gets $|g_0\rangle$ or $|g_1\rangle$, he can not detect cheating solely by entanglement property. If he gets $|g_2\rangle$ or $|g_3\rangle$, he immediately concludes that P_1 is cheating. But we have AMD code for extra layer of verification, whose probability of undetected cheating is $\frac{d+1}{q}$. Thus, the probability of undetected cheating, both by entanglement and AMD code, for any round $l \leq r$ is $(\frac{1}{2} \cdot \frac{d+1}{q})$. After the revelation round, P_2 has no incentive to detect cheating as P_2 has already got the correct output. Thus, we can write the expected

failure probability of P_2 conditioning on the event that $l \leq r$ to detect cheating by P_1 is

$$\begin{aligned}&(\frac{1}{2}\cdot\frac{d+1}{q})\Pr(l<r)+(\frac{1}{2}\cdot\frac{d+1}{q})\Pr(l=r)\\&=(\frac{1}{2}\cdot\frac{d+1}{q})(1-\gamma)+(\frac{1}{2}\cdot\frac{d+1}{q})\gamma=\frac{d+1}{2q}.\end{aligned}$$

The same situation arises when we assume that P_2 is cheating. □

The following result shows that the fairness condition is maintained when a player corrupted by a Byzantine adversary aborts the protocol prematurely.

Theorem 8. *In an arbitrary round l, the protocol $\mathbf{\Pi}^{\mathbf{QEP}}_{\mathbf{Fair}}$ achieves fairness considering the early abort of a corrupted player.*

Proof. Let us assume that P_1 aborts in round l. As P_1 is always computing its output first, followed by P_2, the aborting of P_1 plays an important role in achieving the fairness of the protocol. The early abort of P_2 will terminate the protocol up to that round in which P_2 aborts. In that case, either both get the correct value or neither gets the correct value. Thus, early abort of P_2 does not affect the fairness condition.

According to our protocol, if P_1 aborts in a round $l < r$, P_2 outputs $b_{l-1} = f_2(\hat{x}, y)$. In this case P_1 outputs a_l which is equal to $f_1(x, \hat{y})$. In *ideal model*, for $l < r$ the trusted party sends $f_1(x, \hat{y})$ to P_1 and $f_2(\hat{x}, y)$ to P_2. Thus, if P_1 aborts in round $l < r$, P_1 gets $f_1(x, \hat{y})$ and the trusted party sends $f_2(\hat{x}, y)$ to P_2. Thus for $l < r$, we get

$$\begin{aligned}&\Pr\Big[\Big(VIEW_{ideal}(x,y),OUT_{ideal}(x,y)\Big)=(f(x,\hat{y}),f(\hat{x},y))|l<r\Big]\\&=\Pr\Big[\Big(VIEW_{hybrid}(x,y),OUT_{hybrid}(x,y)\Big)=(f(x,\hat{y}),f(\hat{x},y))|l<r\Big].\end{aligned}$$

When $l = r$, P_1 has already got the correct output whereas P_2 outputs $b_{r-1} = f_2(\hat{x}, y)$. In *ideal model*, when $l = r$, the trusted party sends $f_1(x, y)$ (res. $f_2(x, y)$) to P_1 (res. P_2). The following analysis shows how fairness is maintained in this case.

Here, we first recall the table for embedded XOR. We get that in the case of P_1, $\Pr[f_1(x_1, \hat{y}) = 0] = \Pr[f_1(x_2, \hat{y}) = 0] = \Pr[y \in \{y_1, y_2\}] = \frac{1}{2}$ and $\Pr[f_1(x_1, \hat{y}) = 1]$
$= \Pr[f_1(x_2, \hat{y}) = 1] = \Pr[y \in \{y_1, y_2\}] = \frac{1}{2}$, whereas $\Pr[f_1(x_3, \hat{y}) = 0] = 0$ and $\Pr[f_1(x_3, \hat{y}) = 1] = 1$. In case of P_2, $\Pr[f_2(\hat{x}, y) = 0] = \Pr[x = x_1] = \frac{1}{3}$ and $\Pr[f_2(\hat{x}, y) = 1] = \Pr[x \in \{x_2, x_3\}] = \frac{2}{3}$. Thus, we can write the following:

$$\begin{aligned}\Pr\Big[\Big(VIEW_{ideal}(x,y),OUT_{ideal}(x,y)\Big)=(0,0)\Big|l=r\Big]&=\frac{1}{3}.\frac{1}{3},\\\Pr\Big[\Big(VIEW_{ideal}(x,y),OUT_{ideal}(x,y)\Big)=(0,1)\Big|l=r\Big]&=\frac{1}{3}.\frac{2}{3},\\\Pr\Big[\Big(VIEW_{ideal}(x,y),OUT_{ideal}(x,y)\Big)=(1,0)\Big|l=r\Big]&=\frac{2}{3}.\frac{1}{3},\\\Pr\Big[\Big(VIEW_{ideal}(x,y),OUT_{ideal}(x,y)\Big)=(1,1)\Big|l=r\Big]&=\frac{2}{3}.\frac{2}{3}.\end{aligned}$$

Similarly, in a hybrid world,

$$\begin{aligned}\Pr\Big[\Big(VIEW_{hybrid}(x,y),OUT_{hybrid}(x,y)\Big)=(0,0)\Big|l=r\Big]&=\frac{1}{3}.\frac{1}{3},\\\Pr\Big[\Big(VIEW_{hybrid}(x,y),OUT_{hybrid}(x,y)\Big)=(0,1)\Big|l=r\Big]&=\frac{1}{3}.\frac{2}{3},\\\Pr\Big[\Big(VIEW_{hybrid}(x,y),OUT_{hybrid}(x,y)\Big)=(1,0)\Big|l=r\Big]&=\frac{2}{3}.\frac{1}{3},\\\Pr\Big[\Big(VIEW_{hybrid}(x,y),OUT_{hybrid}(x,y)\Big)=(1,1)\Big|l=r\Big]&=\frac{2}{3}.\frac{2}{3}.\end{aligned}$$

The above probability calculations show that when $l = r$, the adversary does not do more harm in the hybrid world than he can do in the ideal world. Thus, our protocol achieves fairness.

Fairness is obvious if we consider the abort of P_1 at a round $l > r$, as in this situation, both in the ideal world and in the hybrid world, P_1 as well as P_2 obtain the correct output in iteration r. □

We can actually build a quantum 2PC protocol for embedded XOR in real-world model also using the similar kind of idea as Section 4.

5.3 Quantum 2PC Protocol for Embedded XOR in Real-World Model

Here the function f returns 0 iff $i = j$. Now, if we remember the 2PC which we have constructed in Section 4, it returns 0 iff $i \leq j$. Let's name the previous 2PC protocol as g here. Then we can write $f(x_i, y_j) = g(x_i, y_j) \vee g(y_j, x_i)$. In fact, f returns 0 iff both $g(x_i, y_j)$ and $g(y_j, x_i)$ returns 0, i.e. if both $i \leq j$ and $j \leq i$ holds simultaneously, i.e. $i = j$. So we can generate a secure 2PC easily to handle this specific functionality in the real world.

Now for the other functionalities of the dealer or trusted third party, here the dealer chooses a random value r following a geometric distribution, and the actual function values are revealed in each round l, where $l \geq r$. To maintain this functionality, we can do the following:

1. Each party P_i first chooses a set of random values following a geometric distribution. Say the name of the set is R_i. Then it uses ZKP to prove to the other party that these values have indeed been chosen following a geometric distribution (for $i \in \{1, 2\}$).
2. Then the party P_i chooses one random element r_i from the set R_i.
3. After that, they set their inputs for upcoming share generation rounds as follows:
   - Party P_1 sets the inputs for the function f_1 for each round as its actual secret input. Then, for the function f_2, it sets the input as secret input for any round l with $l \geq r_1$, but for the other rounds, it sets any garbage value (as described in Section 4.3.2) as the input of that round.
   - Similarly, Party P_2 sets the inputs for the function f_2 for each round as its actual secret input. Then, for the function f_1, it sets the input as secret input for any round l with $l \geq r_2$, but for the other rounds, it sets any garbage value (as described in Section 4.3.2) as the input of that round.
4. Then they proceed in an exactly similar way as previously described in Section 4.3.2.

6 Details of the Zero-Knowledge Argument

This section specifies the zero-knowledge argument Π_{QZKP} run in Phase 1 of Algorithm 4. Section 6.1 states the relation, Section 6.2 arithmetizes it, Section 6.3 lists the zero-knowledge ingredients, Section 6.4 states the requirements on the polynomial commitment, Section 6.5 gives the protocol and Section 6.6 its security. Section 6.7 explains how the argument is bound to the execution of GF.

6.1 The Relation

All inputs are integers in $\{0, \ldots, 2^\ell - 1\}$ for a public ℓ, with $0 < z_1 \leq z_N < 2^\ell - 1$, so that values below z_1 (the set S_1) and above z_N (the set S_2) exist. We work over a prime field $\mathbb{F}$ with $|\mathbb{F}| > 2^{\ell+2}$ and $|\mathbb{F}| \geq 2^{\Omega(\lambda)}$, and identify integers in $(-|\mathbb{F}|/2, |\mathbb{F}|/2)$ with field elements.

For the prover P_a, let $u = (u_k)_{k\in[N]}$ be its per-round inputs to its own function f_a and $v = (v_k)_{k\in[N]}$ its per-round inputs to the other function f_{3-a}. For P_1, $u_k = x_1^{(k)}$ and $v_k = x_2^{(k)}$; for P_2, $u_k = y_2^{(k)}$ and $v_k = y_1^{(k)}$ (Algorithm 3). Each is committed bitwise: $u_k = \sum_{t=0}^{\ell-1} 2^t u_k^{(t)}$ and $v_k = \sum_{t=0}^{\ell-1} 2^t v_k^{(t)}$ with $u^{(t)}, v^{(t)} \in \{0,1\}^N$.

Definition 3 (Input-consistency relation $\mathcal{R}_{\mathsf{cons}}$). *The statement consists of the public range $[z_1, z_N]$ and commitments $\mathsf{cm}_u = (\mathsf{cm}_{u^{(t)}})_t$, $\mathsf{cm}_v = (\mathsf{cm}_{v^{(t)}})_t$ to the bit vectors of u and v. The witness consists of openings of these commitments and vectors $e \in \{0,1\}^N$, $\sigma \in \{0,1\}^N$, $g \in \mathbb{F}^N$ such that, for all $k \in [N]$:*

(C1) $e_k(e_k - 1) = 0$ *and* $\sum_k e_k = 1$*;*

(C2) $u_k = e_k\, v_k + (1 - e_k)\, g_k$*;*
(C3) $v_k = v_1$*;*
(C4) $z_1 \le v_k \le z_N$*;*
(C5) $\sigma_k(\sigma_k - 1) = 0$*; if* $\sigma_k = 0$ *then* $g_k < z_1$*, and if* $\sigma_k = 1$ *then* $g_k > z_N$*.*

Consequence. If $\mathcal{R}_{\mathsf{cons}}$ holds, let k^* be the unique index with $e_{k^*} = 1$ and $z := v_1$. Then $z \in [z_1, z_N]$, the input to the other function equals z in every round, $u_{k^*} = z$, and every other u_k lies strictly below z_1 or strictly above z_N. This is exactly what the trusted dealer guaranteed in the hybrid model (Section 4.1).
Why each constraint is needed. (C1)+(C2) say that exactly one round of the party's own function uses a real input. Without them, a cheating party could submit in-range values in several rounds and binary-search the other party's input. (C3) is needed because the party does not know in which round the other party's real output is revealed; without it, the party could use different values for the two functions. (C2)+(C3) together also give index and value consistency: the value used in the real round of f_a is the value used for f_{3-a} in every round. (C4) and (C5) are range conditions; (C5) replaces membership in $S_1 \cup S_2$, since S_1 lies below z_1 and S_2 above z_N, and the receiver's check in Phase 2 of Algorithm 4 only uses on which side a garbage value lies.
Remark. We do not require garbage values to be distinct. Distinctness would need a permutation or sorting argument, and the security analysis only uses that each garbage output is fixed by the side (σ_k) on which the garbage value lies.

$\mathcal{R}_{\mathsf{cons}}$ is an NP relation, so zero-knowledge proofs for it that remain secure against quantum verifiers exist assuming quantum-computationally concealing commitments [48, 49]. The remainder of this section gives a more efficient argument.

6.2 Arithmetization

Without loss of generality $N = 2^\mu$; otherwise pad with positions $k > N$ where $e_k = 0$, $\sigma_k = 0$, $u_k = g_k = 0$ and $v_k = v_1$. Padded positions are never used by the protocol. We index positions by $k \in \{0,1\}^\mu$ and write $\tilde{w}$ for the multilinear extension of a vector $w \in \mathbb{F}^N$, and

$$\widetilde{\mathrm{eq}}(X, Y) = \prod_{t=1}^{\mu} \big(X_t Y_t + (1 - X_t)(1 - Y_t)\big), \qquad \tilde{w}(X) = \sum_{k \in \{0,1\}^\mu} \widetilde{\mathrm{eq}}(X, k)\, w_k.$$

Witness vectors. Besides the committed bit vectors $u^{(t)}, v^{(t)}$ and the vectors e, σ, g, the prover commits to bit vectors $a^{(t)}, a'^{(t)}, b^{(t)} \in \{0,1\}^N$ for $t \in \{0, \dots, \ell - 1\}$ (range proofs). We write $u = \sum_t 2^t u^{(t)}$ and $v = \sum_t 2^t v^{(t)}$; these are linear in committed vectors, so their extensions are computable from the openings.
Pointwise constraints. Let $Q_1, \dots, Q_m$ (with $m = 5\ell + 6$) be the following polynomials in the entries at position k, each of degree at most 2:

$w_k(w_k - 1)$	for $w \in \{e, \sigma\} \cup \{u^{(t)}, v^{(t)}, a^{(t)}, a'^{(t)}, b^{(t)}\}_t$,
$u_k - e_k v_k - (1 - e_k) g_k$	(C2),
$v_k - z_1 - \sum_t 2^t a_k^{(t)}, \qquad z_N - v_k - \sum_t 2^t a_k'^{(t)}$	(C4),
$(1 - \sigma_k)(z_1 - 1 - g_k) + \sigma_k(g_k - z_N - 1) - \sum_t 2^t b_k^{(t)}$	(C5).

Since $u_k, v_k \in [0, 2^\ell)$ and $|\mathbb{F}| > 2^{\ell+2}$, each equation in $\mathbb{F}$ holds over the integers whenever it holds in $\mathbb{F}$, so the bit decompositions certify the inequalities. When $e_k = 0$, (C2) gives $g_k = u_k \in [0, 2^\ell)$, so (C5) constrains u_k itself.
Zero-check. All Q_j vanish on $\{0,1\}^\mu$ iff the multilinear polynomial with coefficients $\sum_j \gamma^j Q_j(w_k)$ is zero. For random $\gamma \in \mathbb{F}$ and $\tau \in \mathbb{F}^\mu$, if some Q_j is non-zero somewhere on the cube then

$$\Pr_{\gamma,\tau}\Big[\textstyle\sum_k \widetilde{\mathrm{eq}}(\tau, k) \sum_j \gamma^j Q_j(w_k) = 0\Big] \le \tfrac{m+\mu}{|\mathbb{F}|}$$

by the Schwartz–Zippel lemma [39, 56].
Global constraints. (C1) is the sum $\sum_k e_k = 1$. For (C3) we use the following fact.

Lemma 1. *For $v \in \mathbb{F}^N$ and uniformly random $\tau', \tau'' \in \mathbb{F}^\mu$: if v is constant then $\sum_k \big(\widetilde{\mathrm{eq}}(\tau', k) - \widetilde{\mathrm{eq}}(\tau'', k)\big) v_k = 0$; otherwise this sum is non-zero except with probability at most $\mu/|\mathbb{F}|$.*

Proof. The sum equals $\tilde{v}(\tau') - \tilde{v}(\tau'')$. A multilinear polynomial is constant iff its values on the cube are equal. If v is not constant, $\tilde{v}(X) - \tilde{v}(Y)$ is a non-zero polynomial of total degree at most μ, and the Schwartz–Zippel lemma gives the bound. □

One sumcheck. With verifier randomness $\rho_0, \rho_2 \in \mathbb{F}$ (and $\rho_1 := 1$), all constraints reduce to the single claim

$$\sum_{k \in \{0,1\}^\mu} F(k) = 1, \qquad F(X) = \rho_0 \, \widetilde{\mathrm{eq}}(\tau, X) \sum_j \gamma^j Q_j(\hat{w}(X)) + \hat{e}(X) + \rho_2 \big(\widetilde{\mathrm{eq}}(\tau', X) - \widetilde{\mathrm{eq}}(\tau'', X)\big) \hat{v}(X), \quad (7)$$

where $\hat{w}$ denotes the randomized extensions of Section 6.3. They agree with $\tilde{w}$ on the cube, so the sum is unchanged. If any of the three component claims fails, (7) fails except with probability $2/|\mathbb{F}|$ over ρ_0, ρ_2. With randomized extensions of individual degree 2 and $\deg Q_j \le 2$, F has individual degree $d \le 5$.

This replaces the univariate quotient checks and the identity "$C(1) = \sum_i c_i$" of earlier versions. That identity does not hold in a Lagrange basis in general, and mixing univariate encodings with a hypercube sumcheck is unsound. An equivalent univariate arithmetization is possible (quotients modulo $Z_{\mathcal{H}}$ plus the univariate sumcheck of [5]); we use the multilinear one throughout, as in [9, 40].

6.3 Zero-Knowledge Ingredients

Plain sumcheck is not zero-knowledge: each round polynomial reveals partial sums of F, and the final evaluations reveal linear functions of the witness. We use two standard countermeasures.

Randomized extensions [50]. For each committed vector w, the prover samples $c_{w,1}, \ldots, c_{w,\mu} \leftarrow \mathbb{F}$ and commits to

$$\hat{w}(X) = \tilde{w}(X) + \sum_{t=1}^{\mu} c_{w,t} \, X_t(1 - X_t).$$

$\hat{w}$ agrees with $\tilde{w}$ on $\{0,1\}^\mu$. Each $\hat{w}$ is opened at a single point r; if $r_t \notin \{0,1\}$ for some t (which fails with probability at most $(2/|\mathbb{F}|)^\mu$), then $\hat{w}(r)$ is uniform and independent of w.

Masked sumcheck [11,50]. The prover commits to a random masking polynomial $p(X) = p_0 + \sum_{t=1}^{\mu} p_t(X_t)$ with each p_t a uniformly random univariate polynomial of degree d, and sends $P = \sum_{k \in \{0,1\}^\mu} p(k)$. After receiving $\rho_p \in \mathbb{F}$, both parties run the sumcheck protocol [32] on $F + \rho_p \, p$ with claimed sum $1 + \rho_p P$. At the end, the prover opens $p(r)$. The round polynomials are then simulatable without the witness; we use this result as proven in [50] and do not reprove it.

6.4 Polynomial Commitment

We require a polynomial commitment scheme PolyCommit for μ-variate polynomials of individual degree at most d with the following properties against quantum polynomial-time adversaries:

(P1) **Hiding:** commitments to any two polynomials are computationally indistinguishable.
(P2) **Collapse-binding** [46]. Computational binding alone is not sufficient for rewinding-based extraction against quantum adversaries.
(P3) **Evaluation arguments** that are complete, knowledge-sound against quantum polynomial-time provers, and honest-verifier zero-knowledge, and that support batching several openings at a common point.

Instantiations. (i) *Hash-based:* a linear-code-based multilinear commitment such as Ligero [1] or Brakedown [19], with salted Merkle leaves for hiding, compiled with the BCS transform [6]. Its soundness in the quantum random oracle model (QROM) follows from [12], and random-oracle-based commitments are collapse-binding [46]. (ii) *Lattice-based:* BDLOP commitments [3] under Module-SIS/Module-LWE, with evaluation arguments such as Greyhound [36] or LaBRADOR-based arguments [7].

Commitments based on discrete logarithms or on groups of unknown order are excluded, since Shor's algorithm [42] breaks them. A Fiat–Shamir-with-aborts proof of knowledge [33, 34] proves knowledge of

Algorithm 7: Opening Proof for a BDLOP-Style Commitment

Input: Public parameters: ring $R_q = \mathbb{Z}_q[X]/(X^n+1)$, uniformly random $A_1 \in R_q^{k\times m_1}$, $A_2 \in R_q^{k\times m_2}$, challenge set $\mathcal{C} \subset R_q$ of short polynomials, norm bound B. Statement: commitment $C \in R_q^k$. Witness: short (r, f) with $C = A_1 r + A_2 f$.

Output: Verifier outputs **accept** or **reject**.

Prover samples $y_r \leftarrow D_\sigma^{m_1}$, $y_f \leftarrow D_\sigma^{m_2}$ and computes $t := A_1 y_r + A_2 y_f$
Prover computes the challenge $c := H(A_1 \,\|\, A_2 \,\|\, C \,\|\, t) \in \mathcal{C}$
Prover computes $z_r := y_r + c\, r$ and $z_f := y_f + c\, f$, and restarts unless rejection sampling accepts [34]

Verifier recomputes $c := H(A_1 \,\|\, A_2 \,\|\, C \,\|\, t)$
if $A_1 z_r + A_2 z_f \neq t + c\,C$ ***or*** $\|(z_r, z_f)\| > B$ **then return reject**
return accept

a short opening; it is not by itself a polynomial commitment with evaluation arguments. For reference, Algorithm 7 gives the opening proof for BDLOP-style commitments used inside instantiation (ii).

Two corrections relative to earlier versions. First, the randomness is part of the committed linear form, so the verification equation holds: $A_1 z_r + A_2 z_f = t + c(A_1 r + A_2 f) = t + cC$. With $C = Af + r$ and $z = y + cf$ one gets $Az = t + c(C - r) \neq t + cC$. Second, challenges are short ring elements, not elements of $\mathbb{Z}_q$. Extraction is *relaxed*: from two accepting transcripts with challenges $c \neq c'$ one obtains short $(\bar{r}, \bar{f})$ with $A_1 \bar{r} + A_2 \bar{f} = \bar{c}\, C$ for $\bar{c} = c - c'$. Binding arguments must therefore be stated for relaxed openings, as in [3].

6.5 The Protocol

Algorithm 8 gives Π_{QZKP} in its Fiat–Shamir form. H is a hash function modeled as a quantum random oracle, and every challenge is H applied to the transcript so far.

Algorithm 8: Π_{QZKP}: Argument for $\mathcal{R}_{\mathsf{cons}}$

Input: Statement: $[z_1, z_N]$ and commitments $\mathsf{cm}_u, \mathsf{cm}_v$ (made in Algorithm 3). Prover's witness as in Definition 3.

Output: Verifier outputs **accept** or **reject**.

`// Commit`
Prover commits with PolyCommit to the randomized extensions $\hat{e}, \hat{\sigma}, \hat{g}$ and $\hat{a}^{(t)}, \hat{a}'^{(t)}, \hat{b}^{(t)}$ for all t, and to a masking polynomial p; sends $P = \sum_k p(k)$
`// Challenges for batching`
$(\gamma, \tau, \tau', \tau'', \rho_0, \rho_2, \rho_p) \leftarrow H(\text{transcript})$
`// Masked sumcheck on` $F + \rho_p\, p$ `(Eq. (7))`
for $i \leftarrow 1$ **to** μ **do**
  Prover sends $s_i(X_i) = \sum_{k_{i+1},\ldots,k_\mu \in \{0,1\}} (F + \rho_p p)(r_1, \ldots, r_{i-1}, X_i, k_{i+1}, \ldots, k_\mu)$, of degree $\leq d$
  Verifier checks $s_i(0) + s_i(1) = 1 + \rho_p P$ if $i = 1$, and $= s_{i-1}(r_{i-1})$ if $i > 1$; else **reject**
  $r_i \leftarrow H(\text{transcript})$
`// Final evaluations`
Prover sends $\hat{w}(r)$ for every committed w and $p(r)$, with one batched evaluation argument at $r = (r_1, \ldots, r_\mu)$
Verifier computes $\widetilde{\mathrm{eq}}(\tau, r), \widetilde{\mathrm{eq}}(\tau', r), \widetilde{\mathrm{eq}}(\tau'', r)$, then $\hat{u}(r) = \sum_t 2^t \hat{u}^{(t)}(r)$ and $\hat{v}(r)$, and from these $F(r)$
if *the evaluation argument fails* ***or*** $s_\mu(r_\mu) \neq F(r) + \rho_p\, p(r)$ **then return reject**
return accept

The verifier never computes a sum over $[N]$ itself. All global constraints are discharged by the sumcheck, and the verifier only evaluates $\widetilde{\mathrm{eq}}$ and the opened values at the single point r.

6.6 Security

Theorem 9. *Assume* PolyCommit *satisfies (P1)–(P3), and let $d \le 5$ and $m = 5\ell + 6$ be as above. Then:*

(a) ***Completeness.*** *Π_{QZKP} is perfectly complete, given a perfectly complete evaluation argument.*

(b) ***Knowledge soundness.*** *Run interactively with public coins, Π_{QZKP} is an argument of knowledge for $\mathcal{R}_{\mathsf{cons}}$ against quantum polynomial-time provers, with knowledge error at most $\big((d+2)\mu + m + 3\big)/|\mathbb{F}|$ plus the knowledge error of the evaluation argument. Its Fiat–Shamir form is an argument of knowledge in the QROM: for instantiation (i) via [12], using round-by-round soundness, and in general via the multi-round measure-and-reprogram technique [16]. The latter loses a factor $(2q+1)^{2(\mu+1)}$ for q oracle queries; this is quasi-polynomial since $\mu = O(\log \lambda)$, so the error remains negligible for $|\mathbb{F}| \ge 2^{\Omega(\lambda)}$.*

(c) ***Zero-knowledge.*** *The interactive protocol is computational honest-verifier zero-knowledge. The Fiat–Shamir form is computational zero-knowledge in the QROM against arbitrary quantum polynomial-time verifiers.*

Proof (Proof sketch). **(a)** An honest witness satisfies (C1)–(C5), hence all Q_j vanish on the cube, $\sum_k e_k = 1$, and v is constant. The randomized extensions agree with $\tilde{w}$ on the cube, so (7) holds, the masked claim $1+\rho_p P$ is correct, and the sumcheck and evaluation arguments accept with probability 1.

(b) The extractor runs the extractor of the evaluation argument (P3) to obtain the committed polynomials. By collapse-binding (P2), these are well defined even when extraction rewinds a quantum prover [46]. Their restrictions to the cube give a candidate witness w. Suppose $w \notin \mathcal{R}_{\mathsf{cons}}$. Then a pointwise constraint fails, or $\sum_k e_k \neq 1$, or v is not constant. Acceptance then requires one of the following events: the zero-check misses ($\le (m+\mu)/|\mathbb{F}|$), Lemma 1 misses ($\le \mu/|\mathbb{F}|$), batching misses ($\le 3/|\mathbb{F}|$, including ρ_p), or the sumcheck accepts a false claim for a polynomial of individual degree d ($\le d\mu/|\mathbb{F}|$ [32]). A union bound gives the stated error. The Fiat–Shamir statements follow from the cited compilers.

(c) The simulator samples all challenges itself. It samples each opened value $\hat{w}(r)$ and $p(r)$ uniformly, which is justified by the randomized extensions (Section 6.3). It simulates the masked sumcheck transcript as in [11, 50], consistently with these values. It replaces all commitments by commitments to zero (P1) and simulates the evaluation argument with its honest-verifier simulator (P3). Hybrid arguments over (P1) and (P3) give computational indistinguishability. In the Fiat–Shamir form, the verifier sends no messages, and the simulator programs the random oracle at the transcript prefixes; zero-knowledge against arbitrary quantum verifiers then follows from the standard QROM programming argument for Fiat–Shamir [47]. □

Theorem 9 replaces the claims of *perfect* and *statistical* security in earlier versions. With computationally hiding (LWE- or hash-based) commitments, zero-knowledge is computational. Soundness relies on the binding of the commitment, so the protocol is an *argument* of knowledge.

6.7 Linking the Argument to the Execution

The argument constrains the execution only because its statement commitments are the ones through which inputs enter GF.

1. In Algorithm 3, before any invocation of GF, each party commits bitwise to u and v with PolyCommit. These commitments $\mathsf{cm}_u, \mathsf{cm}_v$ are the statement of Π_{QZKP}.
2. GF is run in commit-and-prove mode. Whenever a party enters an input bit $x = x_0 \oplus x_1$ as XOR shares, it proves in zero knowledge that $x_0 \oplus x_1$ equals the committed bit.
3. In the Byzantine setting, GF must be a maliciously secure 2PC. The protocol of Section 4.3.1 is described for honest-but-curious parties and must be compiled accordingly, for example with zero-knowledge against quantum verifiers [48, 49] over the same commitments.

In the proof of Theorem 4.2, the simulator obtains the corrupted party's effective inputs from the knowledge extractor of Theorem 9(b).

6.8 Complexity

The witness consists of $5\ell + 3$ vectors of length N. The prover performs $O(\ell N)$ field operations for the sumcheck, using the linear-time algorithm for multilinear sumcheck, plus the cost of committing to $O(\ell)$ polynomials of size N. The verifier performs $O(\mu+\ell)$ field operations plus one batched evaluation verification. Communication is $O(d\mu)$ field elements for the sumcheck, $O(\ell)$ opened values, and one batched evaluation proof, whose size depends on the instantiation. The soundness error is made at most $2^{-\lambda}$ by choosing $|\mathbb{F}| \geq 2^{\lambda+\log((d+2)\mu+m+3)}$, or by repetition over a smaller field.

7 Conclusion and Future Work

In 1997, Lo [30] showed the impossibility of unconditionally secure two-party quantum computation, when one of the parties is malicious. In this direction, we obtain a positive result for two types of functions, restricting the computational power of the quantum adversary. This does not contradict the generalized impossibility results of [4] in the broadcast channel model, since there they dealt with quantum adversaries with unconditional power, whereas our result deals with polynomial-time bounded adversary in non-simultaneous channel model.

We have shown that, assuming the existence of secure one-way functions against computationally bounded quantum adversaries, secure two-party quantum computation is achievable for all functions without an embedded XOR, and also for a specific function containing an embedded XOR. Note that, [20] also gave the classical solution for a specific function containing embedded XOR and our quantum solution covers the same function.

The main obstacle to extending this result to arbitrary embedded-XOR functionalities lies in the intrinsic information leakage through XOR dependencies. Unlike non-embedded-XOR functions, the output structure may allow one party to infer nontrivial correlation with the other party's inputs, making standard simulation techniques insufficient in the quantum setting. In particular, quantum adversaries may exploit entanglement and deferred measurements to preserve and combine partial information obtained across protocol rounds, preventing a straightforward extension of existing security proofs. Resolving this problem may therefore require new techniques for enforcing input privacy and simulatability under quantum correlations. This will possibly necessitate stronger commitment primitives, new rewinding-independent simulation methods, or protocols that explicitly neutralize XOR-dependent leakage.

Furthermore, extending these two-party protocols to the n-party setting remains an important direction for future work, especially in the non-simultaneous channel model.

References

1. Ames, S., Hazay, C., Ishai, Y., Venkitasubramaniam, M.: Ligero: Lightweight sublinear arguments without a trusted setup. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS) (2017)
2. Bartusek, J., Coladangelo, A., Khurana, D., Ma, F.: One-way functions imply secure computation in a quantum world. In: Advances in Cryptology – CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part I. p. 467–496. Springer-Verlag, Berlin, Heidelberg (2021). https://doi.org/10.1007/978-3-030-84242-0_17, https://doi.org/10.1007/978-3-030-84242-0_17
3. Baum, C., Damgård, I., Lyubashevsky, V., Oechsner, S., Peikert, C.: More efficient commitments from structured lattice assumptions. In: Security and Cryptography for Networks – SCN 2018 (2018)
4. Ben-Or, M., Crepeau, C., Gottesman, D., Hassidim, A., Smith, A.: Secure multiparty quantum computation with (only) a strict honest majority. In: 2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06). pp. 249–260 (2006). https://doi.org/10.1109/FOCS.2006.68
5. Ben-Sasson, E., Chiesa, A., Riabzev, M., Spooner, N., Virza, M., Ward, N.P.: Aurora: Transparent succinct arguments for r1cs. In: Advances in Cryptology – EUROCRYPT 2019 (2019)
6. Ben-Sasson, E., Chiesa, A., Spooner, N.: Interactive oracle proofs. In: Theory of Cryptography – TCC 2016-B (2016)
7. Beullens, W., Seiler, G.: LaBRADOR: Compact proofs for R1CS from Module-SIS. In: Advances in Cryptology – CRYPTO 2023 (2023)

8. Canetti, R.: Security and composition of multiparty cryptographic protocols. J. Cryptology **13**, 143–202 (2000). https://doi.org/10.1007/s001459910006
9. Chen, B., Bünz, B., Boneh, D., Zhang, Z.: Hyperplonk: Plonk with linear-time prover and high-degree custom gates. In: Advances in Cryptology – EUROCRYPT 2023 (2023)
10. Chen, X.B., Xu, G., Niu, X.X., Wen, Q.Y., Yang, Y.X.: An efficient protocol for the private comparison of equal information based on the triplet entangled state and single-particle measurement. Opt. Commun. **283**(7), 1561–1565 (2010)
11. Chiesa, A., Forbes, M.A., Spooner, N.: A zero knowledge sumcheck and its applications. Cryptology ePrint Archive, Paper 2017/305 (2017)
12. Chiesa, A., Manohar, P., Spooner, N.: Succinct arguments in the quantum random oracle model. In: Theory of Cryptography – TCC 2019 (2019)
13. Cramer, R., Dodis, Y., Fehr, S., Padró, C., Wichs, D.: Detection of algebraic manipulation with applications to robust secret sharing and fuzzy extractors. In: Smart, N. (ed.) Advances in Cryptology – EUROCRYPT 2008. pp. 471–488. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)
14. Crépeau, C.: Quantum oblivious transfer. Journal of Modern Optics **41**(12), 2445–2454 (1994). https://doi.org/10.1080/09500349414552291
15. Crépeau, C., Légaré, F., Salvail, L.: How to convert the flavor of a quantum bit commitment. In: Pfitzmann, B. (ed.) Advances in Cryptology — EUROCRYPT 2001. pp. 60–77. Springer Berlin Heidelberg, Berlin, Heidelberg (2001)
16. Don, J., Fehr, S., Majenz, C.: The measure-and-reprogram technique 2.0: Multi-round Fiat-Shamir and more. In: Advances in Cryptology – CRYPTO 2020 (2020)
17. Dumais, P., Mayers, D., Salvail, L.: Perfectly concealing quantum bit commitment from any quantum one-way permutation. In: Preneel, B. (ed.) Advances in Cryptology — EUROCRYPT 2000. pp. 300–315. Springer Berlin Heidelberg, Berlin, Heidelberg (2000)
18. Goldreich, O., Micali, S., Wigderson, A.: Proofs that yield nothing but their validity or all languages in np have zero-knowledge proof systems. J. ACM **38**(3), 690–728 (Jul 1991). https://doi.org/10.1145/116825.116852, https://doi.org/10.1145/116825.116852
19. Golovnev, A., Lee, J., Setty, S., Thaler, J., Wahby, R.S.: Brakedown: Linear-time and field-agnostic SNARKs for R1CS. In: Advances in Cryptology – CRYPTO 2023 (2023)
20. Gordon, D.S., Carmit, H., Katz, J., Lindell, Y.: Complete fairness in secure two-party computation. In: Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing. p. 413–422. STOC '08, Association for Computing Machinery, New York, NY, USA (2008)
21. Gordon, S.D., Hazay, C., Katz, J., Lindell, Y.: Complete fairness in secure two-party computation. J. ACM **58**(6) (Dec 2011)
22. Grilo, A.B., Lin, H., Song, F., Vaikuntanathan, V.: Oblivious transfer is in miniqcrypt. In: Advances in Cryptology – EUROCRYPT 2021: 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17–21, 2021, Proceedings, Part II. p. 531–561. Springer-Verlag, Berlin, Heidelberg (2021). https://doi.org/10.1007/978-3-030-77886-6_18, https://doi.org/10.1007/978-3-030-77886-6_18
23. HE, G.P.: Simple quantum protocols for the millionaire problem with a semi-honest third party. International Journal of Quantum Information **11**(02), 1350025 (2013). https://doi.org/10.1142/S0219749913500251
24. Impagliazzo, R., Rudich, S.: Limits on the provable consequences of one-way permutations. In: Goldwasser, S. (ed.) Advances in Cryptology — CRYPTO' 88. pp. 8–26. Springer New York, New York, NY (1990)
25. Jia, H.Y., Wen, Q.Y., Song, T.T., Gao, F.: Quantum protocol for millionaire problem. Opt. Commun. **284**(1), 545–549 (2011)
26. Khawasik, M., Elsayed, W., Rashad, M., Younes, A.: A secured quantum two-bit commitment protocol for communication systems. IEEE Access **10**, 50218–50226 (2022). https://doi.org/10.1109/ACCESS.2022.3173645
27. Khurana, D., Tomer, K.: Commitments from quantum one-wayness. In: Proceedings of the 56th Annual ACM Symposium on Theory of Computing. p. 968–978. STOC 2024, Association for Computing Machinery, New York, NY, USA (2024). https://doi.org/10.1145/3618260.3649654
28. Lindell, Y.: Composition of Secure Multi-Party Protocols, A Comprehensive Study, vol. 2815 (2003). https://doi.org/10.1007/b13246
29. Liu, W., Wang, Y.B., Jiang, Z.T.: An efficient protocol for the quantum private comparison of equality with w state. Optics Communications **284**(12), 3160–3163 (2011). https://doi.org/https://doi.org/10.1016/j.optcom.2011.02.017
30. Lo, H.K.: Insecurity of quantum secure computations. Phys. Rev. A **56**, 1154–1162 (1997). https://doi.org/10.1103/PhysRevA.56.1154
31. Lo, H.K., Chau, H.F.: Is quantum bit commitment really possible? Phys. Rev. Lett. **78**, 3410–3413 (1997). https://doi.org/10.1103/PhysRevLett.78.3410

32. Lund, C., Fortnow, L., Karloff, H., Nisan, N.: Algebraic methods for interactive proof systems. In: Proceedings of the 31st Annual Symposium on Foundations of Computer Science. pp. 2–10. IEEE Computer Society (1990). https://doi.org/10.1109/FSCS.1990.89518
33. Lyubashevsky, V.: Fiat-shamir with aborts: Applications to lattice and factoring-based signatures. In: Fischlin, M. (ed.) Advances in Cryptology – ASIACRYPT 2009. pp. 598–616. Springer Berlin Heidelberg, Berlin, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10366-7_35
34. Lyubashevsky, V.: Lattice signatures without trapdoors. In: Pointcheval, D., Johansson, T. (eds.) Advances in Cryptology – EUROCRYPT 2012. pp. 738–755. Springer Berlin Heidelberg, Berlin, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_43
35. Mayers, D.: Unconditionally secure quantum bit commitment is impossible. Phys. Rev. Lett. **78**, 3414–3417 (1997). https://doi.org/10.1103/PhysRevLett.78.3414
36. Nguyen, N.K., Seiler, G.: Greyhound: Fast polynomial commitments from lattices. In: Advances in Cryptology – CRYPTO 2024 (2024)
37. Nielsen, M.A., Chuang, I.L.: Quantum Computation and Quantum Information: 10th Anniversary Edition. Cambridge University Press, Cambridge (2010)
38. Rivest, R.L., Adleman, L., Dertouzos, M.L.: On data banks and privacy homomorphisms. Foundations of Secure Computation, Academia Press pp. 169–179 (1978)
39. Schwartz, J.T.: Fast probabilistic algorithms for verification of polynomial identities. Journal of the ACM **27**(4), 701–717 (1980). https://doi.org/10.1145/322217.322225
40. Setty, S.: Spartan: Efficient and general-purpose zksnarks without trusted setup. In: Advances in Cryptology – CRYPTO 2020 (2020)
41. Shamir, A.: How to share a secret. Commun. ACM **22**(11), 612–613 (Nov 1979). https://doi.org/10.1145/359168.359176
42. Shor, P.: Algorithms for quantum computation: discrete logarithms and factoring. In: Proceedings 35th Annual Symposium on Foundations of Computer Science. pp. 124–134 (1994). https://doi.org/10.1109/SFCS.1994.365700
43. Srikantaiah, J., Dasgupta, A.: Quantum mechanical effects in bulk MOSFETs from a compact modeling perspective: A review. IETE Technical Review **29**, 3–28 (2012)
44. Tseng, H.Y., Lin, H.C., Hwang, T.: New quantum private comparison protocol using epr pairs. Quantum Information Processing **11**, 373–384 (2012). https://doi.org/10.1007/s11128-011-0251-0
45. Unruh, D.: Universally composable quantum multi-party computation. In: Gilbert, H. (ed.) Advances in Cryptology – EUROCRYPT 2010. pp. 486–505. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
46. Unruh, D.: Computationally binding quantum commitments. In: Advances in Cryptology – EUROCRYPT 2016 (2016)
47. Unruh, D.: Post-quantum security of Fiat-Shamir. In: Advances in Cryptology – ASIACRYPT 2017 (2017)
48. Watrous, J.: Zero-knowledge against quantum attacks. p. 296–305. STOC '06, Association for Computing Machinery, New York, NY, USA (2006). https://doi.org/10.1145/1132516.1132560
49. Watrous, J.: Zero-knowledge against quantum attacks. SIAM Journal on Computing **39**(1) (2009)
50. Xie, T., Zhang, J., Zhang, Y., Papamanthou, C., Song, D.: Libra: Succinct zero-knowledge proofs with optimal prover computation. In: Advances in Cryptology – CRYPTO 2019 (2019)
51. Yang, Y.G., Wen, Q.Y.: An efficient two-party quantum private comparison protocol with decoy photons and two-photon entanglement. Journal of Physics A-mathematical and Theoretical - J PHYS A-MATH THEOR **42** (02 2009). https://doi.org/10.1088/1751-8113/42/5/055305
52. Yang, Y.G., Xia, J., Jia, X., Zhang, H.: Comment on quantum private comparison protocols with a semi-honest third party. Quant. Inf. Proc. **12**(2), 877–885 (2013). https://doi.org/10.1007/s11128-012-0433-4
53. Yao, A.C.C.: Protocols for secure computations. 23rd Annual Symposium on Foundations of Computer Science (sfcs 1982) pp. 160–164 (1982)
54. Yao, A.C.C.: How to generate and exchange secrets. In: 27th Annual Symposium on Foundations of Computer Science (sfcs 1986). pp. 162–167 (1986). https://doi.org/10.1109/SFCS.1986.25
55. Zhou, Y.H., Shi, W.M., Yang, Y.G.: A Quantum Protocol for Millionaire Problem with Continuous Variables. Communications in Theoretical Physics **61**(4), 452-456 (2014). https://doi.org/10.1088/0253-6102/61/4/08
56. Zippel, R.: Probabilistic algorithms for sparse polynomials. Symbolic and Algebraic Computation pp. 216–226 (1979). https://doi.org/10.1007/3-540-09519-5_73