Quantum Multi-Key Homomorphic Encryption for Polynomial-Sized Circuits

Rishab Goyal
`rgoyal@cs.utexas.edu`

Abstract

Fully homomorphic encryption (FHE) is a powerful notion of encryption which allows data to be encrypted in such a way that anyone can perform arbitrary computations over the encrypted data without decryption or knowledge of the secret key. Traditionally, FHE only allows for computations over data encrypted under a *single public key*. López-Alt et al. (STOC 2012) introduced a new notion of FHE, called multi-key FHE (MFHE), which permits joint computations over data encrypted under *multiple independently-generated* (unrelated) keys such that any evaluated ciphertext could be (jointly) decrypted by the parties involved in the computation. Such MFHE schemes could be readily used to delegate computation to cloud securely.

Recently a number of works have studied the problem of constructing *quantum* homomorphic encryption (QHE) which is to perform quantum computations over encrypted quantum data. In this work we initiate the study of quantum *multi-key* homomorphic encryption (QMHE) and obtain the following results:

1. We formally define the notion of quantum multi-key homomorphic encryption and construct such schemes from their classical counterpart. Building on the framework of Broadbent and Jeffery (Crypto 2015) and Dulek et al. (Crypto 2016), we show that *any* classical multi-key leveled homomorphic encryption can be used to build a quantum multi-key leveled homomorphic encryption if we also have certain suitable error-correcting quantum gadgets. The length of the evaluation key grows linearly with the number of T-gates in the quantum circuit, thereby giving us a quantum multi-key leveled homomorphic encryption for circuits with polynomial but bounded number of T-gates.
2. To enable a generic transformation from any classical multi-key scheme, we introduce and construct a new cryptographic primitive which we call conditional oblivious quantum transform (COQT). A COQT is a distributed non-interactive encoding scheme that captures the essence of error-correcting gadgets required for quantum homomorphic encryption in the multi-key setting. We then build COQTs themselves from any classical multi-key leveled homomorphic encryption with $\mathbf{NC}^1$ decryption. We believe that COQTs might be an object of independent interest.
3. We also show that our quantum multi-key homomorphic encryption schemes support distributed decryption of multi-key ciphertexts as well as allows ciphertext re-randomizability (thereby achieves quantum circuit privacy) if the underlying classical scheme also supports distributed decryption and satisfies classical circuit privacy. We show usefulness of distributed decryption and ciphertext re-randomizability for QMHE by providing efficient templates for building multi-party delegated/server-assisted quantum computation protocols from QMHE.

   Additionally, due to our generic transformation, our quantum multi-key HE scheme inherits various features of the underlying classical scheme such as: identity/attribute-based, multi-hop, etc.

1 Introduction

Fully homomorphic encryption (FHE) is a powerful notion of encryption which allows data to be encrypted in such a way that anyone can perform arbitrary computations over the encrypted data without decryption or knowledge of the secret key. With widescale adoption of cloud storage, it has become essential to encrypt sensitive data before moving it to the cloud. If one encrypts the data using FHE, then it not only solves the privacy problem but also provides the capability to delegate computation and perform numerous useful tasks without sacrificing confidentiality.

The problem of computing over encrypted data was introduced by Rivest, Adleman and Dertouzos [RAD78], however constructing a general purpose FHE scheme remained an open problem for more than three decades. In a breakthrough result in 2009, Gentry [Gen09] provided the first construction for FHE for general circuits.[1] Since then there has been a tremendous progress [Gen10, vDGHV10, BV11b, BV11a, GH11b, GH11a, CMNT11, CNT12, BGV12, GHS12a, GHS12b, LATV12, Bra12, GSW13] with constructions being simplified greatly and their security being based on hardness of the learning with errors (LWE) assumption [Reg05], a standard cryptographic assumption which is based on the worst-case hardness of certain lattice problems ($\mathsf{GapSVP}, \mathsf{SIVP}$).

In today's world, where distributed computing is widely prevalent, FHE is not always sufficient as it only allows for computations over data encrypted under a *single public key*. This shortcoming of FHE prevents its more wider adoption. Consider the following scenario with two users — Alice and Bob. They both have some private data and would like to perform joint computations over their combined datasets, however they would like to delegate this task as they do not have sufficient computational power to complete the task. Now we can solve this problem using FHE, but with the extra assumption of existence of a *fully trusted* third party/ cloud server, i.e. Alice and Bob can publish their data encrypted under server's public key, after which the server could perform the specified computations and relay the respective outputs. Clearly without a *trusted* third party this approach does not work. To this end, López-Alt et al. [LATV12] introduced a new notion of FHE which they called multi-key FHE (MFHE). In a few words, MFHE allows joint computations over data encrypted under multiple independently-generated (unrelated) keys such that any evaluated ciphertext could be (jointly) decrypted by the parties involved in the computation. Such MFHE schemes could be readily used to delegate computation to cloud securely. Since its introduction there has been a steady stream of work on FHE in the multi-key setting [CM15, MW16, BP16, PS16, DHRW16, CO17] with variety of improvements ranging from basing hardness on LWE to providing 1-round threshold decryption to achieving circuit privacy and more.

The problem of computing over encrypted data becomes even more interesting when one considers the existence of quantum computers. Now if (and when) quantum computing becomes practical, it would unlock several new facets of this problem. Broadly, those could be divided in the following two categories — (1) constructing (multi-key) homomorphic encryption schemes for encrypting *classical* information that are secure even under *quantum attacks*, and (2) constructing secure (multi-key) homomorphic encryption schemes that could be used to encrypt *quantum* information. Informally, one involves tightening security of classical primitives in presence of quantum adversaries and other explores providing new functionality in presence of quantum information. Now LWE is widely believed to be a *quantum-safe assumption* as it is conjectured that approximating GapSVP (or SIVP) within any polynomial factors (or even certain subexponential factors) is hard even for polynomial time quantum algorithms. Thus it is one of the most suitable candidates to base quantum-resistant cryptography. Since most current (M)FHE constructions rely on LWE for security, thus the first direction has already been explored. In this work, we make progress along the second direction. One of our main contributions is to construct multi-key homomorphic encryption schemes for encrypting and evaluating quantum data.[2]

Recently a number of works [Lia13, YPDF14, OTF15, BJ15, DSS16] have studied the problem of constructing quantum fully homomorphic encryption (QFHE). The idea of quantum homomorphic encryption

[1]Although some progress was made in [GM84, Elg84, Pai99, SYY99, BGN05, IP07], none of these solved the problem completely.

[2]We would like to point out that, due to our modular approach, we also make qualitative improvements in single-key setting and not just new constructions for multi-key schemes. Concretely, we show how to construct a quantum HE scheme from any classical HE scheme without assuming any special structure or additional properties like space/time-efficient decryption in the starting scheme. Instead we require only for bootstrapping our quantum *gadget* that there exists a classical HE scheme with a log-depth decryption. This allows our transformation to preserve all the properties of the underlying classical HE system like it being identity-based and/or attribute based [GSW13] etc. Although the same result in the single-key setting could be achieved in a *non-black box* way by exploiting the underlying structure of [DSS16], we achieve this generically.

first appeared in [Lia13] where they proposed a restricted symmetric-key variant satisfying limited homomorphism using quantum one-time pads [AMTdW00] without making any computational assumption (i.e., in the information-theoretic setting). Although the problem of quantum homomorphic encryption was informally introduced in [RFG12, Lia13], the formal definitions (both in the public-key and symmetric-key settings) first appeared in [BJ15]. An important contribution made by Broadbent and Jeffery [BJ15] was to present a security notion for quantum encryption schemes which they referred to as *quantum indistinguishability under chosen plaintext attacks* (q-IND-CPA), and is a natural extension of the standard IND-CPA (semantic) security to the quantum setting.

Although one cannot hope to construct information-theoretically secure classical FHE schemes, as that would break the lower bound on communication complexity of private information retrieval protocols [CGKS95, KO97, CKGS98], a natural question to ask is whether the impossibility could be bypassed in the quantum world. Unfortunately, Yu et al. [YPDF14] proved impossibility of information-theoretically secure QFHE by proving an exponential lower bound on the size of encrypted states in such a system. Informally, they showed that there cannot exist a perfectly secure QFHE scheme that can be used to evaluate arbitrarily large quantum circuits while also satisfying compactness.[3] These negative results, however, still leave a gap as in they do not immediately rule out existence of a *computationally*-secure QFHE, or a perfectly secure quantum *leveled* homomorphic encryption (QLHE).[4] Recent works by Ouyang et al. [OTF15], Broadbent and Jeffery [BJ15], and Dulek et al. [DSS16] make progress towards filling this gap with [OTF15] constructing information-theoretically secure QLHE for circuits with a constant number of non-Clifford operations, and [BJ15] and [DSS16] constructing computationally secure QLHE for circuits with constant T-gate depth and polynomially many (but a-priori bounded) T-gates, respectively.[5]

Since the impossibility result of Yu et al. [YPDF14] readily extends to the multi-key setting, therefore a natural question is whether it could be bypassed using computational assumptions as in the single-key setting. In this work, we answer this in the affirmative by presenting a quantum multi-key leveled homomorphic encryption scheme based on the LWE assumption. We start by formally defining quantum multi-key homomorphic encryption. As in prior works [BJ15, DSS16, Mah17][6] we consider security against chosen plaintext attacks (q-IND-CPA), and our definition is a generalization of QFHE definition to the multi-key setting. Towards constructing quantum multi-key homomorphic encryption, we match the current state-of-the-art results in the single-key setting [DSS16], that is we provide a construction of quantum multi-key homomorphic encryption for *poly-sized quantum circuits*. At a high level, our approach is similar to that used in prior works which is to use quantum one-time pads in conjunction with a classical homomorphic encryption scheme. A detailed technical overview is given in Section 2.

Applications and Related Work. The most natural application of homomorphic encryption is delegation of computation. As observed by prior works [BJ15, DSS16, Mah17], any quantum homomorphic encryption scheme can be used to construct *constant-round* protocols for *blind delegated quantum computation* [Chi05, AS06, BFK09, ABOE08].[7] Typically, in blind quantum computation we have two parties — Alice (client) and Bob (server). Here Alice does not have sufficient quantum technology whereas Bob has unlimited quantum resources. Alice wants Bob to perform some computation using his quantum resources such that Bob does not learn anything about Alice's input, output or, the function evaluated other than

[3]Roughly, a (Q)FHE scheme is compact if the size of evaluated ciphertexts is independent of size of circuit evaluated.

[4]Throughout this paper we say a (quantum) homomorphic encryption scheme is leveled if the class of (quantum) circuits that can be homomorphically evaluated is bounded/fixed during key generation.

[5]Broadbent and Jeffery [BJ15] also provide another construction that can be used to evaluate arbitrary quantum circuits, although it only satisfies a weaker compactness notion, namely *quasi-compactness*. However, in this work, we only focus on compact homomorphic encryption schemes.

[6]In [Mah17], the authors give constructions of quantum (single-key) homomorphic encryption that remove the a-priori bound on the number of T-gates, but are less general as they use more restricted primitives like indistinguishability obfuscation and classical (single-key) homomorphic encryption with certain special properties, and also lead to imperfect correctness. In this work, we focus on generic transformations from classical multi-key homomorphic encryption to quantum multi-key homomorphic encryption.

[7]As pointed out in [DSS16], QFHE schemes can be used to do delegated quantum computation in *two* or *three* rounds depending on the quantum resources available to the client as well as the underlying QFHE construction.

trivial to compute information. Although there has been a long line of work to construct protocols for blind delegated quantum computation [Chi05, AS06, BFK09, ABOE08, RFG12, BGS13, DFPR14, FBS+14, Bro15, TKO+16, ABOEM17], none of these protocols match efficiency (in terms of round complexity, communication complexity, universality of computation) of those obtained via QFHE. Therefore, this makes constructing QFHE an even more interesting problem.

A quantum multi-key homomorphic encryption scheme not only subsumes standard quantum (single-key) homomorphic encryption, but also all its potential applications like delegation and blind quantum computation. Additionally, it could also lead to more efficient solutions for *multi-party quantum computation* [CGS02, BOCG+06] (MPQC) as well as *multi-party* blind quantum computing [KP16]. In the classical setting, López-Alt et al. [LATV12] showed that a multi-key FHE could be used to do multi-party computation (MPC) [Yao82, Yao86, GMW87] by providing a simple and elegant template. In a follow-up work, Mukherjee and Wichs [MW16] extended the MFHE definition to include a 1-round threshold decryption algorithm which lead to an even simpler template and more efficient protocol for MPC. At a high level, the MPC protocol proceeds as follows. Each party starts by independently generating an MFHE key pair and encrypting its private input under its respective public key. In the first round of communication, each party broadcasts its encrypted input to all other parties. Next, each party (independently and) homomorphically computes the desired function over the encrypted inputs. Finally, they run the threshold decryption algorithm on the evaluated ciphertext and broadcast the partial decryption which could be used by any party to reconstruct the final output of computation.

The most attractive property of this template is that it is extremely suitable to achieve *round-efficient* MPC in the classical setting as shown in [MW16, BHP17]. The MPC protocols constructed in [LATV12, MW16, BHP17] are first proven to be secure against semi-malicious adversaries [AJW11, AJL+12], and later their security is generically boosted to full security (i.e., against malicious adversaries) using non-interactive zero knowledge (NIZK) arguments. An interesting question is whether such a template could be extended to the quantum setting. To this end, we make some partial progress and show how to extend the classical template to the quantum setting. Independently and concurrently, Brakerski [Bra18] presented a QFHE scheme with classical key generation building on the work of Mahadev [Mah17] and improving the underlying assumption. Another interesting question is whether similar ideas could be used in the multi-key setting as well.

Our Results. In this work we initiate the study of quantum multi-key homomorphic encryption (QMHE) and study the notion of threshold decryption of quantum multi-key ciphertexts and its potential applications. We start by formally defining the notion of quantum multi-key homomorphic encryption. We then build quantum multi-key leveled homomorphic encryption for circuits with polynomial (but bounded) number of T-gates from classical multi-key leveled homomorphic encryption, and certain error-correcting quantum gadgets which we call conditional oblivious quantum transform (COQT). We show that COQTs can be generically constructed from any classical multi-key leveled homomorphic encryption that support $\mathbf{NC}^1$ decryption.

Our QMHE achieves compactness in the sense that the size of ciphertexts is independent of the size of quantum circuits evaluated, however the size of the evaluation key grows linearly with the number of T-gates which can be homomorphically evaluated. We also define the notion of distributed (non-interactive) decryption for quantum multi-key ciphertexts. Note that (unlike classical multi-key ciphertexts) quantum multi-key ciphertexts can not be in possession of more than one party at any point of time due to unclonability of quantum states, thus there is not necessarily any single natural definition that one could consider in quantum setting. Our definition is inspired by its potential applications to multi-party quantum computation, delegation etc. We show that our QMHE scheme supports distributed (non-interactive) decryption as well as allows ciphertext re-randomizability (thereby achieves quantum circuit privacy) if the underlying classical scheme also supports distributed decryption and satisfies classical circuit privacy. We show that QMHE schemes that support threshold decryption can be directly used for multi-party delegated quantum computation [KP16], thereby answering the question (in affirmative) raised in [KP16] on whether quantum FHE could be adapted to multiple parties and if it could be used to multi-party delegated quantum computation.

2 Technical Overview

We start by briefly reviewing the quantum (single-key) homomorphic encryption schemes provided in [BJ15, DSS16]. Next, we discuss some natural ideas to extend those schemes to the multi-key setting as well as the technical difficulties faced in doing so. We then explain the new abstractions and framework we introduce to construct quantum multi-key homomorphic encryption as well as show how to instantiate our framework using only classical multi-key homomorphic encryption schemes. Finally, we describe the notion of threshold decryption for quantum multi-key ciphertexts and discuss its applications.

2.1 Prior Work

Broadbent and Jeffery [BJ15] start with a simpler goal of constructing QMHE schemes for evaluating only Clifford circuits. To that end, they describe a simple and elegant scheme called Clifford scheme (or CL). The main idea in the CL scheme is to encrypt the quantum message state using a quantum one-time pad, and encrypt the corresponding one-time pad keys using a classical homomorphic encryption scheme. Since all the unitaries in Clifford group commute with the Pauli matrices, thus if one directly evaluates a Clifford circuit over the encrypted quantum state, then (due to commutativity property) the circuit evaluation on encrypted cipherstate can be rewritten as a one-time pad encryption of the circuit evaluation on message-states instead. Also, the final one-time pad keys will be fixed functions of initial one-time pad keys and the circuit being evaluated. Thus, to homomorphically evaluate a quantum circuit, the evaluator simply proceeds as follows — first, it directly evaluates the quantum circuit on the encrypted cipherstates; next, it updates the one-time pad keys using classical homomorphic evaluation algorithm.

Now, it turns out that performing non-Clifford operations (homomorphically) on encrypted states in CL scheme, without losing correctness or security, is not a straightforward task. Concretely, if we evaluate a T-gate (i.e., a non-Clifford group gate) on a quantum one-time pad encrypted state $\mathsf{X}^a\mathsf{Z}^b\,|\psi\rangle$, then the resulting state is of the form $\mathsf{P}^a\mathsf{X}^a\mathsf{Z}^b\mathsf{T}\,|\psi\rangle$, i.e. the evaluated state might contain an unwanted (non-Pauli) error P^a. Thus, one either needs to leak the one-time pad key a sacrificing security, or lose correctness. Since the Clifford group is not a universal quantum gate-set[8], thus this implies that the CL scheme cannot be directly used to homomorphically evaluate arbitrary poly-sized quantum circuits.

In order to evaluate non-Clifford operations, prior works [BJ15, DSS16] follow a similar template which is to extend the basic CL scheme so to allow evaluation of a bounded number of T-gates. Very informally, the central idea is to provide auxiliary (classical-)quantum states as part of the evaluation key that can be used to *instantaneously* remove the non-Pauli errors after each T-gate evaluation.[9] Most recently, Dulek et al. [DSS16] proposed a new QHE scheme, named TP, in which one could evaluate polynomial (but a-priori bounded) number of T-gates.

At a very high level, their idea was to construct efficient *error-correcting gadgets* that do not break q-IND-CPA security, but could be used to remove such evaluation errors on-the-fly. Observe that, in the CL scheme, if the evaluator knows the underlying quantum one-time pad keys, which it could learn if it knows the classical HE secret key, then it could instantly correct the error in the quantum state after each T gate evaluation. More generally, it would be sufficient for the evaluator to have access to a *quantum channel* which takes an encrypted quantum state and encrypted one-time pad key as inputs and removes the P gate error (if any). Concretely, consider the following quantum channel which takes two inputs — an (encrypted) quantum state ρ and encryption of a classical bit a under HE public key pk. If $a = 0$, then it simply maps the output to state ρ, otherwise it maps the output to state $\mathsf{P}^\dagger\rho\mathsf{P}$, i.e. it simply maps the output to $(\mathsf{P}^a)^\dagger\rho\mathsf{P}^a$. There are a few important observations that we can make about such a channel — (1) it can be used to remove a P error introduced by quantum evaluation of a T gate, (2) it needs to contain information about classical HE secret key sk, (3) it can be used to learn the plaintext value m in any ciphertext $\mathsf{Enc}_{\mathsf{pk}}(m)$.

The first point suggests that any evaluator, given access to such a channel, can instantaneously correct non-Pauli errors introduced while evaluating a T gate. The second point suggests that it could only be

[8]However, by including any non-Clifford gate, we get a universal gate-set [NRS01]. As in prior works [BJ15, DSS16], we will use the T-gate, sometimes called $\pi/8$-gate, to extend the Clifford group to a universal set.

[9][BJ15] also gave a non-compact solution, but here we are only interested in *compact* solutions.

created by someone who knows the secret key sk. And, the last observation suggests it could be used to decrypt a quantum one-time pad given the encryptions of the corresponding one-time pad keys. Therefore, such a channel will not be useful since one cannot hope to hide the secret key sk in any reasonable way. However, if such a channel also re-randomized the state ρ *after* removing the P error by say re-encrypting using a fresh quantum one-time pad, then at least intuitively it seems to prevent the obvious attack. At a high level, this is the approach taken in [DSS16] where they show how to efficiently construct quantum gadgets which can simulate the behaviour of such a quantum channel. It turns out that in their scheme, one gadget is required per T-gate and each gadget is "consumed" during error correction process, therefore the number of gadgets in the evaluation key bounds the number of T-gates that can be homomorphically evaluated.

More formally, in TP scheme, an evaluator who knows (classical) HE encryption of the one-time pad keys (a, b) can efficiently transform a state $\mathsf{P}^a\mathsf{X}^a\mathsf{Z}^b\mathsf{T}\,|\psi\rangle$ to $\mathsf{X}^{a'}\mathsf{Z}^{b'}\mathsf{T}\,|\psi\rangle$, where (a', b') depend on keys (a, b) and other auxiliary information included as part of the evaluation key. Additionally, the evaluator can update the one-time pad keys from (a, b) to (a', b') using only classical homomorphic evaluations and the auxiliary information. Thus, TP scheme is identical to the Clifford scheme CL, except (1) during setup, polynomially many gadgets (for T-gate evaluation) are constructed and stored them as part of the evaluation key, (2) during evaluation of a T gate, the evaluator first applies the T gate directly over the encrypted cipherstate and then uses a gadget from the evaluation key to remove any P error and finally updates the one-time pad keys homomorphically.

Now to construct such quantum gadgets, they rely on quantum teleportation and recent results in the area of instantaneous non-local quantum computation [Spe16]. In instantaneous non-local computation, there are two parties who want to jointly perform a quantum operation, using only some pre-shared entanglement and a single round of simultaneous communication. At a high level, the non-Pauli error correction could be visualized as an application of instantaneous non-local computation, where one party (key generator) holds classical HE secret key, while the other party (evaluator) holds the encrypted cipherstate and HE encryption of one-time pad keys. Since both the parties jointly have all the information to determine whether a P error was introduced or not, thus the joint quantum operation they want to perform is applying an inverse phase gate $\mathsf{P}^\dagger$ on the cipherstate. Dulek et al. show using Speelman's results [Spe16] that the key generator can pre-compute all its operations with only a bounded amount of entanglement, thereby removing the need to interact with the evaluator completely. An important caveat in their gadget construction is that the underlying classical FHE scheme must have a log-space computable decryption procedure. This is because, for applying Speelman's results, it is necessary for the FHE decryption procedure to have low garden-hose complexity [BFSS13].

2.2 This Work

We will now describe our quantum multi-key homomorphic encryption scheme for poly-sized quantum circuits. Our starting point is the CL scheme. It turns out that CL already is a quantum *multi-key* homomorphic encryption scheme (for Clifford circuits) if the underlying classical scheme used is multi-key as well. That is, if the quantum one-time pad keys are encrypted using a classical multi-key scheme, then the resulting scheme will satisfy quantum multi-key homomorphism and can be used to evaluate arbitrary Clifford circuits. Now as in the single-key setting, multi-key version of CL scheme cannot be directly used to evaluate non-Clifford gates. Thus, to build quantum multi-key scheme for poly-sized quantum circuits, we need a mechanism to perform non-Clifford operations.

At this point, a natural question to ask is whether the TP scheme can be made to be a multi-key scheme similarly by simply using a multi-key classical scheme? Unfortunately, this is not the case. Recall that TP scheme crucially relies on their quantum gadgets to perform error correction, where a gadget basically takes a quantum cipherstate and one-time pad keys encrypted under a *single* public key pk. Now the main reason why TP does not directly extend to the multi-key setting is because of these gadgets. To understand this in more detail, consider encryptions of qubits $|\psi_1\rangle$ and $|\psi_2\rangle$ under keys pk_1 and pk_2 (respectively), i.e. let $\sigma_i = \mathsf{X}^{a_i}\mathsf{Z}^{b_i}\,|\psi_i\rangle$ and $\mathsf{ct}_i = \mathsf{Enc}_{\mathsf{pk}_i}(a_i, b_i)$ for $i = 1, 2$. Note that encryption of $|\psi_i\rangle$ consists of both σ_i and ct_i. Suppose we want to homomorphically evaluate a CNOT gate on these two qubits. The evaluator starts by directly applying a CNOT on σ_1 and σ_2. Next, using multi-key homomorphism it could update the one-time pad keys. Now note that the updated one-time pad keys will now be encrypted under both pk_1 and pk_2. Let σ_i' and ct_i' denote the evaluated cipherstate and ciphertexts, respectively. Now suppose we also want to evaluate a T-gate on the first qubit. Following TP, the evaluator will apply T directly on σ_1' which could introduce a non-Pauli error. Now to remove the error one would hope to use a quantum gadget as before, however that does not directly work as the one-time pad keys are encrypted under *two independently-generated* keys instead of one. Thus, in order to remove such errors, we need better "distributed" gadgets where each gadget is associated with only a single-key, but could be combined with gadget(s) associated with other key(s) such that a combined gadget could be used to remove errors where the one-time pad keys are encrypted under multiple keys. Another very important aspect that is required from these gadgets which makes them hard to construct is that the gadget generation should be completely *independent* of other keys, as otherwise it will not be useful. Now such gadgets are not known to be implied by Speelman's results [Spe16] since here the (instantaneous) quantum computation could be between more than two parties[10] with an additional constraint that it should be possible for all parties except one to pre-compute all their computation and that too without any interaction. Looking ahead, it turns out that the second construction of single-key quantum gadgets by Dulek et al. (used for building quantum HE from classical HE with $\mathbf{NC}^1$ decryption) can be extended for building such multi-key gadgets. Due to the distributed and non-interactive generation process of these gadgets as described above, the single-key to multi-key gadget extension has to be done in a non-black box way which we discuss later.

Our Framework. We start by introducing and constructing a new cryptographic primitive which we call conditional oblivious quantum transform (COQT). A COQT is a distributed non-interactive encoding scheme that captures the essence of gadgets that we require. It consists of three poly-time algorithms — encode, apply, and decode that work as follows. Consider a classical circuit C that takes as input $N+1$ strings of possibly unequal lengths. The encoding algorithm takes as input a bit string x, classical circuit C and an index $i \leq N$, and outputs an encoded quantum state σ along with a classical decoding key key. The apply algorithm takes as input a sequence of N encoded states $\{\sigma_i\}_{i \leq N}$, another input string y and a single-qubit state ρ, and outputs a transformed quantum state ρ' along with some classical auxiliary information aux. And, the decoding algorithm takes as input N decoding keys $\{\mathsf{key}_i\}_{i \leq N}$ and auxiliary information aux, and outputs two bits (a, b).

Informally, a COQT encoding procedure can be used to encode N bit-strings $x_1, \ldots, x_N$ *independently* with respect to a classical circuit C into N encoded quantum states $\{\sigma_i\}_{i \leq N}$ and classical decoding keys $\{\mathsf{key}_i\}_{i \leq N}$ such that the encodings $\sigma_1, \ldots, \sigma_N$ could be used in combination to *conditionally* apply a quantum gate G on a single-qubit state ρ, where the condition/predicate is $C(x_1, \ldots, x_N, x_{N+1})$ and x_{N+1} is any arbitrary bit-string of appropriate length chosen/known only at the time of applying the transform. Now for correctness we only require that applying the COQT on state ρ transforms it into state ρ' such that

$$\rho' = (\mathsf{X}^a \mathsf{Z}^b \mathsf{G}^{C(x_1, \ldots, x_{N+1})}) \rho (\mathsf{X}^a \mathsf{Z}^b \mathsf{G}^{C(x_1, \ldots, x_{N+1})})^\dagger,$$

where (a, b) is the decoding algorithm's output. For security, we require that an encoded state σ_i should not reveal any information about the encoded input/bit-string x_i if one does not know the decoding key key_i associated with it.

Now it turns out that if we have a secure COQT for applying the $\mathsf{P}^\dagger$-gate, then that is sufficient to build a quantum multi-key homomorphic encryption scheme from its classical counterpart. For an easier exposition, let us start with a simpler goal of building a relaxed notion of QMHE which we call quantum multi-key *positional* homomorphic encryption (QMPHE). A QMPHE scheme is just like the standard non-positional scheme, except the key generation algorithm now also takes as input the number of parties N and a position $i \leq N$. This puts two restrictions — (1) number of users/parties need to be known and fixed during key generation, and (2) ciphertexts computed using public keys of any two users/parties, generated for the same

[10]In the multi-key setting, we will have $N+1$ parties where one party (evaluator) holds the encrypted cipherstate (encrypted under N public keys) and other N parties (key generators), each hold a classical HE secret key. And, they jointly want to remove a non-Pauli error (if any).

position i, cannot be combined/evaluated together. Looking ahead, taking position as an additional input during key generation helps in generating appropriate COQTs. Both COQTs and QMPHE are formally introduced later in Section 4.

Building QMPHE using COQTs. As mentioned before, our QMPHE scheme is an extension of the multi-key CL scheme in which during setup, the key generation algorithm also generates some quantum gadgets in the form of COQTs which will be used later while homomorphically evaluating T gates. At a high level, the scheme works as follows. The key generation algorithm takes as input the number of users N, position i, and an upper bound on the number of T-gate evaluations supported, say k. Now it samples k COQTs along with a MFHE key pair $(\mathsf{pk}, \mathsf{ek}, \mathsf{sk})$. Each COQT is an encoding of key sk for MFHE decryption circuit, index i and gate $\mathsf{P}^\dagger$. Let $(\sigma_j, \mathsf{key}_j)$ be the corresponding quantum encodings and classical decoding keys for $j \leq k$. Next, it computes ct_j as encryption of key key_j under public key pk. Finally, it sets sk and pk as the secret and public keys, and the quantum evaluation key will consist of the classical key ek along with all $(\sigma_j, \mathsf{ct}_j)_{j \leq k}$ pairs. At a high level, the idea is that an evaluator will pick one COQT from evaluation key of each party and together apply those on the encrypted cipherstate to remove P errors after a T-gate evaluation.

Concretely, to evaluate a T-gate, the evaluator first applies the T-gate on the corresponding (encrypted) state followed by a COQT application. To apply the COQT on the encrypted state, it first selects an unused encoding in i^{th} user's (quantum) evaluation key as the encodings for i^{th} position and sets the ciphertext encrypting the error indicator bit[11] as the input string. By correctness of COQT, this removes the P error (if any). Let aux be the auxiliary information generated during COQT application. Next, it generates the one-time pad key updates by homomorphically running the COQT decoding algorithm (with aux hardwired) on the ciphertexts encrypting appropriate decoding keys.

Now in order to prove security, the above construction has to be slightly modified similar to the TP scheme [DSS16]. The full scheme along with a more detailed outline is provided in Section 5. Also, we would like to point out that to apply the above transformation we only need COQTs for circuit class which contains the classical decryption circuit. In other words, if we have COQTs that can encode the classical decryption circuit and be used for applying $\mathsf{P}^\dagger$-gate, then we could construct a quantum multi-key positional homomorphic encryption scheme from *any* classical multi-key homomorphic encryption scheme. We believe this to be a very useful feature of our abstraction and framework.

Constructing COQTs for poly-sized circuits. We construct conditional oblivious quantum transform schemes in two steps. First, we build COQTs for $\mathbf{NC}^1$ circuits and gate set $\{\mathsf{P}^\dagger\}$, and prove it to be unconditionally secure. Next, we show that any COQT for class of log-depth circuits can be securely bootstrapped to all poly-size circuits using any classical multi-key HE scheme with log-depth decryption circuit. Since most existing classical multi-key schemes already have low-depth decryption circuits, they could be used to build COQTs for all poly-sized classical circuits. Below we give a brief overview. The starting point of our COQT construction for log-depth circuits is the quantum gadget construction of Dulek et al. [DSS16]. Roughly, our COQTs for $N = 1$ (i.e., in the single-key setting) are principally the same as the Dulek et al. error-correcting gadgets. In the main body we highlight the similarities in greater detail.

COQTs for $\mathbf{NC}^1$. At a high level, the scheme for log-depth circuits works as follows. To encode an input x for some index pos with circuit C and gate G, the encoder provides a partial encoding of the branching program (corresponding to the circuit C) in the form of a set of entangled pairs of qubits with gate G applied on one special qubit.[12] This encoding could be visualized as an alternate branching program representation of circuit $C(\ldots, x, \ldots)$, i.e. circuit C with input x hardwired at pos^{th} input position. Now, the information about the pairs that are entangled and the qubit that has G applied to it, is stored as part of the decoding key. During application, these partial (branching program) encodings, for each index $i \leq N$, are first re-arranged

[11]By error indicator bit we mean bit a which denotes whether the encrypted state has X-gate applied or not. Note that P error gets introduced only if X was present.

[12]For proving security it is necessary to encrypt one qubit in each pair of entangled qubits using a quantum one-time pad. The corresponding one-time pad keys will be stored as part of the decoding key.

to get a single conjoined encoding of a branching program corresponding to circuit $C(x_1, \ldots, x_N, \cdot)$, i.e. C with inputs $x_1, \ldots, x_N$ hardwired. Next, the input qubit ρ is teleported through the branching program, such that G is applied on ρ if only if $C(x_1, \ldots, x_N, y) = 1$ where y is chosen by the evaluator. These sequential teleportations require many Bell measurements, and the output of each measurement is stored as part of auxiliary information. Finally, the decoding algorithm simply traces the path of input qubit ρ after each teleportation and using the decoding keys as well as measurement outcomes, it computes final Pauli coefficients. From Barrington's theorem [Bar86] we know that log-depth circuits have poly-size branching program, thus we get that the size of each encoding is polynomially bounded as required.

Bootstrapping to $\mathbf{P}/\mathsf{poly}$. The idea behind bootstrapping to poly-sized circuits is to combine COQTs with multi-key HE schemes in a way that circuit/predicate computation is done using homomorphic evaluations and the quantum transform on input qubit is later done using only $\mathbf{NC}^1$ COQTs. Concretely, the encoder first samples a key pair $(\mathsf{pk}, \mathsf{sk})$ for a multi-key HE system. Next, it computes ciphertext ct as encryption of input x (being encoded) under key pk, and also creates a COQT (σ, key) for multi-key HE decryption circuit with secret key sk as the input bit string. Finally, it sets σ and ct as the final encoded state, and key as the decoding key. During evaluation, one could simply evaluate the circuit homomorphically on the encrypted inputs and later apply the transform on the input qubit ρ with the ciphertext encrypting the circuit output as the input. The decoding procedure will be identical to that of the underlying scheme.

Later in Sections 8 and 9 we provide a more detailed overview as well as our COQT constructions for $\mathbf{NC}^1$ and $\mathbf{P}/\mathsf{poly}$, respectively.

Removing *Positional* Constraint, Threshold Decryption and More. Recall that the quantum multi-key homomorphic encryption scheme we describe above is restricted in the sense that the key generation algorithm takes as input a position i. Later in Section 7, we show how to construct a standard (non-positional) quantum multi-key homomorphic encryption scheme for bounded number of users. In other words, the key generation algorithm will no longer take as input a position, but it will only take as input an upper bound on the number of users. At a very high level, the idea is to add redundancy in the system by providing many more COQTs such that each evaluation key contains all the information to be used as a positional evaluation key for any position $\leq N$. Additionally, we discuss various other improvements like achieving multi-hop evaluation, etc.

Threshold Decryption and Ciphertext Re-Randomizability. First, we would like to point out that we have to be careful in defining the notion of distributed decryption in the quantum setting as there is not necessarily any single natural definition that one could consider directly. Briefly, the issue is that unlike classical multi-key ciphertexts, quantum multi-key ciphertexts can not be in possession of more than one party at any point of time due to unclonability of quantum states. Thus, while defining the notion of threshold decryption, we need to consider that how our notion of threshold decryption could possibly be useful for applications such multi-party computation, delegation etc. With this observation, we visualize threshold decryption for QMHE as follows. We consider that a quantum cipherstate σ can be divided into two components — purely classical and purely quantum, i.e. say $\sigma = \rho(\mathsf{ct}) \otimes \widetilde{\sigma}$ (where ct denotes the purely classical component and $\widetilde{\sigma}$ denotes the quantum component). Now for threshold decryption, we consider that the classical component ct is distributed to all parties (which hold the corresponding secret keys), and each party can compute a partial decryption of ct using their secret key. Our goal here is that given all these partial decryptions of ct, anyone who possesses the quantum component $\widetilde{\sigma}$ can recover the underlying message state efficiently. At first, it might seem that there could be many trivial ways to build QMHE with threshold decryption (as described above), however we require the scheme to satisfy certain useful simulation-based security properties additionally and that makes it hard to construct and more useful for further applications. In the main body, we discuss the above stated aspects in more detail, and provide our definitions of threshold decryption and ciphertext re-randomizability for quantum multi-key homomorphic encryption schemes in Section 3.3, and later provide our threshold decryption algorithm and modifications required for ciphertext re-randomizability in Section 6.

Next, we briefly describe a protocol for multi-party delegated quantum computation using threshold

QMHE (as defined above) as the underlying primitive.[13] Let us start with the most natural idea which is to first ask each delegating party to encrypt its input using QMHE, and then sending the encrypted input (as well as the evaluation keys) over to the powerful server. The server then homomorphically evaluates the desired quantum circuit on the encrypted quantum input states, and broadcasts the evaluated cipherstates. Finally, all the delegating parties run a distributed decryption protocol on the (evaluated) encrypted quantum output states. However, this doesn't work because of the well known "no-cloning" folklore. Note that unlike classical ciphertexts, ciphertexts encrypting arbitrary quantum information can not be copied thus they can not be broadcast to all parties. To this end, we resort to the observation made above that every encrypted cipherstate in our QMHE scheme into a purely classical and a quantum component, where we could perform distributed decryption on only the classical component of the cipherstate, and reconstruct a decoding key from these partial decryptions which could eventually be used to decrypt the quantum component of the cipherstate thereby giving the final quantum output state. At a high level, the template we finally follow can be informally described as follows:

Setup and Input Encryption Phase. In the first round, each party independently samples a QMHE key pair and encrypts their input states under their respective public keys. Then they send their public key, (quantum) evaluation key and cipherstate to the server. (Here each party broadcasts the classical components of their keys and cipherstates to all other parties, but sends the quantum states only to the server.)

Evaluation Phase. In the second round, the server receives public keys, (quantum) evaluation keys and input cipherstates from all the N parties. It homomorphically evaluates the quantum circuit on these input cipherstates, and sends the output cipherstates to the respective parties. (As before, classical components of the cipherstates are broadcast.)

Distributed Decryption Phase. Finally, each party computes partial decryption of all the classical components of the output cipherstates. It only broadcasts those partial decryption values that do not correspond to its own output cipherstate.

Offline Output Phase. Each party first reconstructs the decoding key using the partial decryptions and eventually decrypts the quantum component of its cipherstate using the decoding key.

Now one might expect the above protocol to achieve security notions such as quantum analogue of semi-malicious security [AJW11, AJL$^+$12], however we do not know that to be the case. Briefly, this is due to the fact that weaker classical notions of security do not seem to translate well to the quantum setting. We elaborate more on such issues as well as formalize the above template and describe the protocol in detail later in Section 10.

3 Preliminaries

Notations. Let PPT denote probabilistic polynomial-time. We will use bold letters for vectors (e.g. $\mathbf{v}, \mathbf{T}$). For any positive integer n, we denote the set of all positive integers upto n as $[n] := \{1, \ldots, n\}$. For any finite set S, $x \leftarrow S$ denotes a uniformly random element x from the set S. Similarly, for any distribution $\mathcal{D}$, $x \leftarrow \mathcal{D}$ denotes an element x drawn from distribution $\mathcal{D}$. For any set $U = \{1, \ldots, n\}$, we say that $\{S_1, \ldots, S_\ell\}$ are ℓ set intervals of U if for all i, S_i are non-empty and there exists integers $m_1, m_2, \ldots, m_{\ell-1} \in U$ such that $m_1 \leq m_2 \leq \ldots \leq m_{\ell-1}$ and $S_i = \{m_{i-1}, \ldots, m_i\}$ for all i, where $m_0 = 1, m_\ell = n$. Also, we define $|\Phi^+\rangle := \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ to denote an EPR pair.

[13] We note that the same template also seems be useful for building protocols for "on-the-fly" multi-party quantum computation (i.e., server-assisted MPQC). Here on-the-fly MPQC can be defined as an extension of its classical counterpart [LATV12].

3.1 Quantum Computation: Gates, Circuits and One-time Pad

We work with the following set of unitary gates:

$$\mathsf{X} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad \mathsf{Z} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \quad \mathsf{P} = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix},$$

$$\mathsf{H} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad \mathsf{T} = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}, \quad \mathsf{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

The single-qubit Pauli matrices are $\mathsf{X}, \mathsf{Y}, \mathsf{Z}$ and I, where $\mathsf{Y} = i\mathsf{X}\mathsf{Z}$ and I is the identity matrix. An n-qubit Pauli operator is the tensor product of n single-qubit Pauli matrices. Note that the single-qubit Pauli group can be generated by X, Z up to a global phase. Throughout this paper, we ignore the global phase since it is not observable by any measurement.

The Clifford group is defined as the normalizer of the Pauli group, i.e. a Clifford group on n qubits consists of all unitaries that commute with the Pauli group. The set $\{\mathsf{H}, \mathsf{P}, \mathsf{CNOT}\}$ generates the Clifford group up to a global phase when applied to arbitrary qubits [Got98]. Also, all Pauli operators are part of the Clifford group. Now, the Clifford group does not give a universal gate-set, however by including any non-Clifford gate, we get a universal gate-set [NRS01]. As in prior works [BJ15, DSS16], we will use the T-gate, sometimes called $\pi/8$-gate, to extend the Clifford group to a universal set.

A density operator on a (complex Euclidean) space $\mathcal{X}$ is a positive semidefinite, Hermitian operator of trace 1 acting on $\mathcal{X}$. We denote the set of density operators on any space $\mathcal{X}$ by $D(\mathcal{X})$. For a random variable X defined over the possible basis states B for a quantum system, we use $\rho(X)$ to denote the density matrix corresponding to X, i.e. $\rho(X) := \sum_{x \in B} \Pr[X = x] \, |x\rangle\langle x|$. The density matrix of a completely mixed state on any system X of dimension d is simply the matrix $\mathbb{I}_d/d$. Also, throughout the paper we represent a classical-quantum state (over two systems $\mathcal{M}$ and $\mathcal{A}$) jointly as $\rho^{\mathcal{M}\mathcal{A}} := \sum_x \Pr[X = x] \, |x\rangle\langle x|^{\mathcal{M}} \otimes \rho_x^{\mathcal{A}}$.

A quantum channel Φ from space $\mathcal{X}$ to $\mathcal{Y}$ refers to any physically-realizable mapping on quantum registers. Let two quantum channels Φ_1 and Φ_2 on spaces $\mathcal{X}_1$ and $\mathcal{X}_2$, then $\Phi_1 \otimes \Phi_2$ denotes the quantum channel acting on joint registers in $D(\mathcal{X}_1 \otimes \mathcal{X}_2)$ with Φ_i acting on $\mathcal{X}_i$ register. We denote the identity channel on space $\mathcal{E}$ as $\mathbb{I}^{\mathcal{E}}$. For simplicity of notation, we drop the superscript $\mathcal{E}$, and also omit writing the identity channel explicitly (in a multi-register setting) whenever clear from context.

Now the trace norm of a state ρ is denoted by $\|\rho\|_1$, and is defined as $\|\rho\|_1 := \mathsf{Tr}\left(\sqrt{\rho^\dagger \rho}\right)$. Also, let Φ_C denote the channel induced by some circuit C, then the diamond norm on the channel Φ_C is defined as $\|\Phi_C\|_\diamond := \max_\rho \|(\Phi_C \otimes \mathbb{I})\rho\|_1$. That is, diamond norm represents the maximzation of a quantum channel over all input states ρ.

Quantum one-time pads [AMTdW00] provide a way to encrypt qubits in a perfectly secure manner. To encrypt, one simply chooses a Pauli operator uniformly at random and applies it on the message state. For instance, let ρ be a single-qubit system, and a, b be two classical bits. Quantum one-time pad encryption of ρ with keys a, b results in the cipherstate $\mathsf{X}^a\mathsf{Z}^b\rho\mathsf{Z}^b\mathsf{X}^a$. To decrypt, it suffices to apply the same Pauli operator $(\mathsf{X}^a\mathsf{Z}^b)$ on the cipherstate. The correctness of the scheme follows directly. The important property of a quantum one-time pad is that if a, b are chosen uniformly at random, then encryption results in a completely mixed state. Concretely, we know that for all ρ,

$$\sum_{a,b} \left(\frac{1}{4} \mathsf{X}^a\mathsf{Z}^b\rho\mathsf{Z}^b\mathsf{X}^a \right) = \frac{\mathbb{I}_2}{2}.$$

Therefore, quantum one-time pad provides perfect secrecy even against computationally unbounded quantum adversaries.

3.2 Multi-Key Homomorphic Encryption

A multi-key leveled homomorphic encryption (MLHE) scheme MLHE with message space $\mathcal{M}$ consists of six poly-time algorithms $\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Expand}, \mathsf{Eval}, \mathsf{Dec}$ with the following syntax:

- $\mathsf{Setup}(1^\lambda, 1^d) \to \mathsf{params}$. The setup algorithm takes as input the security parameter λ and the circuit depth bound d and outputs the system parameters params.
- $\mathsf{KeyGen}(\mathsf{params}) \to (\mathsf{pk}, \mathsf{ek}, \mathsf{sk})$. The key generation algorithm takes as input the system parameters params and outputs a public key pk, an evaluation key ek and a secret key sk.
- $\mathsf{Enc}(\mathsf{pk}, m) \to \mathsf{ct}$. The encryption algorithm takes as input a public key pk, message $m \in \mathcal{M}$, and outputs a ciphertext ct.
- $\mathsf{Expand}((\mathsf{pk}_1, \ldots, \mathsf{pk}_N), i, \mathsf{ct}) \to \widehat{\mathsf{ct}}$. The expansion algorithm takes as input a sequence of N public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_N$, a ciphertext ct encrypted under the i^{th} public key, and it outputs an *expanded* ciphertext $\widehat{\mathsf{ct}}$.
- $\mathsf{Eval}(C, (\mathsf{ek}_1, \ldots, \mathsf{ek}_{\ell'}), (\widehat{\mathsf{ct}}_1, \ldots, \widehat{\mathsf{ct}}_\ell)) \to \widehat{\mathsf{ct}}'$. The evaluation algorithm takes as input description of a circuit C along with two tuples comprising ℓ' evaluation keys ek_i and ℓ ciphertexts $\widehat{\mathsf{ct}}_i$. It outputs an *evaluated* ciphertext $\widehat{\mathsf{ct}}'$.
- $\mathsf{Dec}(\mathsf{sk}_1, \ldots, \mathsf{sk}_N, \widehat{\mathsf{ct}}) \to m$. The decryption algorithm takes as input a sequence of N secret keys $\mathsf{sk}_1, \ldots, \mathsf{sk}_N$, an expanded ciphertext $\widehat{\mathsf{ct}}$, and outputs a message m.

We now define correctness, compactness and security properties for MLHE schemes. First, we have the correctness and compactness properties. Informally, correctness states that decryption of an *expanded* ciphertext must output the underlying message as well as decryption of a homomorphically evaluated ciphertext must output the circuit evaluation on the underlying messages. And, the compactness property says that size of any *evaluated* ciphertext must not depend on the circuit being evaluated.

Correctness and Compactness. For any security parameter λ, circuit-depth bound d, consider parameters $\mathsf{params} \leftarrow \mathsf{Setup}(1^\lambda, 1^d)$, and any sequence of N key tuples $(\mathsf{pk}_i, \mathsf{ek}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{params})$ (for $i \leq N$). For any ℓ tuples of message-index pairs $(m_1, j_1), \ldots, (m_\ell, j_\ell)$, consider ciphertexts $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}_{j_i}, m_i)$ (i.e. encryptions of i^{th} message under its corresponding public key), and corresponding expanded ciphertexts $\widehat{\mathsf{ct}}_i \leftarrow \mathsf{Expand}((\mathsf{pk}_1, \ldots, \mathsf{pk}_N), j_i, \mathsf{ct}_i)$ (for $i \leq \ell$). The scheme MLHE is said to be (perfectly) correct if for any circuit C of depth $\leq d$, and evaluated ciphertext $\widehat{\mathsf{ct}} \leftarrow \mathsf{Eval}(C, (\mathsf{ek}_1, \ldots, \mathsf{ek}_N), (\widehat{\mathsf{ct}}_1, \ldots, \widehat{\mathsf{ct}}_\ell))$, the following holds:

- **Expansion.** $\forall i \leq \ell$**,** $\mathsf{Dec}(\mathsf{sk}_1, \ldots, \mathsf{sk}_N, \widehat{\mathsf{ct}}_i) = m_i$.
- **Evaluation.** $\mathsf{Dec}(\mathsf{sk}_1, \ldots, \mathsf{sk}_N, \widehat{\mathsf{ct}}) = C(m_1, \ldots, m_\ell)$.

Also, it is also said to be compact if there exists a polynomial $p(\cdot,\cdot,\cdot)$ such that $|\widehat{\mathsf{ct}}| \leq p(\lambda, d, N)$, i.e. size of $\widehat{\mathsf{ct}}$ does not depend on circuit C or the number of inputs ℓ.

Security. For security, we require the scheme to satisfy q-IND-CPA security, which is identical to IND-CPA except the adversary could now be a quantum algorithm.

Definition 3.1 (q-IND-CPA)**.** A multi-key leveled homomorphic encryption scheme MLHE = (Setup, KeyGen, Enc, Expand, Eval, Dec) is IND-CPA secure if for every *stateful quantum* PT adversary $\mathcal{A}$, there exists a negligible function $\mathrm{negl}(\cdot)$, such that for all $\lambda \in \mathbb{N}$, the following holds:

$$\left| \Pr\left[\mathcal{A}(\mathsf{ct}_b) = b : \begin{array}{c} \mathsf{params} \leftarrow \mathsf{Setup}(1^\lambda, 1^d);\ (\mathsf{pk}, \mathsf{ek}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{params}) \\ m \leftarrow \mathcal{A}(\mathsf{params}, \mathsf{pk}, \mathsf{ek});\ b \leftarrow \{0,1\} \\ \mathsf{ct}_0 \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathbf{0});\ \mathsf{ct}_1 \leftarrow \mathsf{Enc}(\mathsf{pk}, m) \end{array} \right] - \frac{1}{2} \right| \leq \mathrm{negl}(\lambda).$$

The above definition is based on the work of Mukherjee and Wichs [MW16] who formally introduced the notion of "expanding" ciphertexts. Here we work with this abstraction as it allows for a simpler exposition.

3.2.1 Threshold Decryption

An MLHE scheme that also supports a 1-round distributed decryption protocol additionally provides two more algorithms — $\mathsf{PartDec}$ and FinDec. The $\mathsf{PartDec}$ algorithm performs *partial* decryption of any expanded (and possibly evaluated) ciphertext given only a single party's secret key. And, the FinDec algorithm reconstructs the output message from the partial decryptions. Formally, we have

- $\mathsf{PartDec}((\mathsf{pk}_1, \ldots, \mathsf{pk}_N), i, \mathsf{sk}_i, \widehat{\mathsf{ct}}) \to \mathsf{sh}_i$. The algorithm takes as input a sequence of N public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_N$, secret key for the i^{th} user, an expanded ciphertext $\widehat{\mathsf{ct}}$, and outputs a *partial* decryption sh_i.
- $\mathsf{FinDec}(\mathsf{sh}_1, \ldots, \mathsf{sh}_N) \to m$. The algorithm takes as input a sequence of N partial decryptions $\mathsf{sh}_1, \ldots, \mathsf{sh}_N$, and outputs a message m.

For correctness, it is required that the message computed by first running partial decryption independently using all N keys and later combining them using the FinDec algorithm matches the actual output. And for security, apart from q-IND-CPA, it is also required that partial decryption for any user can be simulated given secret keys for all other users, and the encrypted message.

Correctness and Simulation Security. For any security parameter λ, circuit-depth bound d, consider parameters $\mathsf{params} \leftarrow \mathsf{Setup}(1^\lambda, 1^d)$, and any sequence of N key tuples $(\mathsf{pk}_i, \mathsf{ek}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{params})$ (for $i \leq N$). For any ℓ tuples of message-index pairs $(m_1, j_1), \ldots, (m_\ell, j_\ell)$, consider ciphertexts $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}_{j_i}, m_i)$ (i.e. encryptions of i^{th} message under its corresponding public key), and corresponding expanded ciphertexts $\widehat{\mathsf{ct}}_i \leftarrow \mathsf{Expand}((\mathsf{pk}_1, \ldots, \mathsf{pk}_N), j_i, \mathsf{ct}_i)$ (for $i \leq \ell$). The scheme MLHE is said to be (perfectly) correct if for any circuit C of depth $\leq d$, evaluated ciphertext $\widehat{\mathsf{ct}} \leftarrow \mathsf{Eval}(C, (\mathsf{ek}_1, \ldots, \mathsf{ek}_N), (\widehat{\mathsf{ct}}_1, \ldots, \widehat{\mathsf{ct}}_\ell))$, and partial decryptions $\mathsf{sh}_i \leftarrow \mathsf{PartDec}((\mathsf{pk}_1, \ldots, \mathsf{pk}_N), i, \mathsf{sk}_i, \widehat{\mathsf{ct}})$ for $i \in [N]$, the following holds:

- **Reconstruction.** $\mathsf{FinDec}(\mathsf{sh}_1, \ldots, \mathsf{sh}_N) = C(m_1, \ldots, m_\ell)$.
- **Simulation of Partial Decryption.** There exists a PPT simulator Sim^{thr} which on input an index $i \in [N]$, all but the i^{th} keys $\{\mathsf{sk}_j\}_{j \in [N] \setminus \{i\}}$, the evaluated ciphertext $\widehat{\mathsf{ct}}$, and the output message $\mathsf{msg} = C(m_1, \ldots, m_\ell)$ produces a simulated partial decryption sh'_i. The scheme is said to satisfy partial decryption simulation security if there exists a negligible function $\mathrm{negl}(\cdot)$ such that

$$\mathsf{SD}(\mathsf{Dist}, \mathsf{Dist}') \leq \mathrm{negl}(\lambda), \text{ where } \begin{array}{l} \mathsf{Dist} = \left\{\mathsf{sh}_i : \mathsf{sh}_i \leftarrow \mathsf{PartDec}((\mathsf{pk}_1, \ldots, \mathsf{pk}_N), i, \mathsf{sk}_i, \widehat{\mathsf{ct}})\right\} \\ \mathsf{Dist}' = \left\{\mathsf{sh}'_i : \mathsf{sh}'_i \leftarrow \mathsf{Sim}^{thr}(\mathsf{msg}, \widehat{\mathsf{ct}}, i, \{\mathsf{sk}_j\}_{j \in [N] \setminus \{i\}})\right\} \end{array}$$

  and SD denotes the statistical distance.

Definition 3.2. We say that a scheme MLHE satisfies partial decryption simulation security property if the above conditions hold.

3.3 Quantum Multi-Key Homomorphic Encryption

A quantum multi-key leveled homomorphic encryption (QMLHE) scheme QMLHE with message space $\mathcal{M}$ consists of six poly-time algorithms $\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Expand}, \mathsf{Eval}, \mathsf{Dec}$ with the following syntax:

- $\mathsf{Setup}(1^\lambda, 1^d) \to \mathsf{params}$. The setup algorithm takes as input the security parameter λ and the circuit size bound d and outputs the (classical) system parameters params.
- $\mathsf{KeyGen}(\mathsf{params}) \to (\mathsf{pk}, \rho_{\mathsf{ek}}, \mathsf{sk})$. The key generation algorithm takes as input the system parameters params and outputs a (classical) public key pk, a quantum evaluation key $\rho_{\mathsf{ek}} \in D(\mathcal{R}_{\mathsf{ek}})$ and (classical) secret key sk.

- $\mathsf{Enc}(\mathsf{pk}, \rho) \to \sigma$. The encryption algorithm takes as input a public key pk, a quantum state $\rho \in D(\mathcal{M})$ and outputs a cipherstate $\sigma \in D(\mathcal{C})$.
- $\mathsf{Expand}((\mathsf{pk}_1, \ldots, \mathsf{pk}_N), i, \sigma) \to \widehat{\sigma}$. The expansion algorithm takes as input a sequence of N public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_N$, a single cipherstate σ encrypted under the i^{th} public key, and it outputs an expanded cipherstate $\widehat{\sigma}$. For expanding multi-fold cipherstates, the expansion algorithm will be run as many times on each cipherstate.
- $\mathsf{Eval}(C, (\rho_{\mathsf{ek}_1}, \ldots, \rho_{\mathsf{ek}_\ell}), \widehat{\sigma}) \to \widehat{\sigma}'$. The evaluation algorithm takes as input description of a quantum circuit C along with an ℓ-tuple of evaluation keys ρ_{ek_i}, an n-fold cipherstates $\widehat{\sigma}$, and outputs an m-fold cipherstates $\widehat{\sigma}'$, where the circuit C takes as input n qubits and outputs m qubits.
- $\mathsf{Dec}(\mathsf{sk}_1, \ldots, \mathsf{sk}_N, \widehat{\sigma}) \to \rho$. The decryption algorithm takes as input a sequence of N secret keys $\mathsf{sk}_1, \ldots, \mathsf{sk}_N$, a single (expanded) cipherstate $\widehat{\sigma}$, and outputs a quantum state ρ in the message space $\mathcal{M}$. For decrypting multi-fold cipherstates, the decryption algorithm will be run as many times on each cipherstate.

In the above description, one could alternatively view the encryption, expansion, evaluation, and decryption algorithms as quantum channels where the evaluation algorithm could consume the evaluation keys partially, or even completely during the process. In this work, we consider only divisible schemes [BJ15] where decryption algorithm works qubit-by-qubit instead of decrypting a multi-qubit quantum cipherstate at once. We now define correctness, compactness and security properties for QMLHE schemes analogous to its classical counterpart. Here and throughout whenever we talk about quantum circuits of some bounded size d, we consider quantum circuits of size at most d represented using an apriori fixed gate set.

Correctness and Compactness. For any security parameter λ, circuit-size bound d, consider parameters $\mathsf{params} \leftarrow \mathsf{Setup}(1^\lambda, 1^d)$, and any sequence of N key tuples $(\mathsf{pk}_i, \rho_{\mathsf{ek}_i}, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{params})$ (for $i \le N$). Let $\Phi_{\mathsf{Enc}}^{\mathsf{pk}_i}$ denote the induced quantum channel that on input a quantum message state in $D(\mathcal{M})$, outputs the cipherstate encrypted under key pk_i in $D(\mathcal{C})$. Similarly, let $\Phi_{\mathsf{Expand}}^{(\mathsf{pk}_j)_j, i}$, $\Phi_{\mathsf{Eval}}^{C, (\rho_{\mathsf{ek}_j})_j, i}$ and $\Phi_{\mathsf{Dec}}^{(\mathsf{sk}_j)_j}$ denote the induced quantum channels for cipherstate expansion, evaluation, and decryption (respectively). The scheme QMLHE is said to be (perfectly) correct if the following holds:

Expansion. For every index $i \in N$, $\left\| \Phi_{\mathsf{Dec}}^{(\mathsf{sk}_j)_j} \circ \Phi_{\mathsf{Expand}}^{i, (\mathsf{pk}_j)_j} \circ \Phi_{\mathsf{Enc}}^{\mathsf{pk}_i} - \mathbb{I} \right\|_\diamond = 0$.

Evaluation. For any N (non-negative) sequence of indices $n_1, \ldots, n_N$, and for any quantum circuit C of size $\le d$ (represented using an apriori fixed gate set) with induced channel $\Phi_C : \mathcal{M}^{\otimes \sum_i n_i} \to \mathcal{M}^{\otimes m}$,

$$\left\| \left(\Phi_{\mathsf{Dec}}^{(\mathsf{sk}_j)_j} \right)^{\otimes m} \circ \Phi_{\mathsf{Eval}}^{C, (\rho_{\mathsf{ek}_j})_j} \circ \otimes_{i=1}^{N} \left(\Phi_{\mathsf{Expand}}^{i, (\mathsf{pk}_j)_j} \circ \Phi_{\mathsf{Enc}}^{\mathsf{pk}_i} \right)^{\otimes n_i} - \Phi_C \right\|_\diamond = 0.$$

Let $\widehat{\sigma}$ denote the quantum cipherstate that is output by the evaluation algorithm above, i.e. output of the quantum channel $\Phi_{\mathsf{Eval}}^{C, (\rho_{\mathsf{ek}_j})_j}$. The scheme QMLHE is said to be compact if there exists a polynomial $p(\cdot, \cdot, \cdot)$ such that $|\widehat{\sigma}| \le p(\lambda, d, N)$, i.e. size of $\widehat{\sigma}'$ does not depend on circuit C or the number of inputs $\sum_i n_i$.[14]

Security. For security, we require the scheme to satisfy q-IND-CPA security.

Definition 3.3 (q-IND-CPA)**.** A quantum multi-key leveled homomorphic encryption scheme QMLHE = (Setup, KeyGen, Enc, Expand, Eval, Dec) is q-IND-CPA secure if for every *stateful quantum* PT adversary $\mathcal{A}$,

[14]Here and throughout when we say that the size of encrypted state does not depend on the circuit C we mean that it should not grow with the total number of quantum gates in C but is allowed to (linearly) depend on the number of output qubits, that is m.

there exists a negligible function negl$(\cdot)$, such that for all $\lambda \in \mathbb{N}$, the following holds:

$$\left| \Pr\left[\mathcal{A}(\mathcal{R}_{\mathcal{C}}^{(b)}) = b : \begin{array}{c} \mathsf{params} \leftarrow \mathsf{Setup}(1^\lambda, 1^d);\ (\mathsf{pk}, \mathcal{R}_{\mathsf{ek}}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{params}) \\ \mathcal{R}_{\mathcal{M}} \leftarrow \mathcal{A}(\mathsf{params}, \mathsf{pk}, \mathcal{R}_{\mathsf{ek}});\ b \leftarrow \{0,1\} \\ \mathcal{R}_{\mathcal{C}}^{(0)} \leftarrow \mathsf{Enc}(\mathsf{pk}, |0\rangle\langle 0|);\ \mathcal{R}_{\mathcal{C}}^{(1)} \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathcal{R}_{\mathcal{M}}) \end{array} \right] - \frac{1}{2} \right| \leq \mathrm{negl}(\lambda).$$

Here $\mathcal{R}_{\mathsf{ek}}$ corresponds to the quantum registers containing the evaluation key, $\mathcal{R}_{\mathcal{M}}$ corresponds to a quantum register containing a state in $\mathcal{M}$, and $\mathcal{R}_{\mathcal{C}}^{(0)}, \mathcal{R}_{\mathcal{C}}^{(1)}$ correspond to the registers encrypting the state $|0\rangle$ and the state in $\mathcal{R}_{\mathcal{M}}$, respectively. Note that since the adversary is a stateful quantum adversary, therefore the contents of the register $\mathcal{R}_{\mathcal{M}}$ are allowed to be arbitrarily entangled with the adversary's local states. (Also, in the above experiment, the challenger basically receives a input-state from the adversary and either encrypts it directly (if $b = 1$), otherwise erases the register and then encrypts it.)

3.3.1 Threshold Decryption

Due to the no-cloning folklore, we know that one can not create copies of an arbitrary quantum state. Thus, the notion of threshold decryption in the case of quantum multi-key homomorphic encryption can not be defined in an analogous way to its classical counterpart, at least directly. To avoid the no-cloning pitfall, we assume that the quantum ciphertext $\widehat{\sigma}$ can be divided into two components — purely classical and purely quantum. Now the $\mathsf{PartDec}$ algorithm only takes as input the classical component of the ciphertext $\widehat{\sigma}$, and outputs some partial decryption information such that FinDec algorithm combines all the partial decryption information along with the quantum component of $\widehat{\sigma}$ to compute the message state. Formally, we have that every expanded (possibly evaluated) cipherstate $\widehat{\sigma}$ can be decomposed as $\widehat{\sigma} = \rho(\widehat{\mathsf{ct}}) \otimes \widetilde{\sigma}$ (where $\widehat{\mathsf{ct}}$ denotes the purely classical component and $\widetilde{\sigma}$ denotes the quantum component). Now due to technical reasons pertaining to defining simulation security for QMLHE, we divide FinDec procedure into two separate algorithms — $\mathsf{FinDecPre}$ and $\mathsf{FinDecPost}$. Here $\mathsf{FinDecPre}$ performs a classical reconstruction operation given only partial decryptions, and $\mathsf{FinDecPost}$ takes that reconstructed key to perform the final quantum decryption operation. The algorithms are described as follows

- $\mathsf{PartDec}((\mathsf{pk}_1, \ldots, \mathsf{pk}_N), i, \mathsf{sk}_i, \widehat{\mathsf{ct}}) \to \mathsf{sh}_i$. The algorithm takes as input a sequence of N public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_N$, secret key for the i^{th} user, classical component of an expanded ciphertext $\widehat{\mathsf{ct}}$, and outputs a *partial* decryption sh_i.
- $\mathsf{FinDecPre}(\mathsf{sh}_1, \ldots, \mathsf{sh}_N) \to \mathsf{rk}$. The algorithm takes as input a sequence of N partial decryptions $\mathsf{sh}_1, \ldots, \mathsf{sh}_N$, and outputs a reconstructed key rk.
- $\mathsf{FinDecPost}(\widetilde{\sigma}, \mathsf{rk}) \to \rho$. The algorithm takes as input a quantum component of an expanded ciphertext $\widetilde{\sigma}$, a reconstructed key rk, and outputs a quantum state ρ in the message space $\mathcal{M}$.

Now the correctness condition is defined analogously. And for simulation security, it is only required that partial decryption for any user can be simulated given secret keys for all other users, and the final reconstruction key. Additionally, we require that the reconstruction key rk is known at the time of encryption for any ciphertext.

Correctness, Unique Reconstruction and Simulation Security. For any security parameter λ, circuit-size bound d, consider parameters $\mathsf{params} \leftarrow \mathsf{Setup}(1^\lambda, 1^d)$, and any sequence of N key tuples $(\mathsf{pk}_i, \rho_{\mathsf{ek}_i}, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{params})$ (for $i \leq N$). The scheme QMLHE is said to be (perfectly) correct if the following holds:

Reconstruction Correctness. For any N (non-negative) indices $n_1, \ldots, n_N$, any input-state $\rho \in D(\mathcal{M}^{\otimes \sum_i n_i} \otimes \mathcal{E})$, any quantum circuit C of size $\leq d$ (represented using an apriori fixed gate set) with induced channel

$\Phi_C : \mathcal{M}^{\otimes \sum_i n_i} \to \mathcal{M}^{\otimes m}$, the following holds:

$$\left\| \begin{array}{r} \left(\Phi^{\mathsf{rk}}_{\mathsf{FinDecPost}} \otimes \mathbb{I}^{\mathcal{E}}\right) \widetilde{\sigma} \\ - \left(\Phi_C \otimes \mathbb{I}^{\mathcal{E}}\right) \rho \end{array} \right\|_1 = 0, \text{ where } \quad \begin{array}{c} \widehat{\sigma} = \left(\left(\Phi^{C,(\rho_{\mathsf{ek}_j})_j}_{\mathsf{Eval}} \circ \otimes_{i=1}^{N} \left(\Phi^{i,(\mathsf{pk}_j)_j}_{\mathsf{Expand}} \circ \Phi^{\mathsf{pk}_i}_{\mathsf{Enc}} \right)^{\otimes n_i} \right) \otimes \mathbb{I}^{\mathcal{E}} \right) \rho, \\ \widehat{\sigma} = \rho(\widehat{\mathsf{ct}}) \otimes \widetilde{\sigma}, \quad (\forall i \in [n]) \ \mathsf{sh}_i \leftarrow \mathsf{PartDec}((\mathsf{pk}_j)_j, i, \mathsf{sk}_i, \widehat{\mathsf{ct}}), \\ \mathsf{rk} = \mathsf{FinDecPre}(\mathsf{sh}_1, \ldots, \mathsf{sh}_N). \end{array}$$

In the expression $\widehat{\sigma} = \rho(\widehat{\mathsf{ct}}) \otimes \widetilde{\sigma}$, we mean that $\widehat{\mathsf{ct}}$ denotes the purely classical component of the cipherstate $\widehat{\sigma}$, and $\widetilde{\sigma}$ the quantum component.

Unique Reconstruction. For every index $i \in [N]$, and any input-state $\rho \in D(\mathcal{M} \otimes \mathcal{E})$, consider the cipherstate σ which encrypts ρ under pk_i, that is $\sigma = \left(\Phi^{\mathsf{pk}_i}_{\mathsf{Enc}} \otimes \mathbb{I}^{\mathcal{E}}\right) \rho$. Let ct denote the purely classical component of cipherstate σ, and $\widetilde{\sigma}$ the quantum component. Also, let r_i denote the classical random coins[15] used while computing the quantum component of cipherstate $\widetilde{\sigma}$ from the input-state ρ. We say that the scheme satisfies unique reconstruction property if there exists a deterministic PPT algorithm $\mathsf{Extract}$ such that $\mathsf{Extract}(r) = \mathsf{rk} = \mathsf{FinDecPre}(\mathsf{PartDec}(\mathsf{pk}_i, 1, \mathsf{sk}_i, \mathsf{ct}))$.[16]

Simulation of Partial Decryption. There exists a quantum PT simulator Sim^{thr} which on input an index $i \in [N]$, all but the i^{th} keys $\{\mathsf{sk}_j\}_{j \in [N] \setminus \{i\}}$, the classical component of evaluated ciphertext $\widehat{\mathsf{ct}}$, and the reconstruction key rk produces a simulated partial decryption sh'_i. The scheme is said to satisfy partial decryption simulation security if for any N (non-negative) indices $n_1, \ldots, n_N$, any input-state $\rho \in D(\mathcal{M}^{\otimes \sum_i n_i} \otimes \mathcal{E})$, any quantum circuit C of size $\leq d$, there exists a negligible function $\mathrm{negl}(\cdot)$ such that

$$\mathsf{SD}(\mathsf{Dist}, \mathsf{Dist}') \leq \mathrm{negl}(\lambda), \text{ where } \quad \begin{array}{c} \mathsf{Dist} = \left\{ \mathsf{sh}_i : \mathsf{sh}_i \leftarrow \mathsf{PartDec}((\mathsf{pk}_1, \ldots, \mathsf{pk}_N), i, \mathsf{sk}_i, \widehat{\mathsf{ct}}) \right\} \\ \mathsf{Dist}' = \left\{ \mathsf{sh}'_i : \mathsf{sh}'_i \leftarrow \mathsf{Sim}^{thr}(\mathsf{rk}, \widehat{\mathsf{ct}}, i, \{\mathsf{sk}_j\}_{j \in [N] \setminus \{i\}}) \right\} \\ \widehat{\sigma} = \left(\left(\Phi^{C,(\rho_{\mathsf{ek}_j})_j}_{\mathsf{Eval}} \circ \otimes_{i=1}^{N} \left(\Phi^{i,(\mathsf{pk}_j)_j}_{\mathsf{Expand}} \circ \Phi^{\mathsf{pk}_i}_{\mathsf{Enc}} \right)^{\otimes n_i} \right) \otimes \mathbb{I}^{\mathcal{E}} \right) \rho, \\ \widehat{\sigma} = \rho(\widehat{\mathsf{ct}}) \otimes \widetilde{\sigma}, \quad (\forall i \in [n]) \ \mathsf{sh}_i \leftarrow \mathsf{PartDec}((\mathsf{pk}_j)_j, i, \mathsf{sk}_i, \widehat{\mathsf{ct}}), \\ \mathsf{rk} = \mathsf{FinDecPre}(\mathsf{sh}_1, \ldots, \mathsf{sh}_N). \end{array}$$

and SD denotes the statistical distance.

Definition 3.4. We say that a scheme QMLHE satisfies partial decryption simulation security property if the above conditions hold.

3.3.2 Re-Randomizability of Cipherstates

Now we define the notion of re-randomization for quantum multi-key HE. It is a slightly stronger property than quantum circuit privacy. At a high level, it states that there exists a re-randomization algorithm that takes as input HE public keys and a possibly evaluated cipherstate, and re-randomizes the cipherstate such that new cipherstate looks indistinguishable from a freshly generated cipherstate encrypting the same quantum state. Formally, we have an additional algorithm ReRand described as follows

- $\mathsf{ReRand}((\mathsf{pk}_1, \ldots, \mathsf{pk}_N), \widehat{\sigma}) \to \widehat{\sigma}'$. The re-randomization algorithm takes as input a sequence of N public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_N$, and an expanded (possibly evaluated) cipherstate $\widehat{\sigma}$, and outputs a cipherstate $\widehat{\sigma}'$.

Now correctness is defined naturally, i.e. the re-randomized cipherstate should decrypt to the correct message state. For security, we require that a re-randomized cipherstate is indistinguishable from a freshly expanded cipherstate encrypting the same message state.

[15]We would like to stress that here the state σ corresponds to an honestly encrypted input-state, and *not* an adversarially constructed state. Since we only define the unique reconstruction property for honestly computed cipherstates, thus we could explicitly refer to the classical random coins used by the quantum algorithm during encryption.

[16]Similarly, one could also extend this property such that the classical random coins used by Expand algorithm could be used to generate the updates for the reconstructed key during ciphertext expansion.

Correctness and Re-Randomizability. For any security parameter λ, circuit-size bound d, consider parameters $\mathsf{params} \leftarrow \mathsf{Setup}(1^\lambda, 1^d)$, and any sequence of N key tuples $(\mathsf{pk}_i, \rho_{\mathsf{ek}_i}, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{params})$ (for $i \leq N$). Let $\Phi^{\mathsf{pk}_i}_{\mathsf{Enc}}$, $\Phi^{(\mathsf{pk}_j)_j, i}_{\mathsf{Expand}}$, $\Phi^{C,(\rho_{\mathsf{ek}_j})_j, i}_{\mathsf{Eval}}$, $\Phi^{(\mathsf{sk}_j)_j}_{\mathsf{Dec}}$, and $\Phi^{(\mathsf{pk}_j)_j}_{\mathsf{ReRand}}$ denote the induced quantum channels for encryption, cipherstate expansion, evaluation, decryption, and re-randomization (respectively). The scheme QMLHE is said to be (perfectly) correct if the following holds:

Decryption of Re-Randomized Ciphertext. For any N (non-negative) sequence of indices $n_1, \ldots, n_N$, and for any quantum circuit C of size $\leq d$ (represented using an apriori fixed gate set) with induced channel $\Phi_C : \mathcal{M}^{\otimes \sum_i n_i} \to \mathcal{M}^{\otimes m}$,

$$\left\| \left(\Phi^{(\mathsf{sk}_j)_j}_{\mathsf{Dec}}\right)^{\otimes m} \circ \left(\Phi^{(\mathsf{pk}_j)_j}_{\mathsf{ReRand}}\right)^{\otimes m} \circ \Phi^{C,(\rho_{\mathsf{ek}_j})_j}_{\mathsf{Eval}} \circ \otimes_{i=1}^{N} \left(\Phi^{i,(\mathsf{pk}_j)_j}_{\mathsf{Expand}} \circ \Phi^{\mathsf{pk}_i}_{\mathsf{Enc}}\right)^{\otimes n_i} - \Phi_C \right\|_\diamond = 0.$$

Re-Randomizability. The scheme is said to be re-randomizable if for any index $k \in [N]$, any N (non-negative) sequence of indices $n_1, \ldots, n_N$, and for any quantum circuit C of size $\leq d$ (represented using an apriori fixed gate set) with induced channel $\Phi_C : \mathcal{M}^{\otimes \sum_i n_i} \to \mathcal{M}^{\otimes m}$, there exists a negligible function $\mathrm{negl}(\cdot)$ such that

$$\left\| \left(\Phi^{(\mathsf{pk}_j)_j}_{\mathsf{ReRand}}\right)^{\otimes m} \circ \Phi^{C,(\rho_{\mathsf{ek}_j})_j}_{\mathsf{Eval}} \circ \otimes_{i=1}^{N} \left(\Phi^{i,(\mathsf{pk}_j)_j}_{\mathsf{Expand}} \circ \Phi^{\mathsf{pk}_i}_{\mathsf{Enc}}\right)^{\otimes n_i} - \left(\Phi^{k,(\mathsf{pk}_j)_j}_{\mathsf{Expand}} \circ \Phi^{\mathsf{pk}_k}_{\mathsf{Enc}}\right)^{\otimes m} \circ \Phi_C \right\|_\diamond \leq \mathrm{negl}(\lambda).$$

Definition 3.5. We say that a scheme QMLHE satisfies ciphertext re-randomizability property if the above conditions hold.

3.4 Branching Programs

Branching programs are a model of computation used to capture space-bounded computations [BDFP86, Bar86]. In this work, we will be using a restricted notion called *permutation branching programs*.

Definition 3.6 (Permutation Branching Program)**.** A permutation branching program of length L, width w and input space $\{0,1\}^n$ consists of a sequence of $2L$ permutations $\sigma_{i,b} : \{1, \ldots, w\} \to \{1, \ldots, w\}$ for $1 \leq i \leq L, b \in \{0,1\}$, an input selection function $\mathsf{inp} : \{1, \ldots, L\} \to \{1, \ldots, n\}$, an accepting state $\mathsf{acc} \in \{1, \ldots, w\}$ and a rejection state $\mathsf{rej} \in \{1, \ldots, w\}$. The starting state st_0 is set to be 1 without loss of generality. The branching program evaluation on input $x \in \{0,1\}^n$ proceeds as follows:

- For $i = 1$ to L,
  - Let $\mathsf{pos} = \mathsf{inp}(i)$ and $b = x_{\mathsf{pos}}$. Compute $\mathsf{st}_i = \sigma_{i,b}(\mathsf{st}_{i-1})$.
- If $\mathsf{st}_L = \mathsf{acc}$, output 1. If $\mathsf{st}_L = \mathsf{rej}$, output 0, else output $\perp$.

In a remarkable result, Barrington [Bar86] showed that any circuit of depth d can be simulated by a permutation branching program of width 5 and length 4^d.

Theorem 3.1 ([Bar86])**.** For any boolean circuit C with input space $\{0,1\}^n$ and depth d, there exists a permutation branching program BP of width 5 and length 4^d such that for all inputs $x \in \{0,1\}^n$, $C(x) = \mathsf{BP}(x)$.

This permutation property will be useful later in our construction. We also need to define a notion of "interval-alternating" branching programs. A branching program (for n-bit inputs) is said to be an $\mathbf{S}$-interval-alternating branching program (where $\mathbf{S} = (S_1, \ldots, S_{N+1})$ is a sequence of $N+1$ intervals of set $\{1, \ldots, n\}$ for some $N \geq 1$) if the input bits read during odd transitions lie in interval S_{N+1} and the input bit read during i^{th} even transition lies in interval $S_{(i-1 \bmod N)+1}$. In other words, the input bit read at each transition alternates between first N intervals and the last interval, and input bits (in the even transitions) are read in a round-robin fashion. Also, the length of an interval-alternating branching program must always be even. Formally, it is defined as follows.

Definition 3.7 (Interval-Alternating Permutation Branching Program)**.** A permutation branching program of length L with input space $\{0,1\}^n$ is said to be an $(S_1, \ldots, S_{N+1})$-interval-alternating branching program if L is even, and for all $\ell \leq L/2$, $\mathsf{inp}(2\ell - 1) \in S_{N+1}$ and $\mathsf{inp}(2\ell) \in S_{(\ell-1 \bmod N)+1}$.

We would like to point out that any permutation branching program of length L and input space $\{0,1\}^n$ can be easily transformed to an $\mathbf{S}$-interval-alternating branching program of length $2N \cdot L$ for any sequence of $N+1$ set intervals $\mathbf{S} = (S_1, \ldots, S_{N+1})$. We will use the following corollary, which follows from Theorem 3.1.

Corollary 3.1. For any boolean circuit C with input space $\{0,1\}^n$ and depth d, and for any set intervals $\mathbf{S} = (S_1, \ldots, S_{N+1})$, there exists an $\mathbf{S}$-interval-alternating branching program BP of width 5 and length $2 \cdot N \cdot 4^d$ such that for all inputs $x \in \{0,1\}^n$, $C(x) = \mathsf{BP}(x)$.

4 Our Framework: QMPHE and COQT

4.1 Quantum Multi-Key Positional Homomorphic Encryption (QMPHE)

A quantum multi-key positional homomorphic encryption (QMPHE) scheme is a relaxation of quantum multi-key homomorphic encryption in which the key generation algorithm also takes as input the number of parties N and a position $i \leq N$. This puts two restrictions — (1) number of users/parties need to be known and fixed during key generation, and (2) ciphertexts computed using public keys of any two users/parties, generated for the same position i, can not be combined together, i.e. expansion and evaluation algorithms can not be used to perform homomorphic computations on multiple keys for the same position. Later we show how to remove these restrictions. Specifically, we show that a (quantum) multi-key positional HE scheme gives a (quantum) multi-key HE scheme with bounded number of users.

More formally, a quantum multi-key positional leveled homomorphic encryption (QMPLHE) scheme QMPLHE with message space $\mathcal{M}$ consists of six poly-time algorithms $\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Expand}, \mathsf{Eval}, \mathsf{Dec}$[17] with all algorithms having same syntax as for a QMLHE scheme, except the key generation and evaluation algorithms that have the following syntax:

- $\mathsf{KeyGen}(\mathsf{params}, N, i) \to (\mathsf{pk}, \rho_{\mathsf{ek}}, \mathsf{sk})$. The setup algorithm takes as input the system parameters params, number of users N, a *position* $i \leq N$, and outputs a (classical) public key pk, a quantum evaluation key $\rho_{\mathsf{ek}} \in D(\mathcal{R}_{\mathsf{ek}}^{(i,N)})$ and (classical) secret key sk.[18]
- $\mathsf{Eval}(C, (\rho_{\mathsf{ek}_1}, \ldots, \rho_{\mathsf{ek}_N}), \widehat{\sigma}) \to \widehat{\sigma}'$. The evaluation algorithm takes as input description of a quantum circuit C along with an N-tuple of evaluation keys ρ_{ek_i}, an n-fold cipherstates $\widehat{\sigma}$, and outputs an m-fold cipherstates $\widehat{\sigma}'$, where the circuit C takes as input n qubits and outputs m qubits.[19]

The compactness and security properties for the positional schemes is same as that for standard (non-positional) schemes.[20] The correctness of a positional scheme is only guaranteed when all the N key pairs are generated for a distinct position in 1 through N. For completeness, below we define the correctness property.[21]

Correctness. For any security parameter λ, circuit-size bound d, consider parameters $\mathsf{params} \leftarrow \mathsf{Setup}(1^\lambda, 1^d)$, and any sequence of N key tuples $(\mathsf{pk}_i, \rho_{\mathsf{ek}_i}, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{params}, N, i)$ (for $i \leq N$). Let $\Phi_{\mathsf{Enc}}^{\mathsf{pk}_i}$ denote the induced quantum channel that on input a quantum message state in $D(\mathcal{M})$, outputs the cipherstate encrypted under key pk_i in $D(\mathcal{C})$. Similarly, let $\Phi_{\mathsf{Expand}}^{(\mathsf{pk}_j)_j, i}$, $\Phi_{\mathsf{Eval}}^{C, (\rho_{\mathsf{ek}_j})_j, i}$ and $\Phi_{\mathsf{Dec}}^{(\mathsf{sk}_j)_j}$ denote the induced quantum channels for cipherstate expansion, evaluation, and decryption (respectively). The scheme QMLHE is said to be (perfectly) correct if the following holds:

[17] The algorithms for threshold decryption are defined exactly as before.

[18] Note that the quantum evaluation key space is allowed to depend on the position and number of users.

[19] The only difference being that now the evaluation algorithm always takes as inputs all N evaluation keys, whereas previously there was no such restriction set during setup.

[20] For security, we would require that q-IND-CPA holds for all positions.

[21] Correctness of threshold decryption can be analogously defined.

Expansion. For every index $i \in N$, $\left\| \Phi_{\mathsf{Dec}}^{(\mathsf{sk}_j)_j} \circ \Phi_{\mathsf{Expand}}^{i,(\mathsf{pk}_j)_j} \circ \Phi_{\mathsf{Enc}}^{\mathsf{pk}_i} - \mathbb{I} \right\|_{\diamond} = 0$.

Evaluation. For any N (non-negative) sequence of indices $n_1, \ldots, n_N$, and for any quantum circuit C of size $\leq d$ (represented using an apriori fixed gate set) with induced channel $\Phi_C : \mathcal{M}^{\otimes \sum_i n_i} \to \mathcal{M}^{\otimes m}$,

$$\left\| \left(\Phi_{\mathsf{Dec}}^{(\mathsf{sk}_j)_j} \right)^{\otimes m} \circ \Phi_{\mathsf{Eval}}^{C,(\rho_{\mathsf{ek}_j})_j} \circ \otimes_{i=1}^{N} \left(\Phi_{\mathsf{Expand}}^{i,(\mathsf{pk}_j)_j} \circ \Phi_{\mathsf{Enc}}^{\mathsf{pk}_i} \right)^{\otimes n_i} - \Phi_C \right\|_{\diamond} = 0.$$

Looking ahead, the position i during key generation will act as a guideline for us to generate the evaluation key during key generation.

4.2 Conditional Oblivious Quantum Transform

A conditional oblivious quantum transform (COQT) is a distributed non-interactive encoding procedure. It allows encoding multiple bit-strings $x_1, \ldots, x_N$ independently with respect to a classical circuit C into encoded states σ_i and decoding key key_i such that the encodings $\sigma_1, \ldots, \sigma_N$ could be used to conditionally apply a quantum gate G on a single-qubit state ρ, where the predicate is $C(x_1, \ldots, x_N, x_{N+1})$ and x_{N+1} is any arbitrary bit-string of appropriate length chosen while applying the transform. The security requirement is that an encoded state σ reveals no information about the associated input/bit-string x if one does not know its corresponding decoding key key.

Looking ahead to our construction, we show how to

1. Construct unconditionally secure COQT scheme for class of log-depth circuits (i.e., $\mathbf{NC}^1$),
2. Bootstrap a secure COQT scheme for class of log-depth circuits to all poly-size circuits using classical multi-key leveled homomorphic encryption with log-depth decryption circuits.

Let $\mathcal{C}_k$ be a class of classical circuits that takes as input k bits and outputs 1 bit, and let GS be a set of single-qubit gates. A conditional oblivious quantum transform COQT for circuit class $\mathcal{C}_k$ and gate set GS consists of three poly-time algorithms $\mathsf{Encode}, \mathsf{Apply}, \mathsf{Decode}$ that have the following syntax:

- $\mathsf{Encode}(C, (S_1, \ldots, S_{N+1}), i, x, \mathsf{G}) \to (\sigma, \mathsf{key})$. The encoding algorithm takes as input a classical circuit $C \in \mathcal{C}_k$, an $(N+1)$-tuple of set intervals[22] $S_i \subset \{1, \ldots k\}$, index $i \leq N$, a bit string x of length $|S_i|$, and a single-qubit gate $\mathsf{G} \in \mathsf{GS}$. It outputs a quantum encoded state σ, classical decoding key key.
- $\mathsf{Apply}(C, (S_1, \ldots, S_{N+1}), \sigma_1, \ldots, \sigma_N, x, \rho) \to (\rho', \mathsf{aux})$. The apply algorithm takes as input a classical circuit $C \in \mathcal{C}_k$, an $(N+1)$-tuple of set intervals S_i, a sequence of N encoded states σ_i, an input string x of length $|S_{N+1}|$, and a state ρ.[23] It outputs a transformed quantum state ρ', auxiliary classical information aux.
- $\mathsf{Decode}(\mathsf{key}_1, \ldots, \mathsf{key}_N, \mathsf{aux}) \to (a, b)$. The decoding algorithm takes as input N decoding keys key_i and auxiliary information aux, and outputs two bits (a, b).

We now define correctness and security properties for COQT schemes.

Correctness. For any circuit $C \in \mathcal{C}_k$, $N+1$ non-empty intervals $S_i \subset \{1, \ldots k\}$, $N+1$ inputs $x_i \in \{0,1\}^{|S_i|}$, single-qubit gate $\mathsf{G} \in \mathsf{GS}$, consider encodings $(\sigma_i, \mathsf{key}_i) \leftarrow \mathsf{Encode}(C, (S_1, \ldots, S_{N+1}), i, x_i, \mathsf{G})$ for all $i \leq N$. For any input-state $\rho \in D(\mathbb{C}^{\{0,1\}} \otimes \mathcal{E})$, consider transformed state $(\rho', \mathsf{aux}) \leftarrow \mathsf{Apply}(C, (S_1, \ldots, S_{N+1}),$

[22]Recall that set intervals are simply contiguous set partitions. For example, if S_1, S_2 are set intervals of $\{1, \ldots, k\}$, then $S_1 = \{1, \ldots, \ell\}$ and $S_2 = \{\ell + 1, \ldots, k\}$ for some ℓ.

[23]Basically, the apply algorithm takes as input a single quantum register, thus transforms a single-qubit state at a time.

$\sigma_1, \ldots, \sigma_N, x_{N+1}, \rho)$.[24] Let $(a, b) \leftarrow \mathsf{Decode}(\mathsf{key}_1, \ldots, \mathsf{key}_N, \mathsf{aux})$. The scheme COQT is said to be (perfectly) correct if the following holds:

$$\left\| \rho' - \left(\mathsf{X}^a \mathsf{Z}^b \mathsf{G}^{C(x_1,\ldots,x_{N+1})} \otimes \mathbb{I}^{\mathcal{E}} \right) \rho \left(\mathsf{X}^a \mathsf{Z}^b \mathsf{G}^{C(x_1,\ldots,x_{N+1})} \otimes \mathbb{I}^{\mathcal{E}} \right)^{\dagger} \right\|_1 = 0.$$

That is, output of the apply algorithm is same as first applying gate G on the first register of state ρ if $C(x_1, \ldots, x_{N+1}) = 1$, and then a quantum one-time pad with keys (a, b) on the same register.

Security. For security, we require that no quantum PT adversary should be able to distinguish quantum encoded states of two different input strings $x^{(0)}, x^{(1)}$. In other words, if an adversary does not receive the decoding keys, then for any circuit C, input intervals, index, single-qubit gate, the encoded state on any two input strings will be indistinguishable.

Definition 4.1. A conditional oblivious quantum transform $\mathsf{COQT} = (\mathsf{Encode}, \mathsf{Apply}, \mathsf{Decode})$ is secure if for every *stateful quantum* PT adversary $\mathcal{A}$, there exists a negligible function $\mathrm{negl}(\cdot)$, such that for every circuit $C \in \mathcal{C}_k$, single-qubit gate $\mathsf{G} \in \mathsf{GS}$, the following holds:

$$\left| \Pr\left[\mathcal{A}(\mathcal{R}_\sigma^{(b)}) = b : \begin{array}{c} ((S_1, \ldots, S_{N+1}), i, x^{(0)}, x^{(1)}) \leftarrow \mathcal{A}(C, \mathsf{G});\ b \leftarrow \{0,1\} \\ (\mathcal{R}_\sigma^{(b)}, \mathsf{key}_b) \leftarrow \mathsf{Encode}(C, (S_1, \ldots, S_{N+1}), i, x^{(b)}, \mathsf{G}) \end{array} \right] - \frac{1}{2} \right| \leq \mathrm{negl}(k),$$

where $\{S_i\}_{i \leq N+1}$ are non-empty intervals ($S_i \subset \{1, \ldots k\}$ for $i \leq N+1$), index $i \leq N$, and strings $x^{(0)}, x^{(1)} \in \{0,1\}^{|S_i|}$, and $\mathcal{R}_\sigma^{(b)}$ denotes the output quantum registers.

5 Constructing Quantum Multi-Key Positional HE

In this section, we construct a quantum multi-key positional homomorphic encryption scheme (for quantum circuits with polynomially bounded T-gates) from a classical multi-key (leveled) homomorphic encryption scheme and a conditional oblivious quantum transform. Our construction satisfies compactness requirement as the size of the ciphertexts does not grow with the quantum circuit being evaluated (however, the size of evaluation key grows linearly with the number of T-gates in the quantum circuit). For simplicity, we describe the construction starting with a (classical) multi-key *fully* homomorphic encryption scheme. Later in Section 7, we discuss how to set the circuit depth bound during setup thereby relying only on a leveled scheme. Also, in Section 6, we show that our scheme also provides threshold decryption if the underlying classical scheme provides threshold decryption. Below we give a brief overview.

Outline. The public parameters will consist of the MFHE parameters params and the T-gate bound k. Now, to sample the keys for position pos and N users, the key generation algorithm starts by sampling $k+1$ MFHE key tuples $(\mathsf{pk}_i, \mathsf{ek}_i, \mathsf{sk}_i)$. Next, it computes k COQTs (i.e., runs COQT encoding algorithm) for MFHE decryption circuit, input string sk_i, position pos and gate $\mathsf{P}^\dagger$. Let $(\sigma_i, \mathsf{key}_i)$ be the corresponding quantum encodings and classical decoding keys. It computes $\mathsf{ct}_{\mathsf{key},i}$ and $\mathsf{ct}_{\mathsf{sk},i}$ as encryptions of keys key_i and sk_i under public key pk_{i+1}. Finally, it sets the secret key as all the MFHE secret keys, the public key as MFHE public keys, and the quantum evaluation key will contain all $\mathsf{ek}_i, \sigma_i, \mathsf{ct}_{\mathsf{key},i}, \mathsf{ct}_{\mathsf{sk},i}$ components. For ease of exposition, we will say $(\mathsf{pk}_i, \mathsf{ek}_i, \mathsf{sk}_i)$ are at level i, and similarly for other components.

Next, we describe the encryption, decryption and expansion algorithms. First, encryption simply involves applying a quantum one-time pad and encrypting the one-time pad keys under the base (i.e., level 1) public key. For decryption, the algorithm first decrypts the MFHE ciphertext using appropriate secret key to get the one-time pad keys and then performs one-time pad decryption. The expansion algorithm is straightforward as it involves expanding only the MFHE ciphertext.

Now, the evaluation algorithm works as follows. It starts by expanding the key and sk MFHE ciphertexts for all k levels and N users. Next, it performs the evaluation gate-by-gate. For evaluating a gate in the

[24]Here the apply algorithm is run on the first register of state ρ.

Clifford group, it applies the corresponding gate directly to the encrypted state(s) and then updates one-time pad keys using MFHE evaluation. For evaluating a T-gate, it starts by applying the T-gate on the corresponding (encrypted) state followed by COQT application. Suppose the MFHE ciphertexts encrypting the one-time pad keys are at level ℓ. For applying the COQT on the encrypted state, it first selects the level ℓ encoding in j^{th} user's (quantum) evaluation key, say $\sigma_\ell^{(j)}$, as the encodings for j^{th} position and sets the ciphertext encrypting the error indicator bit[25] as the input string. By correctness of COQT, this removes the P error (if any). Let aux be the auxiliary information generated during application. Next, it generates the key updates by homomorphically running the COQT decoding algorithm (with aux hardwired) on expanded key ciphertexts at level ℓ. Note that the output of this evaluation will be encrypted under level $\ell + 1$ MFHE public keys of all N users, since level ℓ key and sk ciphertexts are encrypted under level $\ell + 1$ public keys. Thus, before updating the one-time pad keys, the evaluator recrypts the one-time pad keys for *all* wires to level $\ell + 1$ (using expanded sk ciphertexts). And, finally it updates the one-time pad key by (homomorphically) $\oplus$-ing the original one-time pad keys with the key updates. Observe that after ℓ^{th} T-gate evaluation, the quantum one-time pad keys are at level $\ell + 1$. Thus, we can evaluate a quantum circuit with at most k T gates. Below we describe our construction in detail.

5.1 Construction

Let $\mathsf{MFHE} = (\mathsf{MFHE.Setup}, \mathsf{MFHE.KeyGen}, \mathsf{MFHE.Enc}, \mathsf{MFHE.Expand}, \mathsf{MFHE.Eval}, \mathsf{MFHE.Dec})$ be a classical multi-key fully homomorphic encryption scheme for 1-bit messages with decryption circuit of depth $d(\lambda, N)$, expanded ciphertexts of length $p(\lambda, N)$ and secret keys of length $s(\lambda, N)$. Also, let $\mathsf{COQT} = (\mathsf{Encode}, \mathsf{Apply}, \mathsf{Decode})$ be a conditional oblivious quantum transform for circuit class $\mathcal{C}_d$ (i.e., the class of depth $d(\lambda, N)$ circuits) and gate set $\{\mathsf{P}^\dagger\}$. Below we describe our scheme QMPLHE. For notational convenience, let $p = p(\lambda, N), d = d(\lambda, N)$ and $s = s(\lambda, N)$.

- $\mathsf{Setup}(1^\lambda, 1^k)$: The setup algorithm takes as input the security parameter λ and T-gate bound k. It runs $\mathsf{MFHE.Setup}$ to generate public parameters as $\mathsf{params} \leftarrow \mathsf{MFHE.Setup}(1^\lambda)$, and outputs $(\mathsf{params}, 1^k)$ as the public parameters.
- $\mathsf{KeyGen}((\mathsf{params}, 1^k), N, \mathsf{pos})$: The key generation algorithm takes as input the public parameters $(\mathsf{params}, 1^k)$, number of users N and position pos. The key generation proceeds as follows:
  1. First, it generates $k + 1$ classical MFHE key tuples as $(\mathsf{pk}_i, \mathsf{ek}_i, \mathsf{sk}_i) \leftarrow \mathsf{MFHE.KeyGen}(\mathsf{params})$ for $i \leq k + 1$.
  2. Consider set intervals $S_j = \{s \cdot (j-1) + 1, \ldots, s \cdot j\}$ for $j \leq N$, and $S_{N+1} = \{s \cdot N + 1, \ldots, s \cdot N + p\}$. Let $\mathbf{S} = (S_1, \ldots, S_{N+1})$. It computes k COQTs as follows:
  $$(\sigma_i, \mathsf{key}_i) \leftarrow \mathsf{Encode}(\mathsf{MFHE.Dec}, \mathbf{S}, \mathsf{pos}, \mathsf{sk}_i, \mathsf{P}^\dagger) \text{ for } i \leq k.$$
  3. Next, it encrypts the i^{th} classical decoding keys key_i and the i^{th} MFHE secret keys sk_i under the $(i+1)^{th}$ MFHE public key pk_{i+1} as[26]
  $$\mathsf{ct}_{\mathsf{key},i} \leftarrow \mathsf{MFHE.Enc}(\mathsf{pk}_{i+1}, \mathsf{key}_i), \quad \mathsf{ct}_{\mathsf{sk},i} \leftarrow \mathsf{MFHE.Enc}(\mathsf{pk}_{i+1}, \mathsf{sk}_i) \text{ for } i \leq k.$$
  4. Let $\rho_1 = \rho(\mathsf{ct}_{\mathsf{key},1}, \ldots, \mathsf{ct}_{\mathsf{key},k})$, $\rho_2 = \rho(\mathsf{ct}_{\mathsf{sk},1}, \ldots, \mathsf{ct}_{\mathsf{sk},k})$ and $\rho_3 = \rho(\mathsf{ek}_1, \ldots, \mathsf{ek}_{k+1})$. Finally, it outputs the key tuple as
  $$\mathsf{pk} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_{k+1}), \quad \mathsf{sk} = (\mathsf{sk}_1, \ldots, \mathsf{sk}_{k+1})$$
  $$\rho_{\mathsf{ek}} = \rho_1 \otimes \rho_2 \otimes \rho_3 \otimes \sigma_1 \otimes \cdots \otimes \sigma_k.$$

[25]By error indicator bit we mean bit a which denotes whether the encrypted state has X-gate applied or not. Note that P error gets introduced only if X was present.

[26]Note that MFHE scheme supports bit encryption. Therefore, to encrypt multi-bit messages, the $\mathsf{MFHE.Enc}$ algorithm will be run independently on each message bit. However, for notational convenience throughout this section as well as rest of the paper, we overload the notation and use $\mathsf{MFHE.Enc}$, $\mathsf{MFHE.Expand}$ and $\mathsf{MFHE.Dec}$ algorithms to encrypt, expand and decrypt multi-bit messages respectively.

- $\mathsf{Enc}(\mathsf{pk}, \rho)$: The encryption algorithm takes as input a public key pk and a single-qubit ρ.[27] Let $\mathsf{pk} = \left(\mathsf{pk}_1, \ldots, \mathsf{pk}_{k+1}\right)$. It chooses bits a, b uniformly at random, encrypts them under pk_1 and computes a quantum one-time pad of ρ using a, b. Concretely, it outputs a cipherstate σ in the following classical-quantum state
$$\frac{1}{4} \sum_{a,b \in \{0,1\}} \rho(\mathsf{MFHE.Enc}(\mathsf{pk}_1, (a,b))) \otimes (\mathsf{X}^a\mathsf{Z}^b)\rho(\mathsf{X}^a\mathsf{Z}^b)^\dagger.$$

- $\mathsf{Expand}((\mathsf{pk}^{(1)}, \ldots, \mathsf{pk}^{(N)}), i, \sigma)$: The expansion algorithm takes as input N public keys $\mathsf{pk}^{(j)}$ and a single cipherstate σ encrypted under i^{th} public key. Let $\mathsf{pk}^{(j)} = (\mathsf{pk}^{(j)}_1, \ldots, \mathsf{pk}^{(j)}_{k+1})$ for $j \leq N$, and σ is a classical-quantum state with the classical component being a MFHE ciphertext ct and quantum component being a single-qubit σ_{ct}.

  It runs the MFHE expansion algorithm on ciphertext ct to compute expanded ciphertext as $\widehat{\mathsf{ct}} \leftarrow \mathsf{MFHE.Expand}((\mathsf{pk}^{(1)}_1, \ldots, \mathsf{pk}^{(N)}_1), i, \mathsf{ct})$, and outputs $\rho(\widehat{\mathsf{ct}}) \otimes \sigma_{\mathsf{ct}}$ as the expanded cipherstate.

- $\mathsf{Eval}(C, (\rho^{(1)}_{\mathsf{ek}}, \ldots, \rho^{(N)}_{\mathsf{ek}}), \widehat{\sigma})$: The evaluation algorithm takes as input a circuit C, a tuple of N evaluation keys for each position and expanded cipherstate $\widehat{\sigma}$. Let n be the number of input wires to circuit C and t be the number of T-gates in it.

  Let $\rho^{(i)}_{\mathsf{ek}} = \rho^{(i)}_1 \otimes \rho^{(i)}_2 \otimes \rho^{(i)}_3 \otimes \sigma^{(i)}_1 \otimes \cdots \otimes \sigma^{(i)}_k$ for $i \leq N$. Also, let $\rho^{(i)}_1 = \rho(\mathsf{ct}^{(i)}_{\mathsf{key},1}, \ldots, \mathsf{ct}^{(i)}_{\mathsf{key},k})$, $\rho^{(i)}_2 = \rho(\mathsf{ct}^{(i)}_{\mathsf{sk},1}, \ldots, \mathsf{ct}^{(i)}_{\mathsf{sk},k})$ and $\rho^{(i)}_3 = \rho(\mathsf{ek}^{(i)}_1, \ldots, \mathsf{ek}^{(i)}_{k+1})$. The evaluation algorithm first expands the key and sk ciphertexts for all k levels and N users as
$$\widehat{\mathsf{ct}}^{(i)}_{\mathsf{key},j} \leftarrow \mathsf{MFHE.Expand}((\mathsf{pk}^{(1)}_{j+1}, \ldots, \mathsf{pk}^{(N)}_{j+1}), i, \mathsf{ct}^{(i)}_{\mathsf{key},j}), \quad \widehat{\mathsf{ct}}^{(i)}_{\mathsf{sk},j} \leftarrow \mathsf{MFHE.Expand}((\mathsf{pk}^{(1)}_{j+1}, \ldots, \mathsf{pk}^{(N)}_{j+1}), i, \mathsf{ct}^{(i)}_{\mathsf{sk},j})$$

  The expanded cipherstate $\widehat{\sigma}$ consists of an n-fold state $\widehat{\sigma}_{\mathsf{ct}}$ and n MFHE expanded ciphertexts $\left\{\widehat{\mathsf{ct}}_w\right\}_{w=1}^n$, where $\widehat{\mathsf{ct}}_w$ is the ciphertext corresponding to wire w. Let $\widehat{\sigma}_w$ denote the qubit at wire w.

  Let ℓ denote the current MFHE ciphertext level. Each T-gate evaluation increases ℓ by 1. The evaluation algorithm starts with $\ell = 1$ and performs evaluation (gate-by-gate) as follows:

  1. **P-Gate on wire w:** Apply P-gate to $\widehat{\sigma}_w$ (i.e., qubit at wire w). Let f_{P} denote the following circuit on two bits — $f_{\mathsf{P}}(a, b) = (a, a \oplus b)$. Update $\widehat{\mathsf{ct}}_w$ as $\widehat{\mathsf{ct}}_w \leftarrow \mathsf{MFHE.Eval}(f_{\mathsf{P}}, (\mathsf{ek}^{(1)}_\ell, \ldots, \mathsf{ek}^{(N)}_\ell), \widehat{\mathsf{ct}}_w)$.
  2. **H-Gate on wire w:** Apply H-gate to $\widehat{\sigma}_w$. Let f_{H} denote the following circuit on two bits — $f_{\mathsf{H}}(a, b) = (b, a)$. Update $\widehat{\mathsf{ct}}_w$ as $\widehat{\mathsf{ct}}_w \leftarrow \mathsf{MFHE.Eval}(f_{\mathsf{H}}, (\mathsf{ek}^{(1)}_\ell, \ldots, \mathsf{ek}^{(N)}_\ell), \widehat{\mathsf{ct}}_w)$.
  3. **CNOT-Gate on wires v, w:** Apply CNOT-gate to $\widehat{\sigma}_v, \widehat{\sigma}_w$. Let f^v_{CNOT} and f^w_{CNOT} denote the following circuits on four bits — $f^v_{\mathsf{CNOT}}(a, b, c, d) = (a, b \oplus d)$, $f^w_{\mathsf{CNOT}}(a, b, c, d) = (a \oplus c, d)$. Update $\widehat{\mathsf{ct}}_v, \widehat{\mathsf{ct}}_w$ as
$$\widehat{\mathsf{ct}}_v \leftarrow \mathsf{MFHE.Eval}(f^v_{\mathsf{CNOT}}, (\mathsf{ek}^{(1)}_\ell, \ldots, \mathsf{ek}^{(N)}_\ell), (\widehat{\mathsf{ct}}_v, \widehat{\mathsf{ct}}_w)),$$
$$\widehat{\mathsf{ct}}_w \leftarrow \mathsf{MFHE.Eval}(f^w_{\mathsf{CNOT}}, (\mathsf{ek}^{(1)}_\ell, \ldots, \mathsf{ek}^{(N)}_\ell), (\widehat{\mathsf{ct}}_v, \widehat{\mathsf{ct}}_w)).$$
  4. **T-gate on wire w:** Apply T-gate to $\widehat{\sigma}_w$. Note that $\widehat{\mathsf{ct}}_w$ is an encryption of two classical bits (say a, b). Let $\widehat{\mathsf{ct}}_{w,1}$ denote the expanded ciphertext of first bit (i.e., a).[28]

     Consider set intervals $S_i = \{s \cdot (i-1) + 1, \ldots, s \cdot i\}$ for $i \leq N$, and $S_{N+1} = \{s \cdot N + 1, \ldots, s \cdot N + p\}$. Let $\mathbf{S} = (S_1, \ldots, S_{N+1})$. It applies the COQTs on $\widehat{\sigma}_w$ as follows
$$(\widehat{\sigma}_w, \mathsf{aux}) \leftarrow \mathsf{Apply}(\mathsf{MFHE.Dec}, \mathbf{S}, \sigma^{(1)}_\ell, \ldots, \sigma^{(N)}_\ell, \widehat{\mathsf{ct}}_{w,1}, \widehat{\sigma}_w).$$

[27] Multi-qubits could be encrypted analogously.

[28] Recall that the MFHE scheme supports bit encryption, therefore each ciphertext only encrypts one-bit. Thus, each $\widehat{\mathsf{ct}}_w$ consists of two ciphertexts $\widehat{\mathsf{ct}}_{w,1}, \widehat{\mathsf{ct}}_{w,2}$.

Next, it recrypts the current one-time pad keys from level ℓ to $\ell+1$ for all wires $v \leq n$. Let f^v_{Dec} be the $\mathsf{MFHE.Dec}$ circuit with ciphertext $\widehat{\mathsf{ct}}_v$ hardwired. Concretely, $f^v_{\mathsf{Dec}}(x_1,\ldots,x_N) = \mathsf{MFHE.Dec}(x_1,\ldots,x_N,\widehat{\mathsf{ct}}_v)$. The evaluator recrypts $\widehat{\mathsf{ct}}_v$ (for all $v \leq n$) as follows

$$\widehat{\mathsf{ct}}_v \leftarrow \mathsf{MFHE.Eval}(f^v_{\mathsf{Dec}}, (\mathsf{ek}^{(1)}_{\ell+1},\ldots,\mathsf{ek}^{(N)}_{\ell+1}), (\widehat{\mathsf{ct}}^{(1)}_{\mathsf{sk},\ell},\ldots,\widehat{\mathsf{ct}}^{(N)}_{\mathsf{sk},\ell})).$$

Let $f^{\mathsf{aux}}_{\mathsf{Decode}}$ be the Decode circuit with auxiliary information aux hardwired. Concretely, $f^{\mathsf{aux}}_{\mathsf{Decode}}(x_1,\ldots,x_N) = \mathsf{Decode}(x_1,\ldots,x_N,\mathsf{aux})$. Next, it generates key updates for wire w as

$$\widehat{\mathsf{ct}}'_w \leftarrow \mathsf{MFHE.Eval}(f^{\mathsf{aux}}_{\mathsf{Decode}}, (\mathsf{ek}^{(1)}_{\ell+1},\ldots,\mathsf{ek}^{(N)}_{\ell+1}), (\widehat{\mathsf{ct}}^{(1)}_{\mathsf{key},\ell},\ldots,\widehat{\mathsf{ct}}^{(N)}_{\mathsf{key},\ell})).$$

Also, let $f_\oplus$ denote the following circuit on four bits — $f_\oplus(a,b,c,d) = (a\oplus c, b\oplus d)$. The evaluation algorithm updates the wire key as

$$\widehat{\mathsf{ct}}_w \leftarrow \mathsf{MFHE.Eval}(f_\oplus, (\mathsf{ek}^{(1)}_{\ell+1},\ldots,\mathsf{ek}^{(N)}_{\ell+1}), (\widehat{\mathsf{ct}}_w, \widehat{\mathsf{ct}}'_w)).$$

Finally, it increases the MFHE ciphertext level (i.e., sets $\ell = \ell+1$).

- $\mathsf{Dec}(\mathsf{sk}^{(1)},\ldots,\mathsf{sk}^{(N)},\widehat{\sigma})$: Let $\mathsf{sk}^{(i)} = (\mathsf{sk}^{(i)}_1,\ldots,\mathsf{sk}^{(i)}_{k+1})$. The decryption algorithm takes as input N secret keys and a single (expanded) cipherstate $\widehat{\sigma} = \rho(\widehat{\mathsf{ct}}) \otimes \widehat{\sigma}_{\mathsf{ct}}$. Let $\widehat{\mathsf{ct}}$ be a ciphertext under level ℓ public keys.[29] It decrypts $\widehat{\mathsf{ct}}$ using level ℓ keys as $(a,b) \leftarrow \mathsf{MFHE.Dec}(\mathsf{sk}^{(1)}_\ell,\ldots,\mathsf{sk}^{(N)}_\ell,\widehat{\mathsf{ct}})$. Finally, it performs one-time decryption using keys a,b and outputs

$$(\mathsf{X}^a\mathsf{Z}^b)\ \widehat{\sigma}_{\mathsf{ct}}\ (\mathsf{X}^a\mathsf{Z}^b)^\dagger.$$

5.2 Correctness

We will prove that the quantum multi-key positional homomorphic encryption scheme described above satisfies the correctness property. The proof is divided in two parts where we prove correctness of expansion as well as homomorphic evaluation of arbitrary quantum circuits with at most k T-gates. For simplicity of notation, we only look at the message registers and ignore the auxiliary registers for the purposes of the correctness proof. Also, throughout this section whenever we talk about equality on quantum registers, then we simply mean that the trace distance between the states in those registers is 0.

For any security parameter λ, T-gate bound k, consider public parameters $\mathsf{params} \leftarrow \mathsf{MFHE.Setup}(1^\lambda)$. For any number of users N, the key tuple for i^{th} user $(\mathsf{pk}^{(i)}, \rho^{(i)}_{\mathsf{ek}}, \mathsf{sk}^{(i)})$ is of the following form.

$$\mathsf{pk}^{(i)} = \left(\mathsf{pk}^{(i)}_1,\ldots,\mathsf{pk}^{(i)}_{k+1}\right), \quad \mathsf{sk} = \left(\mathsf{sk}^{(i)}_1,\ldots,\mathsf{sk}^{(i)}_{k+1}\right),$$
$$\rho^{(i)}_{\mathsf{ek}} = \rho^{(i)}_1 \otimes \rho^{(i)}_2 \otimes \rho^{(i)}_3 \otimes \sigma^{(i)}_1 \otimes \cdots \otimes \sigma^{(i)}_k,$$

where $\rho^{(i)}_1 = \rho(\mathsf{ct}^{(i)}_{\mathsf{key},1},\ldots,\mathsf{ct}^{(i)}_{\mathsf{key},k})$, $\rho^{(i)}_2 = \rho(\mathsf{ct}^{(i)}_{\mathsf{sk},1},\ldots,\mathsf{ct}^{(i)}_{\mathsf{sk},k})$ and $\rho^{(i)}_3 = \rho(\mathsf{ek}^{(i)}_1,\ldots,\mathsf{ek}^{(i)}_{k+1})$, and MFHE keys are generated as $(\mathsf{pk}^{(i)}_j, \mathsf{ek}^{(i)}_j, \mathsf{sk}^{(i)}_j) \leftarrow \mathsf{MFHE.KeyGen}(\mathsf{params})$, and $(\sigma^{(i)}_j, \mathsf{key}^{(i)}_j) \leftarrow \mathsf{Encode}(\mathsf{MFHE.Dec}, \mathbf{S}, i, \mathsf{sk}^{(i)}_j, \mathsf{P}^\dagger)$ for all $i \leq N, j \leq k+1$. Also, ciphertexts $\mathsf{ct}^{(i)}_{\mathsf{key},j}, \mathsf{ct}^{(i)}_{\mathsf{sk},j}$ are generated as $\mathsf{ct}^{(i)}_{\mathsf{key},j} \leftarrow \mathsf{MFHE.Enc}(\mathsf{pk}^{(i)}_{j+1}, \mathsf{key}^{(i)}_j)$, $\mathsf{ct}^{(i)}_{\mathsf{sk},j} \leftarrow \mathsf{MFHE.Enc}(\mathsf{pk}^{(i)}_{j+1}, \mathsf{sk}^{(i)}_j)$.

Expansion. Consider any single-qubit state $\rho^{\mathcal{M}}$ and any index $i \leq N$. Encryption of $\rho^{\mathcal{M}}$ under $\mathsf{pk}^{(i)}$ will be of the form $\sigma = \rho(\mathsf{ct}) \otimes \sigma_{\mathsf{ct}}$, where $\mathsf{ct} \leftarrow \mathsf{MFHE.Enc}(\mathsf{pk}^{(i)}_1, (a,b))$ and $\sigma_{\mathsf{ct}} = (\mathsf{X}^a\mathsf{Z}^b)\rho^{\mathcal{M}}(\mathsf{X}^a\mathsf{Z}^b)^\dagger$ for some random bits a,b. Note that the expansion algorithm simply expands the classical ciphertext

[29] Note that the expanded ciphertexts might not be encrypted under top level keys. The MFHE ciphertext component in expanded cipherstates (i.e., just after expansion) will be at level 1. And, in evaluated cipherstates, MFHE ciphertexts will be at level t, where t is the number of T-gates in circuit C being evaluated.

$\mathsf{ct} = \mathsf{MFHE.Enc}(\mathsf{pk}_1^{(i)}, (a,b))$ as $\widehat{\mathsf{ct}} \leftarrow \mathsf{MFHE.Expand}((\mathsf{pk}_1^{(1)}, \dots, \mathsf{pk}_1^{(N)}), i, \mathsf{ct})$. The corresponding expanded ciphertext will be $\widehat{\sigma} = \rho(\widehat{\mathsf{ct}}) \otimes \sigma_{\mathsf{ct}}$.

From correctness of MFHE scheme, we know that $\mathsf{MFHE.Dec}(\mathsf{sk}_1^{(1)}, \dots, \mathsf{sk}_1^{(N)}, \widehat{\mathsf{ct}}) = (a,b)$. Also, we know that $(\mathsf{X}^a\mathsf{Z}^b)\sigma_{\mathsf{ct}}(\mathsf{X}^a\mathsf{Z}^b)^\dagger = \rho^{\mathcal{M}}$. Therefore, $\mathsf{Dec}(\mathsf{sk}^{(1)}, \dots, \mathsf{sk}^{(N)}, \widehat{\sigma}) = \rho^{\mathcal{M}}$. As stated before, here by equality we mean that the trace distance between the associated states is zero. This concludes proof of correct expansion.

Evaluation. Since we have proven correctness of expansion, for proving correct evaluation we only need to show that given an expanded cipherstate $\widehat{\sigma}$, the evaluation algorithm preserves correctness at each step. The proof follows from an inductive argument. Below we prove that for every $C \in \{\mathsf{P}, \mathsf{H}, \mathsf{CNOT}, \mathsf{T}\}$, the evaluated cipherstate correctly decrypts.

Case 1 ($C \in \{\mathsf{P}, \mathsf{H}, \mathsf{CNOT}\}$) : The proof of correctness is similar to that of CL scheme of [BJ15]. First, we argue correctness if $C \in \{\mathsf{P}, \mathsf{H}\}$ (i.e., a single-qubit gate).

Let $\widehat{\sigma} = \rho(\widehat{\mathsf{ct}}) \otimes \widehat{\sigma}_{\mathsf{ct}}$ be an expanded cipherstate encrypting a single qubit $\rho^{\mathcal{M}}$ under level ℓ keys. For ease of exposition, consider that $\rho^{\mathcal{M}} = |\psi\rangle\langle\psi|$ (i.e., a single-qubit pure state $|\psi\rangle$). Let $\mathsf{MFHE.Dec}(\mathsf{sk}_\ell^{(1)}, \dots, \mathsf{sk}_\ell^{(N)}, \widehat{\mathsf{ct}}) = (a,b)$ for some bits a, b. Thus, we know that $\widehat{\sigma}_{\mathsf{ct}} = (\mathsf{X}^a\mathsf{Z}^b)\,|\psi\rangle\langle\psi|\,(\mathsf{X}^a\mathsf{Z}^b)^\dagger$. Note that the evaluation algorithm simply applies the gate C to $\widehat{\sigma}_{\mathsf{ct}}$. This can be written as

$$C\mathsf{X}^a\mathsf{Z}^b\,|\psi\rangle = \begin{cases} \mathsf{X}^a\mathsf{Z}^{a\oplus b}C\,|\psi\rangle & \text{if } C = \mathsf{P} \\ \mathsf{X}^b\mathsf{Z}^a C\,|\psi\rangle & \text{if } C = \mathsf{H}. \end{cases}$$

Next, it updates $\widehat{\mathsf{ct}}$ as $\widehat{\mathsf{ct}} = \mathsf{MFHE.Eval}(f_C, (\mathsf{ek}_\ell^{(1)}, \dots, \mathsf{ek}_\ell^{(N)}), \widehat{\mathsf{ct}})$. Since $\widehat{\mathsf{ct}}$ encrypted (a,b) before evaluation. After evaluation, $\widehat{\mathsf{ct}}$ will encrypt $f_C(a,b)$. This follows from correctness of MFHE scheme. Thus, we could write that

$$\mathsf{MFHE.Dec}(\mathsf{sk}_\ell^{(1)}, \dots, \mathsf{sk}_\ell^{(N)}, \widehat{\mathsf{ct}}) = \begin{cases} (a, a\oplus b) & \text{if } C = \mathsf{P} \\ (b,a) & \text{if } C = \mathsf{H}. \end{cases}$$

Therefore, if $C \in \{\mathsf{P}, \mathsf{H}\}$, then evaluated cipherstates correctly decrypt. Next, we argue correctness in the case $C = \mathsf{CNOT}$. Let $\widehat{\sigma} = \rho(\widehat{\mathsf{ct}}_1, \widehat{\mathsf{ct}}_2) \otimes \widehat{\sigma}_{\mathsf{ct}}$ be an expanded cipherstate encrypting two qubits. Let $\mathsf{MFHE.Dec}(\mathsf{sk}_\ell^{(1)}, \dots, \mathsf{sk}_\ell^{(N)}, \widehat{\mathsf{ct}}_i) = (a_i, b_i)$ for some bits a_1, b_1, a_2, b_2. As before assume that $\widehat{\sigma}_{\mathsf{ct}}$ is in pure state, so $\widehat{\sigma}_{\mathsf{ct}} = (\mathsf{X}^{a_1}\mathsf{Z}^{b_1} \otimes \mathsf{X}^{a_2}\mathsf{Z}^{b_2})\,|\psi\rangle\langle\psi|\,(\mathsf{X}^{a_1}\mathsf{Z}^{b_1} \otimes \mathsf{X}^{a_2}\mathsf{Z}^{b_2})^\dagger$. Thus,

$$C(\mathsf{X}^{a_1}\mathsf{Z}^{b_1} \otimes \mathsf{X}^{a_2}\mathsf{Z}^{b_2})\,|\psi\rangle = (\mathsf{X}^{a_1\oplus a_2}\mathsf{Z}^{b_1} \otimes \mathsf{X}^{a_2}\mathsf{Z}^{b_1\oplus b_2})C\,|\psi\rangle\,.$$

Similarly, the correctness of key update follows from the correctness of MFHE scheme. This concludes the proof of evaluation correctness if $C \in \{\mathsf{P}, \mathsf{H}, \mathsf{CNOT}\}$.

Case 2 ($C = \mathsf{T}$) : Let $\widehat{\sigma} = \rho(\widehat{\mathsf{ct}}) \otimes \widehat{\sigma}_{\mathsf{ct}}$ be an expanded cipherstate encrypting a single qubit under level ℓ keys, such that $\mathsf{MFHE.Dec}(\mathsf{sk}_\ell^{(1)}, \dots, \mathsf{sk}_\ell^{(N)}, \widehat{\mathsf{ct}}) = (a,b)$ for some bits a, b, and $\widehat{\sigma}_{\mathsf{ct}} = (\mathsf{X}^a\mathsf{Z}^b)\rho^{\mathcal{M}}(\mathsf{X}^a\mathsf{Z}^b)^\dagger$ for some quantum state $\rho^{\mathcal{M}}$. Also, let $\widehat{\mathsf{ct}} = (\widehat{\mathsf{ct}}_a, \widehat{\mathsf{ct}}_b)$, where $\widehat{\mathsf{ct}}_a$ and $\widehat{\mathsf{ct}}_b$ encrypt a and b, respectively.[30]

The evaluation algorithm starts by applying T-gate to the state $\widehat{\sigma}_{\mathsf{ct}}$. This updates the state $\widehat{\sigma}_{\mathsf{ct}}$ to

$$(\mathsf{P}^a\mathsf{X}^a\mathsf{Z}^b\mathsf{T})\rho^{\mathcal{M}}(\mathsf{P}^a\mathsf{X}^a\mathsf{Z}^b\mathsf{T})^\dagger.$$

Let $\boldsymbol{\sigma}_\ell = (\sigma_\ell^{(1)}, \dots, \sigma_\ell^{(N)})$, $\mathbf{ek}_\ell = (\mathsf{ek}_\ell^{(1)}, \dots, \mathsf{ek}_\ell^{(N)})$ and $\mathbf{sk}_\ell = (\mathsf{sk}_\ell^{(1)}, \dots, \mathsf{sk}_\ell^{(N)})$. Next, the evaluator applies COQTs on $\widehat{\sigma}_{\mathsf{ct}}$ as

$$(\widehat{\sigma}_{\mathsf{ct}}, \mathsf{aux}) \leftarrow \mathsf{Apply}(\mathsf{MFHE.Dec}, \mathbf{S}, \boldsymbol{\sigma}_\ell, \widehat{\mathsf{ct}}_a, \widehat{\sigma}_{\mathsf{ct}}).$$

And, it homomorphically runs the decoding algorithm as

$$\widehat{\mathsf{ct}}' \leftarrow \mathsf{MFHE.Eval}(f_{\mathsf{Decode}}^{\mathsf{aux}}, \mathbf{ek}_{\ell+1}, (\widehat{\mathsf{ct}}_{\mathsf{key},\ell}^{(1)}, \dots, \widehat{\mathsf{ct}}_{\mathsf{key},\ell}^{(N)})).$$

[30]Recall MFHE scheme encrypts bit-by-bit.

First, note that by correctness of MFHE expansion $\widehat{\mathsf{ct}}^{(i)}_{\mathsf{key},\ell}$ is an encryption of $\mathsf{key}^{(i)}_{\ell}$. Also, by correctness of MFHE evaluation, we can claim that $\widehat{\mathsf{ct}}'$ is an MFHE encryption of $(a', b') = \mathsf{Decode}(\mathsf{key}^{(1)}_{\ell}, \ldots, \mathsf{key}^{(N)}_{\ell}, \mathsf{aux})$ under level $\ell + 1$ public keys. Let $\alpha = \mathsf{MFHE.Dec}(\mathbf{sk}_{\ell}, \widehat{\mathsf{ct}}_a)$. By correctness of MFHE decryption, we know that $\alpha = a$. Therefore, we can claim by correctness of COQTs that $\widehat{\sigma}_{\mathsf{ct}}$ is transformed as follows:

$$\begin{aligned}\widehat{\sigma}_{\mathsf{ct}} &= \left(\mathsf{X}^{a'}\mathsf{Z}^{b'}(\mathsf{P}^{\dagger})^{\alpha}\mathsf{P}^{a}\mathsf{X}^{a}\mathsf{Z}^{b}\mathsf{T}\right)\rho^{\mathcal{M}}\left(\mathsf{X}^{a'}\mathsf{Z}^{b'}(\mathsf{P}^{\dagger})^{\alpha}\mathsf{P}^{a}\mathsf{X}^{a}\mathsf{Z}^{b}\mathsf{T}\right)^{\dagger}\\ &= \left(\mathsf{X}^{a'}\mathsf{Z}^{b'}\mathsf{X}^{a}\mathsf{Z}^{b}\mathsf{T}\right)\rho^{\mathcal{M}}\left(\mathsf{X}^{a'}\mathsf{Z}^{b'}\mathsf{X}^{a}\mathsf{Z}^{b}\mathsf{T}\right)^{\dagger}\\ &= \left(\mathsf{X}^{a'\oplus a}\mathsf{Z}^{b'\oplus b}\mathsf{T}\right)\rho^{\mathcal{M}}\left(\mathsf{X}^{a'\oplus a}\mathsf{Z}^{b'\oplus b}\mathsf{T}\right)^{\dagger}\end{aligned}$$

So, $\widehat{\sigma}_{\mathsf{ct}}$ is a quantum one-time pad encryption of $\rho^{\mathcal{M}}$ with $a' \oplus a, b' \oplus b$ as secret keys. The evaluation algorithm also recrypts $\widehat{\mathsf{ct}}$ (i.e., the one-time pad keys a, b) from encryptions under level ℓ keys to level $\ell+1$. So, by correctness of decryption and evaluation, we know that recryption procedure is correct and thus $\widehat{\mathsf{ct}}$ is an encryption of (a, b) under level $\ell + 1$ keys.[31] Finally, the evaluator performs "xor" on ciphertexts $\widehat{\mathsf{ct}}$ and $\widehat{\mathsf{ct}}'$ as follows

$$\widehat{\mathsf{ct}} \leftarrow \mathsf{MFHE.Eval}(f_{\oplus}, \mathbf{ek}_{\ell+1}, (\widehat{\mathsf{ct}}, \widehat{\mathsf{ct}}')).$$

By correctness of MFHE evaluation, we know that $\widehat{\mathsf{ct}}$ now encrypts $(a' \oplus a, b' \oplus b)$ under level $\ell + 1$ public keys. Therefore, the decryption algorithm will first compute the keys $(a' \oplus a, b' \oplus b)$, and then perform one-time pad decryption to correctly recover message state $\rho^{\mathcal{M}}$. This conclude the proof of correctness.

5.3 Security

We will now show that the scheme described above is q-IND-CPA secure as per Definition 3.3. Formally, we prove the following.

Theorem 5.1. If $\mathsf{MFHE} = (\mathsf{MFHE.Setup}, \mathsf{MFHE.KeyGen}, \mathsf{MFHE.Enc}, \mathsf{MFHE.Expand}, \mathsf{MFHE.Eval}, \mathsf{MFHE.Dec})$ is a q-IND-CPA secure multi-key fully homomorphic encryption scheme for 1-bit messages satisfying Definition 3.1, and $\mathsf{COQT} = (\mathsf{Encode}, \mathsf{Apply}, \mathsf{Decode})$ is a secure conditional oblivious quantum transform for circuit class $\mathcal{C}_d$ and gate set $\{\mathsf{P}^{\dagger}\}$ satisfying Definition 4.1, then the scheme QMPLHE (described in Section 5.1) is a q-IND-CPA secure quantum multi-key positional leveled homomorphic encryption scheme for quantum circuits with polynomially bounded T-gates as per Definition 3.3.

Our proof proceeds via a sequence of hybrid games. Each game is played between the challenger and attacker $\mathcal{A}$. Let $\mathcal{A}$ be any quantum PPT adversary that wins the q-IND-CPA game with non-negligible advantage. We argue that such an adversary must break security of at least one underlying primitive. The first game corresponds to the q-IND-CPA game as described in Definition 3.3. In the final game, the challenge cipherstate does not contain any information about the challenge message state. We would like point out that in the hybrid games the adversary is given public-evaluation key pair for N users, position $\mathsf{pos} \leq N$ and T-gate bound k. And, the indistinguishability proofs hold irrespective of the choice of N, pos and k.

The high level proof idea is as follows. First, note that the i^{th} COQT depends on key sk_i, and its decoding key key_i as well as sk_i is encrypted under public key pk_{i+1}. That is, level i decoding key and secret key in encrypted under level $i + 1$ key. Now, the adversary has no information about the secret key for level $k + 1$ (i.e., last level). Thus, using q-IND-CPA security of underlying MFHE scheme, we could switch the corresponding ciphertexts to encryptions of all-zeros strings, thereby removing the decoding key of k^{th} COQT. Next, using COQT security, we could switch to an encoding of all-zeros string instead of the k^{th} secret key. This removes the information about k^{th} secret key completely. Proceeding this way, we could switch all COQTs to be empty transforms and all ciphertexts to encrypt to all-zeros strings. Finally,

[31] Actually the evaluator recrypts all the wire keys at this point. The correctness of all recryptions is guaranteed by correctness of underlying MFHE scheme.

we could switch the encryption of one-time pad keys (in the challenge cipherstate) to encryption of zeros, thereby reducing it to the security of quantum one-time pad. Below we describe the proof in detail.

Throughout the hybrids, the set intervals $\mathbf{S} = (S_1, \ldots, S_{N+1})$ are defined as in the construction, i.e. $S_i = \{s \cdot (i-1)+1, \ldots, s \cdot i\}$ for $i \leq N$ and $S_{N+1} = \{s \cdot N + 1, \ldots, s \cdot N + p\}$. Also, we will use $\mathbf{0}$ to denote the all-zeros string of appropriate length.

Game 1**:** This game is same as the original q-IND-CPA game.

1. **Setup Phase.** The challenger sets up by sampling the MFHE public parameters $\mathsf{params} \leftarrow \mathsf{MFHE.Setup}(1^\lambda)$. Next, it proceeds as follows:
   (a) It generates $k+1$ classical MFHE key tuples as $(\mathsf{pk}_i, \mathsf{ek}_i, \mathsf{sk}_i) \leftarrow \mathsf{MFHE.KeyGen}(\mathsf{params})$ for $i \leq k+1$.
   (b) It computes k COQTs as $(\sigma_i, \mathsf{key}_i) \leftarrow \mathsf{Encode}(\mathsf{MFHE.Dec}, \mathbf{S}, \mathsf{pos}, \mathsf{sk}_i, \mathsf{P}^\dagger)$ for $i \leq k$.
   (c) Next, it computes $2k$ MFHE ciphertexts $\mathsf{ct}_{\mathsf{key},i}, \mathsf{ct}_{\mathsf{sk},i}$ as $\mathsf{ct}_{\mathsf{key},i} \leftarrow \mathsf{MFHE.Enc}(\mathsf{pk}_{i+1}, \mathsf{key}_i)$, $\mathsf{ct}_{\mathsf{sk},i} \leftarrow \mathsf{MFHE.Enc}(\mathsf{pk}_{i+1}, \mathsf{sk}_i)$ for $i \leq k$.
   (d) Let $\rho_1 = \rho(\mathsf{ct}_{\mathsf{key},1}, \ldots, \mathsf{ct}_{\mathsf{key},k})$, $\rho_2 = \rho(\mathsf{ct}_{\mathsf{sk},1}, \ldots, \mathsf{ct}_{\mathsf{sk},k})$ and $\rho_3 = \rho(\mathsf{ek}_1, \ldots, \mathsf{ek}_{k+1})$. Finally, it sends the public parameters, public key and evaluation key to $\mathcal{A}$ as

$$\mathsf{params}' = (\mathsf{params}, 1^k), \quad \mathsf{pk} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_{k+1})$$
$$\rho_{\mathsf{ek}} = \rho_1 \otimes \rho_2 \otimes \rho_3 \otimes \sigma_1 \otimes \cdots \otimes \sigma_k.$$

2. **Challenge.** Next, $\mathcal{A}$ sends the challenge message register $\mathcal{R}^{\mathcal{M}}$ to the challenger.

   The challenger chooses a random bit $\beta \leftarrow \{0,1\}$. If $\beta = 0$, it erases the message state in register $\mathcal{R}^{\mathcal{M}}$, otherwise the register is unchanged. That is, if $\beta = 0$ then the register $\mathcal{R}^{\mathcal{M}}$ contains state $|0\rangle\langle 0|$, otherwise it contains some message state $\rho^{\mathcal{M}}$. It chooses random bits $a, b \leftarrow \{0,1\}$, and applies a quantum one-time pad on register $\mathcal{R}^{\mathcal{M}}$ using a, b. It also encrypts (a,b) under key pk_1 as $\mathsf{ct} \leftarrow \mathsf{MFHE.Enc}(\mathsf{pk}_1, (a,b))$. Finally, it sends the encrypted register $\mathcal{R}^{\mathcal{M}}$ and ciphertext ct to the adversary $\mathcal{A}$.

3. **Guess.** $\mathcal{A}$ outputs it guess β' and wins if $\beta' = \beta$.

We will now define $2k+1$ intermediate games — Game $(2, i^*, 0), (2, i^*, 1)$ for $i^* \in \{0, 1, \ldots, k-1\}$ and Game $(2, k, 0)$. The hybrid game $(2, 0, 0)$ will be same as Game 1. Also, hybrid game $(2, k, 0)$ will be same as Game 3.

Game $(2, i^*, 0)$**:** This is same as Game 1, except the challenger computes COQTs $(\sigma_i, \mathsf{key}_i)$ and ciphertexts $\mathsf{ct}_{\mathsf{key},i}, \mathsf{ct}_{\mathsf{sk},i}$ for top i^* levels (i.e., for $i > k - i^*$) as transforms of $\mathbf{0}$ instead of sk_i, and ciphertexts as encryptions of all-zeros string (respectively).

1. **Setup Phase.** The challenger sets up by sampling the MFHE public parameters $\mathsf{params} \leftarrow \mathsf{MFHE.Setup}(1^\lambda)$. Next, it proceeds as follows:
   (a) It generates $k+1$ classical MFHE key tuples as $(\mathsf{pk}_i, \mathsf{ek}_i, \mathsf{sk}_i) \leftarrow \mathsf{MFHE.KeyGen}(\mathsf{params})$ for $i \leq k+1$.
   (b) It computes k COQTs as $(\sigma_i, \mathsf{key}_i) \leftarrow \mathsf{Encode}(\mathsf{MFHE.Dec}, \mathbf{S}, \mathsf{pos}, \mathsf{sk}_i, \mathsf{P}^\dagger)$ for $i \leq k - i^*$, and $(\sigma_i, \mathsf{key}_i) \leftarrow \mathsf{Encode}(\mathsf{MFHE.Dec}, \mathbf{S}, \mathsf{pos}, \mathbf{0}, \mathsf{P}^\dagger)$ otherwise.
   (c) Next, it computes $2k$ MFHE ciphertexts $\mathsf{ct}_{\mathsf{key},i}, \mathsf{ct}_{\mathsf{sk},i}$ as $\mathsf{ct}_{\mathsf{key},i} \leftarrow \mathsf{MFHE.Enc}(\mathsf{pk}_{i+1}, \mathsf{key}_i)$, $\mathsf{ct}_{\mathsf{sk},i} \leftarrow \mathsf{MFHE.Enc}(\mathsf{pk}_{i+1}, \mathsf{sk}_i)$ for $i \leq k - i^*$, and $\mathsf{ct}_{\mathsf{key},i} \leftarrow \mathsf{MFHE.Enc}(\mathsf{pk}_{i+1}, \mathbf{0})$, $\mathsf{ct}_{\mathsf{sk},i} \leftarrow \mathsf{MFHE.Enc}(\mathsf{pk}_{i+1}, \mathbf{0})$ otherwise.

(d) Let $\rho_1 = \rho(\mathsf{ct}_{\mathsf{key},1}, \ldots, \mathsf{ct}_{\mathsf{key},k})$, $\rho_2 = \rho(\mathsf{ct}_{\mathsf{sk},1}, \ldots, \mathsf{ct}_{\mathsf{sk},k})$ and $\rho_3 = \rho(\mathsf{ek}_1, \ldots, \mathsf{ek}_{k+1})$. Finally, it sends the public parameters, public key and evaluation key to $\mathcal{A}$ as

$$\mathsf{params}' = (\mathsf{params}, 1^k), \quad \mathsf{pk} = \big(\mathsf{pk}_1, \ldots, \mathsf{pk}_{k+1}\big)$$
$$\rho_{\mathsf{ek}} = \rho_1 \otimes \rho_2 \otimes \rho_3 \otimes \sigma_1 \otimes \cdots \otimes \sigma_k.$$

Game $(2, i^*, 1)$: This is same as previous game, except the challenger computes ciphertexts $\mathsf{ct}_{\mathsf{key},k-i^*}, \mathsf{ct}_{\mathsf{sk},k-i^*}$ as encryptions of all-zeros strings.

1. **Setup Phase.** The challenger sets up by sampling the MFHE public parameters $\mathsf{params} \leftarrow \mathsf{MFHE.Setup}(1^\lambda)$. Next, it proceeds as follows:

   (a) It generates $k+1$ classical MFHE key tuples as $(\mathsf{pk}_i, \mathsf{ek}_i, \mathsf{sk}_i) \leftarrow \mathsf{MFHE.KeyGen}(\mathsf{params})$ for $i \leq k+1$.

   (b) It computes k COQTs as $(\sigma_i, \mathsf{key}_i) \leftarrow \mathsf{Encode}(\mathsf{MFHE.Dec}, \mathbf{S}, \mathsf{pos}, \mathsf{sk}_i, \mathsf{P}^\dagger)$ for $i \leq k - i^*$, and $(\sigma_i, \mathsf{key}_i) \leftarrow \mathsf{Encode}(\mathsf{MFHE.Dec}, \mathbf{S}, \mathsf{pos}, \mathbf{0}, \mathsf{P}^\dagger)$ otherwise.

   (c) Next, it computes $2k$ MFHE ciphertexts $\mathsf{ct}_{\mathsf{key},i}, \mathsf{ct}_{\mathsf{sk},i}$ as $\mathsf{ct}_{\mathsf{key},i} \leftarrow \mathsf{MFHE.Enc}(\mathsf{pk}_{i+1}, \mathsf{key}_i)$, $\mathsf{ct}_{\mathsf{sk},i} \leftarrow \mathsf{MFHE.Enc}(\mathsf{pk}_{i+1}, \mathsf{sk}_i)$ for $i \leq k-(i^*+1)$, and $\mathsf{ct}_{\mathsf{key},i} \leftarrow \mathsf{MFHE.Enc}(\mathsf{pk}_{i+1}, \mathbf{0})$, $\mathsf{ct}_{\mathsf{sk},i} \leftarrow \mathsf{MFHE.Enc}(\mathsf{pk}_{i+1}, \mathbf{0})$ otherwise.

   (d) Let $\rho_1 = \rho(\mathsf{ct}_{\mathsf{key},1}, \ldots, \mathsf{ct}_{\mathsf{key},k})$, $\rho_2 = \rho(\mathsf{ct}_{\mathsf{sk},1}, \ldots, \mathsf{ct}_{\mathsf{sk},k})$ and $\rho_3 = \rho(\mathsf{ek}_1, \ldots, \mathsf{ek}_{k+1})$. Finally, it sends the public parameters, public key and evaluation key to $\mathcal{A}$ as

$$\mathsf{params}' = (\mathsf{params}, 1^k), \quad \mathsf{pk} = \big(\mathsf{pk}_1, \ldots, \mathsf{pk}_{k+1}\big)$$
$$\rho_{\mathsf{ek}} = \rho_1 \otimes \rho_2 \otimes \rho_3 \otimes \sigma_1 \otimes \cdots \otimes \sigma_k.$$

Game 3: This is same as Game $(2, k, 0)$, i.e. the challenger computes all COQTs $(\sigma_i, \mathsf{key}_i)$ as transforms of $\mathbf{0}$ instead of sk_i, and all ciphertexts $\mathsf{ct}_{\mathsf{key},i}, \mathsf{ct}_{\mathsf{sk},i}$ as encryptions of all-zeros string.

1. **Setup Phase.** The challenger sets up by sampling the MFHE public parameters $\mathsf{params} \leftarrow \mathsf{MFHE.Setup}(1^\lambda)$. Next, it proceeds as follows:

   (a) It generates $k+1$ classical MFHE key tuples as $(\mathsf{pk}_i, \mathsf{ek}_i, \mathsf{sk}_i) \leftarrow \mathsf{MFHE.KeyGen}(\mathsf{params})$ for $i \leq k+1$.

   (b) It computes k COQTs as $(\sigma_i, \mathsf{key}_i) \leftarrow \mathsf{Encode}(\mathsf{MFHE.Dec}, \mathbf{S}, \mathsf{pos}, \mathbf{0}, \mathsf{P}^\dagger)$ for $i \leq k$.

   (c) Next, it computes $2k$ MFHE ciphertexts $\mathsf{ct}_{\mathsf{key},i}, \mathsf{ct}_{\mathsf{sk},i}$ as $\mathsf{ct}_{\mathsf{key},i} \leftarrow \mathsf{MFHE.Enc}(\mathsf{pk}_{i+1}, \mathbf{0})$, $\mathsf{ct}_{\mathsf{sk},i} \leftarrow \mathsf{MFHE.Enc}(\mathsf{pk}_{i+1}, \mathbf{0})$ for $i \leq k$.

   (d) Let $\rho_1 = \rho(\mathsf{ct}_{\mathsf{key},1}, \ldots, \mathsf{ct}_{\mathsf{key},k})$, $\rho_2 = \rho(\mathsf{ct}_{\mathsf{sk},1}, \ldots, \mathsf{ct}_{\mathsf{sk},k})$ and $\rho_3 = \rho(\mathsf{ek}_1, \ldots, \mathsf{ek}_{k+1})$. Finally, it sends the public parameters, public key and evaluation key to $\mathcal{A}$ as

$$\mathsf{params}' = (\mathsf{params}, 1^k), \quad \mathsf{pk} = \big(\mathsf{pk}_1, \ldots, \mathsf{pk}_{k+1}\big)$$
$$\rho_{\mathsf{ek}} = \rho_1 \otimes \rho_2 \otimes \rho_3 \otimes \sigma_1 \otimes \cdots \otimes \sigma_k.$$

Game 4: This is same as Game 3, except the challenger does not encrypt quantum one-time pad keys (a, b) under pk_1 in the challenge ciphertext. Instead it always encrypts $(0, 0)$.

1. **Setup Phase.** The challenger sets up by sampling the MFHE public parameters $\mathsf{params} \leftarrow \mathsf{MFHE.Setup}(1^\lambda)$. Next, it proceeds as follows:

   (a) It generates $k+1$ classical MFHE key tuples as $(\mathsf{pk}_i, \mathsf{ek}_i, \mathsf{sk}_i) \leftarrow \mathsf{MFHE.KeyGen}(\mathsf{params})$ for $i \leq k+1$.

   (b) It computes k COQTs as $(\sigma_i, \mathsf{key}_i) \leftarrow \mathsf{Encode}(\mathsf{MFHE.Dec}, \mathbf{S}, \mathsf{pos}, \mathbf{0}, \mathsf{P}^\dagger)$ for $i \leq k$.

(c) Next, it computes $2k$ MFHE ciphertexts $\mathsf{ct}_{\mathsf{key},i}, \mathsf{ct}_{\mathsf{sk},i}$ as $\mathsf{ct}_{\mathsf{key},i} \leftarrow \mathsf{MFHE.Enc}(\mathsf{pk}_{i+1}, \mathbf{0})$, $\mathsf{ct}_{\mathsf{sk},i} \leftarrow \mathsf{MFHE.Enc}(\mathsf{pk}_{i+1}, \mathbf{0})$ for $i \leq k$.

(d) Let $\rho_1 = \rho(\mathsf{ct}_{\mathsf{key},1}, \ldots, \mathsf{ct}_{\mathsf{key},k})$, $\rho_2 = \rho(\mathsf{ct}_{\mathsf{sk},1}, \ldots, \mathsf{ct}_{\mathsf{sk},k})$ and $\rho_3 = \rho(\mathsf{ek}_1, \ldots, \mathsf{ek}_{k+1})$. Finally, it sends the public parameters, public key and evaluation key to $\mathcal{A}$ as

$$\mathsf{params}' = (\mathsf{params}, 1^k), \quad \mathsf{pk} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_{k+1})$$
$$\rho_{\mathsf{ek}} = \rho_1 \otimes \rho_2 \otimes \rho_3 \otimes \sigma_1 \otimes \cdots \otimes \sigma_k.$$

2. **Challenge.** Next, $\mathcal{A}$ sends the challenge message register $\mathcal{R}^{\mathcal{M}}$ to the challenger.

The challenger chooses a random bit $\beta \leftarrow \{0,1\}$. If $\beta = 0$, it erases the message state in register $\mathcal{R}^{\mathcal{M}}$, otherwise the register is unchanged. That is, if $\beta = 0$ then the register $\mathcal{R}^{\mathcal{M}}$ contains state $|0\rangle\langle 0|$, otherwise it contains some message state $\rho^{\mathcal{M}}$. It chooses random bits $a, b \leftarrow \{0,1\}$, and applies a quantum one-time pad on register $\mathcal{R}^{\mathcal{M}}$ using a, b. It also encrypts $(0,0)$ under key pk_1 as $\mathsf{ct} \leftarrow \mathsf{MFHE.Enc}(\mathsf{pk}_1, (0,0))$. Finally, it sends the encrypted register $\mathcal{R}^{\mathcal{M}}$ and ciphertext ct to the adversary $\mathcal{A}$.

3. **Guess.** $\mathcal{A}$ outputs it guess β' and wins if $\beta' = \beta$.

5.3.1 Analysis

Let $\mathsf{Adv}^i_{\mathcal{A}} = |\Pr[\beta' = \beta] - 1/2|$ denote the advantage of adversary $\mathcal{A}$ in guessing the bit β in Game i. First, note that $|\mathsf{Adv}^1_{\mathcal{A}} - \mathsf{Adv}^{(2,0,0)}_{\mathcal{A}}| = 0$, i.e. $\mathcal{A}$'s advantage in distinguishing Games 1 and $(2,0,0)$ is 0. This is because they are identical games. Similarly, we have that $|\mathsf{Adv}^{(2,k,0)}_{\mathcal{A}} - \mathsf{Adv}^3_{\mathcal{A}}| = 0$.

To complete the proof, we establish via a sequence of lemmas that no quantum PPT adversary $\mathcal{A}$ can distinguish between each adjacent game with non-negligible probability. Below we discuss our lemmas in detail.

Lemma 5.1. If $\mathsf{MFHE} = (\mathsf{MFHE.Setup}, \mathsf{MFHE.KeyGen}, \mathsf{MFHE.Enc}, \mathsf{MFHE.Expand}, \mathsf{MFHE.Eval}, \mathsf{MFHE.Dec})$ is a q-IND-CPA secure multi-key fully homomorphic encryption scheme, then for all $i^* \in \{0, 1, \ldots, k-1\}$ and all quantum PPT adversaries $\mathcal{A}$, $|\mathsf{Adv}^{(2,i^*,0)}_{\mathcal{A}} - \mathsf{Adv}^{(2,i^*,1)}_{\mathcal{A}}|$ is negligible in the security parameter λ.

Proof. Suppose there exists an adversary $\mathcal{A}$ such that $|\mathsf{Adv}^{(2,i^*,0)}_{\mathcal{A}} - \mathsf{Adv}^{(2,i^*,1)}_{\mathcal{A}}|$ is non-negligible for some $i^* \in \{1, \ldots, k-1\}$. We construct an algorithm $\mathcal{B}$ that can distinguish encryptions of classical strings $\mathsf{key}_{k-i^*}, \mathsf{sk}_{k-i^*}$ from encryptions of all zeros strings under public key pk_{k-i^*+1}, therefore break q-IND-CPA security of the MFHE scheme.

The MFHE challenger generates public parameters params and a public-evaluation key pair $(\mathsf{pk}^*, \mathsf{ek}^*)$, and sends these to $\mathcal{B}$. $\mathcal{B}$ samples k MFHE key tuples as $(\mathsf{pk}_i, \mathsf{ek}_i, \mathsf{sk}_i) \leftarrow \mathsf{MFHE.KeyGen}(\mathsf{params})$ for $i \leq k+1$ and $i \neq k - i^* + 1$, and sets $(\mathsf{pk}_{k-i^*+1}, \mathsf{ek}_{k-i^*+1}) = (\mathsf{pk}^*, \mathsf{ek}^*)$. Next, it computes k COQTs as in Game $(2, i^*, 0)$, i.e. $(\sigma_i, \mathsf{key}_i) \leftarrow \mathsf{Encode}(\mathsf{MFHE.Dec}, \mathbf{S}, \mathsf{pos}, \mathsf{sk}_i, \mathsf{P}^\dagger)$ for $i \leq k - i^*$, and $(\sigma_i, \mathsf{key}_i) \leftarrow \mathsf{Encode}(\mathsf{MFHE.Dec}, \mathbf{S}, \mathsf{pos}, \mathbf{0}, \mathsf{P}^\dagger)$ otherwise. It also computes $2k-2$ MFHE ciphertexts $\mathsf{ct}_{\mathsf{key},i}, \mathsf{ct}_{\mathsf{sk},i}$ as $\mathsf{ct}_{\mathsf{key},i} \leftarrow \mathsf{MFHE.Enc}(\mathsf{pk}_{i+1}, \mathsf{key}_i)$, $\mathsf{ct}_{\mathsf{sk},i} \leftarrow \mathsf{MFHE.Enc}(\mathsf{pk}_{i+1}, \mathsf{sk}_i)$ for $i \leq k - (i^* + 1)$, and $\mathsf{ct}_{\mathsf{key},i} \leftarrow \mathsf{MFHE.Enc}(\mathsf{pk}_{i+1}, \mathbf{0})$, $\mathsf{ct}_{\mathsf{sk},i} \leftarrow \mathsf{MFHE.Enc}(\mathsf{pk}_{i+1}, \mathbf{0})$ for $i \geq k - (i^* - 1)$. For computing ciphertexts $\mathsf{ct}_{\mathsf{key},k-i^*}, \mathsf{ct}_{\mathsf{sk},k-i^*}$, it sends $(\mathsf{key}_{k-i^*}, \mathsf{sk}_{k-i^*})$ as its challenge messages to MFHE challenger.[32] The MFHE challenger flips a random bit γ and encrypts either $(\mathsf{key}_{k-i^*}, \mathsf{sk}_{k-i^*})$ or all-zeros strings, and sends the corresponding ciphertexts $\mathsf{ct}^*_1, \mathsf{ct}^*_2$ to $\mathcal{B}$. $\mathcal{B}$ sets $\mathsf{ct}_{\mathsf{key},k-i^*}, \mathsf{ct}_{\mathsf{sk},k-i^*} = \mathsf{ct}^*_1, \mathsf{ct}^*_2$. $\mathcal{B}$ also sets $\mathsf{params}', \mathsf{pk}, \rho_{\mathsf{ek}}$ as in Game $(2, i^*, 0)$, and sends these to the adversary $\mathcal{A}$. Next, $\mathcal{A}$ sends the challenge $\mathcal{R}^{\mathcal{M}}$ to $\mathcal{B}$. $\mathcal{B}$ chooses a random bit $\beta \leftarrow \{0,1\}$. If $\beta = 0$, it erases $\mathcal{R}^{\mathcal{M}}$, otherwise leaves it unchanged. It chooses random bits $a, b \leftarrow \{0,1\}$, encrypts them under pk_1 as $\mathsf{ct} \leftarrow \mathsf{MFHE.Enc}(\mathsf{pk}_1, (a,b))$. Next, it applies a quantum one-time pad on $\mathcal{R}^{\mathcal{M}}$ with keys a, b. $\mathcal{B}$ sends the challenge cipherstate as the ciphertext ct and register $\mathcal{R}^{\mathcal{M}}$ to $\mathcal{A}$. Finally, $\mathcal{A}$ outputs its guess β'. If $\beta = \beta'$,

[32] We would like to note that the reduction algorithm $\mathcal{B}$ is playing a multi-bit/multi-message q-IND-CPA game with MFHE challenger. Since multi-bit/multi-message q-IND-CPA security follows from standard single-bit/single-message q-IND-CPA security via a standard hybrid game, thus our lemma follows.

then $\mathcal{B}$ sends 0 as its guess (i.e., $(\mathsf{key}_{k-i^*}, \mathsf{sk}_{k-i^*})$ were encrypted), otherwise it sends 1 as its guess (i.e., all-zeros strings were encrypted) to the MFHE challenger.

First, note that $\mathcal{B}$ does not need to know the secret sk_{k-i^*+1} (i.e., secret key corresponding to pk^*) in the above reduction because ciphertext $\mathsf{ct}_{\mathsf{sk},k-i^*+1}$ is already an encryption of $\mathbf{0}$ as well as $(k-i^*+1)^{th}$ COQT is an encoding of $\mathbf{0}$. Also, if the MFHE challenger encrypted $(\mathsf{key}_{k-i^*}, \mathsf{sk}_{k-i^*})$ (i.e., $\gamma = 1$), then $\mathcal{B}$ perfectly simulates Game $(2, i^*, 0)$ for adversary $\mathcal{A}$. Otherwise it simulates Game $(2, i^*, 1)$ for $\mathcal{A}$. As a result, if $|\mathsf{Adv}_{\mathcal{A}}^{(2,i^*,0)} - \mathsf{Adv}_{\mathcal{A}}^{(2,i^*,1)}|$ is non-negligible, then $\mathcal{B}$ breaks the MFHE scheme's security with non-negligible advantage. ■

Lemma 5.2. If $\mathsf{COQT} = (\mathsf{Encode}, \mathsf{Apply}, \mathsf{Decode})$ is a secure conditional oblivious quantum transform, then for all $i^* \in \{0, 1, \ldots, k-1\}$ and all quantum PPT adversaries $\mathcal{A}$, $|\mathsf{Adv}_{\mathcal{A}}^{(2,i^*,1)} - \mathsf{Adv}_{\mathcal{A}}^{(2,i^*+1,0)}|$ is negligible in the security parameter λ.

Proof. Suppose there exists an adversary $\mathcal{A}$ such that $|\mathsf{Adv}_{\mathcal{A}}^{(2,i^*,1)} - \mathsf{Adv}_{\mathcal{A}}^{(2,i^*+1,0)}|$ is non-negligible for some $i^* \in \{1, \ldots, k-1\}$. We construct an algorithm $\mathcal{B}$ that can distinguish a COQT of input sk_{k-i^*} from COQT of all zeros strings with circuit $\mathsf{MFHE.Dec}$, set intervals $\mathbf{S}$, position pos and gate $\mathsf{P}^\dagger$, therefore break security of the COQT scheme.

$\mathcal{B}$ generates public parameters params and samples $k+1$ MFHE key pairs as $(\mathsf{pk}_i, \mathsf{ek}_i, \mathsf{sk}_i) \leftarrow \mathsf{MFHE.KeyGen}(\mathsf{params})$ for $i \leq k+1$. Next, it computes $k-1$ COQTs as $(\sigma_i, \mathsf{key}_i) \leftarrow \mathsf{Encode}(\mathsf{MFHE.Dec}, \mathbf{S}, \mathsf{pos}, \mathsf{sk}_i, \mathsf{P}^\dagger)$ for $i \leq k-(i^*+1)$, and $(\sigma_i, \mathsf{key}_i) \leftarrow \mathsf{Encode}(\mathsf{MFHE.Dec}, \mathbf{S}, \mathsf{pos}, \mathbf{0}, \mathsf{P}^\dagger)$ for $i \geq k-(i^*-1)$. To compute σ_{k-i^*}, $\mathcal{B}$ sends circuit $\mathsf{MFHE.Dec}$, set intervals $\mathbf{S}$, index pos, input strings sk_{k-i^*} and $\mathbf{0}$, and gate $\mathsf{P}^\dagger$ to the COQT challenger. The COQT challenger chooses a random bit γ, encodes either sk_{k-i^*} or $\mathbf{0}$, and sends σ^* as the corresponding challenge encoding. $\mathcal{B}$ sets $\sigma_{k-i^*} = \sigma^*$. Next, it computes $2k$ MFHE ciphertexts $\mathsf{ct}_{\mathsf{key},i}, \mathsf{ct}_{\mathsf{sk},i}$ as $\mathsf{ct}_{\mathsf{key},i} \leftarrow \mathsf{MFHE.Enc}(\mathsf{pk}_{i+1}, \mathsf{key}_i)$, $\mathsf{ct}_{\mathsf{sk},i} \leftarrow \mathsf{MFHE.Enc}(\mathsf{pk}_{i+1}, \mathsf{sk}_i)$ for $i \leq k-(i^*+1)$, and $\mathsf{ct}_{\mathsf{key},i} \leftarrow \mathsf{MFHE.Enc}(\mathsf{pk}_{i+1}, \mathbf{0})$, $\mathsf{ct}_{\mathsf{sk},i} \leftarrow \mathsf{MFHE.Enc}(\mathsf{pk}_{i+1}, \mathbf{0})$ for $i \geq k-i^*$. $\mathcal{B}$ sets $\mathsf{params}', \mathsf{pk}, \rho_{\mathsf{ek}}$ as in Game $(2, i^*, 1)$, and sends these to the adversary $\mathcal{A}$. Next, $\mathcal{A}$ sends the challenge $\mathcal{R}^{\mathcal{M}}$ to $\mathcal{B}$. $\mathcal{B}$ chooses a random bit $\beta \leftarrow \{0,1\}$. If $\beta = 0$, it erases $\mathcal{R}^{\mathcal{M}}$, otherwise leaves it unchanged. It chooses random bits $a, b \leftarrow \{0,1\}$, encrypts them under pk_1 as $\mathsf{ct} \leftarrow \mathsf{MFHE.Enc}(\mathsf{pk}_1, (a,b))$. Next, it applies a quantum one-time pad on $\mathcal{R}^{\mathcal{M}}$ with keys a, b. $\mathcal{B}$ sends the challenge cipherstate as the ciphertext ct and register $\mathcal{R}^{\mathcal{M}}$ to $\mathcal{A}$. Finally, $\mathcal{A}$ outputs its guess β'. If $\beta = \beta'$, then $\mathcal{B}$ sends 0 as its guess (i.e., sk_{k-i^*} was encoded), otherwise it sends 1 as its guess (i.e., all-zeros strings was encoded) to the COQT challenger.

First, note that $\mathcal{B}$ does not need to know the secret key_{k-i^*} (i.e., decoding key corresponding to σ^*) in the above reduction because ciphertext $\mathsf{ct}_{\mathsf{key},k-i^*}$ is already an encryption of $\mathbf{0}$. Also, if the COQT challenger encoded sk_{k-i^*} (i.e., $\gamma = 0$), then $\mathcal{B}$ perfectly simulates Game $(2, i^*, 1)$ for adversary $\mathcal{A}$. Otherwise it simulates Game $(2, i^*+1, 0)$ for $\mathcal{A}$. As a result, if $|\mathsf{Adv}_{\mathcal{A}}^{(2,i^*,1)} - \mathsf{Adv}_{\mathcal{A}}^{(2,i^*+1,0)}|$ is non-negligible, then $\mathcal{B}$ breaks the COQT's security with non-negligible advantage. ■

Lemma 5.3. If $\mathsf{MFHE} = (\mathsf{MFHE.Setup}, \mathsf{MFHE.KeyGen}, \mathsf{MFHE.Enc}, \mathsf{MFHE.Expand}, \mathsf{MFHE.Eval}, \mathsf{MFHE.Dec})$ is a q-IND-CPA secure multi-key fully homomorphic encryption scheme, then for all quantum PPT adversaries $\mathcal{A}$, $|\mathsf{Adv}_{\mathcal{A}}^{3} - \mathsf{Adv}_{\mathcal{A}}^{4}|$ is negligible in the security parameter λ.

Proof. The proof of this lemma is similar to that of Lemma 5.1. ■

Lemma 5.4. For all quantum PPT adversaries $\mathcal{A}$, $|\mathsf{Adv}_{\mathcal{A}}^{4}| = 0$.

Proof. The proof of this lemma is similar to that of [BJ15, Theorem 5.3]. Let $\rho^{\mathcal{M}}$ denote the quantum message state encrypted. The main idea is that the quantum component of the challenge cipherstate looks like a completely mixed state. This is because

$$\begin{aligned}\frac{1}{4}\sum_{a,b} \rho(\mathsf{MFHE.Enc}(\mathsf{pk}_1,(0,0))) \otimes (\mathsf{X}^a\mathsf{Z}^b)\rho^{\mathcal{M}}(\mathsf{X}^a\mathsf{Z}^b)^\dagger &= \rho(\mathsf{MFHE.Enc}(\mathsf{pk}_1,(0,0))) \otimes \frac{1}{4}\sum_{a,b}(\mathsf{X}^a\mathsf{Z}^b)\rho^{\mathcal{M}}(\mathsf{X}^a\mathsf{Z}^b)^\dagger \\ &= \rho(\mathsf{MFHE.Enc}(\mathsf{pk}_1,(0,0))) \otimes \frac{1}{2}\mathbb{I}_2.\end{aligned}$$

Therefore, challenge cipherstate σ from the perspective of $\mathcal{A}$ is independent of β. ■

This concludes the analysis and proof of Theorem 5.1.

6 Building Threshold Quantum Multi-Key Positional HE and Achieving Re-Randomizability

In this section, we show that the QMPHE scheme described in Section 5 gives 1-round distributed threshold decryption if the underlying classical scheme provides threshold decryption as well. Additionally, we argue that if the underlying classical scheme satisfies classical circuit privacy in the semi-honest setting [IP07], then our quantum multi-key HE scheme provides re-randomizability of cipherstates.

6.1 Threshold Decryption

Let $\mathsf{MFHE.PartDec}$, $\mathsf{MFHE.FinDec}$ be the threshold decryption algorithms supported by the classical multi-key fully homomorphic encryption scheme. First, we want to point out that any single expanded (and possibly evaluated) ciphertext $\widehat{\sigma}$ in the QMPHE scheme is a classical-quantum state that consists of two parts — (1) expanded classical ciphertext $\widehat{\mathsf{ct}}$, (2) single-qubit state $\widehat{\sigma}_{\mathsf{ct}}$. Below we describe the $\mathsf{PartDec}$, $\mathsf{FinDecPre}$ and $\mathsf{FinDecPost}$ algorithms for the quantum multi-key positional homomorphic encryption scheme. As before, we only focus on decrypting single-qubit cipherstates at a time.

- $\mathsf{PartDec}((\mathsf{pk}_1, \ldots, \mathsf{pk}_N), i, \mathsf{sk}_i, \widehat{\mathsf{ct}})$: The partial decryption algorithm takes as input N public keys $\mathsf{pk}^{(j)}$, secret key of the i^{th} user, and a single (expanded) ciphertext $\widehat{\mathsf{ct}}$. Let $\mathsf{sk}^{(i)} = (\mathsf{sk}_1^{(i)}, \ldots, \mathsf{sk}_{k+1}^{(i)})$, and $\widehat{\mathsf{ct}}$ be a ciphertext under level ℓ public keys.[33] It decrypts $\widehat{\mathsf{ct}}$ using level ℓ keys as $\mathsf{sh}_i \leftarrow \mathsf{MFHE.PartDec}(\mathsf{sk}_\ell^{(i)}, \widehat{\mathsf{ct}})$, and outputs sh_i as the output of partial decryption.
- $\mathsf{FinDecPre}(\mathsf{sh}_1, \ldots, \mathsf{sh}_N)$: The algorithm takes as input N partial decryptions $\mathsf{sh}_1, \ldots, \mathsf{sh}_N$, runs the FinDec algorithm as $\mathsf{rk} = \mathsf{MFHE.FinDec}(\mathsf{sh}_1, \ldots, \mathsf{sh}_N)$, and outputs rk as the reconstructed key.
- $\mathsf{FinDecPost}(\widehat{\sigma}_{\mathsf{ct}}, \mathsf{rk})$: The algorithm takes as input a single cipherstate $\widehat{\sigma}_{\mathsf{ct}}$, and key rk. Let $\mathsf{rk} = (a, b)$. It performs one-time decryption using keys a, b and outputs

$$(\mathsf{X}^a \mathsf{Z}^b)\ \widehat{\sigma}_{\mathsf{ct}}\ (\mathsf{X}^a \mathsf{Z}^b)^\dagger.$$

Correctness of Threshold Decryption. The proof of correct threshold decryption is similar to the proof of correct evaluation provided in Section 5.2. The only difference is that we will now use correctness of threshold decryption of the underlying classical MFHE scheme instead. Since we know that the classical component of the evaluated ciphertext encrypts the correct one-time pad keys, therefore running $\mathsf{MFHE.PartDec}$ on the ciphertext $\widehat{\mathsf{ct}}$ with key sk_i give partial decryption shares sh_i such that $\mathsf{MFHE.FinDec}(\mathsf{sh}_1, \ldots, \mathsf{sh}_N) = (a, b)$ where $\widehat{\sigma}_{\mathsf{ct}} = (\mathsf{X}^a \mathsf{Z}^b)\ C\,|\psi\rangle\ (\mathsf{X}^a \mathsf{Z}^b)^\dagger$. Thus, we get that $\mathsf{FinDecPost}(\widehat{\sigma}_{\mathsf{ct}}, \mathsf{rk} = (a, b)) = C\,|\psi\rangle$.

Unique Reconstruction. Note that during encryption, the algorithm chooses two random bits (a, b) for one-time pad encryption. Now the $\mathsf{Extract}$ algorithm simply outputs these two classical bits that are part of the random coins using during encryption. Clearly this matches the reconstructed key by correctness of the underlying MFHE scheme.

Simulation Security. Let $\mathsf{MFHE.Sim}^{thr}$ denote the simulator for the classical MFHE scheme. The simulator for the QMPHE scheme is identical to the simulator $\mathsf{MFHE.Sim}^{thr}$. The proof of security follows directly from the threshold simulation security of the classical MFHE scheme.

[33] Recall that the expanded ciphertexts might not be encrypted under top level keys.

6.2 Achieving Re-Randomizability

Informally, circuit privacy says that an evaluated cipherstate must be indistinguishable from a fresh encryption of the circuit output. If we modify our construction similar to that in [DSS16, Theorem 3], we also could prove quantum circuit privacy of our multi-key scheme in the semi-honest setting. Now to re-randomize cipherstates in our quantum multi-key HE scheme, we would simply do a fresh one-time pad encryption of the already encrypted state and update the one-time pad keys homomorphically. The idea is that by using quantum circuit privacy, we could argue that the evaluated cipherstates are simulatable given only the output of the evaluation. Next, by performing the re-randomization as described above (i.e., applying fresh quantum one-time pad and updating keys homomorphically) we can again use classical circuit privacy to hide the homomorphic key update and we already that applying a quantum one-time pad in succession simply $\oplus$s the keys which makes them look uniformly random (by a perfect secrecy argument).

7 Quantum Multi-Key HE: Removing Positional Constraint, Multi-Hop and More

In this section, we discuss various improvements to our quantum multi-key positional homomorphic encryption scheme construction described in Section 5. First, we show how to construct a standard (non-positional) quantum multi-key homomorphic encryption scheme for bounded number of users. In other words, the key generation algorithm will no longer take as input a position, but it will only take as input an upper bound on the number of users. Second, we discuss how to support multi-hop quantum computation as our current construction supports only single-hop evaluation where all parties must be known when computation/evaluation starts. Next, we also discuss how to set the circuit depth bound during setup, and finally reducing the number of classical MFHE key pairs required per user.

7.1 Constructing Quantum Multi-Key HE

The construction described in Section 5 can be easily extended to a (leveled) quantum multi-key homomorphic encryption scheme for a-priori bounded number of users, where the bound is fixed during key generation. At a high level, the idea is to redundantly add COQTs in the system such that each evaluation key contains all the information to be used as a positional evaluation key for any position $\leq N$. We briefly sketch the ideas below.

For simplicity, let us first only remove "position" as an input to the key generation algorithm, that is the key generation still takes as input the *exact* number of users, and not just an upper bound. This could be handled as follows — during key generation, the algorithm computes $N \times k$ COQTs as follows:

$$(\sigma_{i,j}, \mathsf{key}_{i,j}) \leftarrow \mathsf{Encode}(\mathsf{MFHE.Dec}, \mathbf{S}, j, \mathsf{sk}_i, \mathsf{P}^\dagger) \text{ for } i \leq k, j \leq N.$$

That is, it outputs k COQTs for each position $\mathsf{pos} \in \{1, \ldots, N\}$ where $\mathbf{S}$ is as defined in the construction. Next, it encrypts $\mathsf{key}_{i,j}$ for all i, j, and includes all these $N \times k$ COQTs as part of the quantum evaluation key. The encryption, decryption and expansion algorithm are same as before. Now the homomorphic evaluation procedure proceeds as before for clifford group operations, but for applying a T-gate on wire w, it first applies T-gate to $\widehat{\sigma}_w$ and then takes applies the COQTs on $\widehat{\sigma}_w$ as follows

$$(\widehat{\sigma}_w, \mathsf{aux}) \leftarrow \mathsf{Apply}(\mathsf{MFHE.Dec}, \mathbf{S}, \sigma_{\ell,1}^{(1)}, \ldots, \sigma_{\ell,N}^{(N)}, \widehat{\mathsf{ct}}_{w,1}, \widehat{\sigma}_w).$$

That is, it picks the i^{th} COQT from the i^{th} evaluation key corresponding to position i for each $i \leq N$. The (quantum) one-time pad key update procedure proceeds analogously. The security proof is also identical. The only modification will be that in the inner hybrids, all N COQTs at any particular level will be simultaneously switched from transforms of secret keys to all-zeros string. Indistinguishability of this step can be proved by a standard hybrid argument assuming security of COQT.

Finally, using similar ideas, above construction could be extended such that key generation algorithm takes as input an upper bound on the number of users, instead of an *exact* number. The idea will be to redundantly generate even more COQTs. More formally, during key generation the algorithm computes $N(N-1)/2 \times k$ COQTs as follows:

$$(\sigma_{i,j,n}, \mathsf{key}_{i,j,n}) \leftarrow \mathsf{Encode}(\mathsf{MFHE.Dec}, \mathbf{S}^n, j, \mathsf{sk}_i, \mathsf{P}^\dagger) \text{ for } i \leq k, j \leq n, n \leq N,$$

where $\mathbf{S}^n = (S_1^n, \ldots, S_{n+1}^n)$ is a sequence of $n+1$ set intervals where $S_i^n = \{s(\lambda, n) \cdot (i-1) + 1, \ldots, s(\lambda, n) \cdot i\}$ for $i \leq n$, and $S_{n+1}^n = \{s(\lambda, n) \cdot N + 1, \ldots, s(\lambda, n) \cdot N + p(\lambda, n)\}$. In other words, it outputs k COQTs for each position $\mathsf{pos} \in \{1, \ldots, n\}$ and every set size (i.e., exact number of users) $n \leq N$. As before, the only other modification will be during T-gate evaluation in which the evaluator will appropriately pick the COQTs and apply on the encrypted state. Also, the proof of security will be analogous with all $N(N-1)/2$ COQTs at any particular level being switched simultaneously.

7.2 Multi-Hop Evaluation

In a multi-hop multi-key homomorphic encryption scheme, for homomorphically evaluating any gate, only the keys associated with the ciphertexts corresponding to each input wire are required. In other words, it is not necessary to know all the evaluation keys associated with input ciphertexts at the start of evaluation phase, and repeated homomorphic evaluations can be performed on evaluated ciphertexts. It turns out that the quantum multi-key homomorphic encryption scheme (for a-priori bounded number of users) described above can easily be turned into a fully dynamic multi-hop scheme if the underlying classical scheme is fully dynamic multi-hop scheme. The only modification that has to be made is in the evaluation procedure. Below we discuss the changes to make it multi-hop. We would like to point out that the following construction also works only for a-priori bounded number of users which has be set during key generation.

In a multi-hop scheme we won't have a expansion algorithm as there is not necessarily any special distinction between evaluated ciphertexts and ciphertexts output by the encryption algorithm. Also, we would consider that evaluation algorithm only takes as input a single gate in $\{\mathsf{P}, \mathsf{H}, \mathsf{CNOT}, \mathsf{T}\}$, one or two cipherstates and all the evaluation keys associated with cipherstates. In order to perform homomorphic operations on larger quantum circuits, one could simply run the evaluation algorithm gate-by-gate. Moving on to our multi-hop construction, the key generation algorithm will be same as that for our QMLHE scheme for a-priori bounded number of parties. In particular, during key generation we sample $k+1$ classical MFHE key pairs, compute $N(N-1)/2 \times k$ COQTs, encrypt all $N(N-1)/2 \times k$ decoding keys and k MFHE secret keys appropriately. Next, the encryption and decryption algorithms will behave identically as well, that is encryption is simply computing a quantum one-time pad and encrypting the one-time pad keys under base public keys, and decryption proceeds by first extracting the quantum one-time pad keys from the classical ciphertexts and then performing a one-time pad decryption. Now the evaluation algorithm proceeds as follows:

1. **Evaluating a P or H gate.** This will mostly stay the same, i.e. first apply the gate to the quantum cipherstate and then update the one-time pad key components using MFHE evaluation. The only difference shall be that the evaluator will provide only the (classical) evaluation keys associated with the ciphertexts instead of all N keys at that level.

2. **Evaluating a T gate.** Suppose the ciphertexts encrypting the one-time pad keys are associated with n keys $\mathsf{pk}_\ell^{(1)}, \ldots, \mathsf{pk}_\ell^{(n)}$ at level ℓ. The evaluator first applies the T gate on the cipherstate, next it applies the COQT on it with set intervals $\mathbf{S}^n$ and quantum encodings $\sigma_{1,n,\ell}^{(1)}, \ldots, \sigma_{n,n,\ell}^{(n)}$ (i.e., encodings at level ℓ corresponding to set size n and appropriate positions). Next, it recrypts the one-time pad keys to level $\ell+1$, computes the decoding circuit, and performs the final $\oplus$ operation homomorphically. Note that for this (classical) homomorphic evaluation it only needs the associated evaluation keys at this level since the underlying scheme is multi-hop.

   At a high level, the T-gate evaluation proceeds analogously, except that now the evaluator only needs the associated evaluation keys and it does not recrypt the one-time pad keys to level $\ell + 1$ for all available wires. Previously, in T-gate evaluation, the one-time pad keys for all wires were recrypted together, but now this can't be done since evaluation is done gate-by-gate and independently. Thus, when we evaluate a CNOT gate, which is the only 2-qubit gate, the input wires might be at different level.

3. **Evaluating a CNOT gate.** As mentioned above, the ciphertexts encrypting the one-time pad keys for the input cipherstates might be at different levels. Suppose one of them is level ℓ and other at level ℓ' with $\ell' \leq \ell$. The evaluator proceeds as follows — first, apply the CNOT gate on appropriate wires; next, recrypt the ciphertexts at level ℓ' to level ℓ; and finally, perform the key updates after both ciphertexts are at level ℓ. In short, the only difference will be that before applying key update, the evaluator must recrypt the ciphertexts encrypting original one-time pad keys to the lowest common level.

7.3 Efficiency and More

Setting Circuit Depth Bound. For simplicity, in our current construction we assumed that the underlying classical scheme is fully homomorphic instead of being a leveled scheme. We would like point out that our transformation also works if we start with a leveled scheme. The circuit depth bound could be set as follows during the setup. Let d_1 and d_2 be the depth of circuits $f^{\mathsf{aux}}_{\mathsf{Decode}}$ and f^{v}_{Dec} (respectively) as defined in the evaluation algorithm. Apart from homomorphic evaluations of these two circuits, the quantum evaluation algorithm only performs $\oplus$ operations on the encrypted keys for each gate evaluation. So, the circuit depth bound could be set as $d_1 + d_2 + c$ where c is an upper bound on the clifford operations performed between any two consecutive T-gate computations.

Reducing number of MFHE key pairs. First, note that as per the current construction during key generation each user samples $k+1$ MFHE key pairs where k is the upper bound on the number of T gate evaluations that the scheme supports. We would like to point out that we could have instead sampled only $\ell + 1$ MFHE key pairs per user where ℓ would have been the maximum T-depth of quantum circuits supported. By T-depth of a quantum circuit, we mean the number of T layers in the quantum circuit if it is represented as a layered quantum circuit [AMMR13].

8 Instantiating Our Framework: Conditional Oblivious Quantum Transform for $\mathbf{NC}^1$

In this section, we construct a conditional oblivious quantum transform scheme for $\mathbf{NC}^1$ circuits and gate set $\{\mathsf{P}^\dagger\}$. We prove our scheme is unconditionally secure. Later in Section 9, we discuss how to bootstrap a conditional oblivious quantum transform scheme for class of log-depth circuits to all poly-size circuits. Below we give a brief overview. We would like to point out that our COQT construction for log-depth circuits is based on the gadget construction of Dulek et al. [DSS16]. Concretely, a COQT with $N = 1$ is the same as the DSS error-correcting gadget.

Outline. To encode an input x for index pos, we first generate an interval-alternating branching program BP corresponding to circuit C with intervals $\{S_i\}_i$. Recall that by definition of an interval-alternating BP, we have that the input bit read in each $2i^{th}$ state transition in BP lies in interval $S_{(i-1 \bmod N)+1}$, and in remaining transitions the input bit read lies in interval S_{N+1}. In other words, during even transitions/levels BP reads bits from first N intervals, and during odd transitions/levels it reads bits from the last interval.

Suppose $N = 2$, pos $= 1$, C takes as input 3 bits, and $S_i = \{i\}$ for $i \leq 3$. Thus, the input x being encoded is just a single bit. Also, for simplicity consider that the interval-alternating branching program corresponding to C reads the bit in following order - 3, 1, 3, 2. Now, the encoding σ of input x consists of 5 EPR pairs, thus 10 qubits. We label the qubits as follows: the i^{th} qubit (for $i \leq 5$) denotes branching program state i and level 2, and for $i > 5$ it denotes state $i-5$ and level 3. Let $\pi = \pi_{2,x}$, i.e. the permutation representing state transition at level 2 when input bit read is x. The qubits at levels 2 and 3 are connected according to permutation π, i.e. qubits i and $5 + \pi(i)$ for $i \leq 5$, are an EPR pair. The encoder also applies random Pauli matrices to the first qubit in each EPR pair, and these Pauli coefficients (for each pair) are included as part of the decoding key. Also, if the input bit read during last state transition lies in S_{pos} (in other words, if $\mathsf{pos} = N$), then the encoder applies gate G on the first qubit before applying random Pauli matrices.

So, at a high level, the quantum encoded state σ will be a collection of $5m$ pairs of entangled qubits (thus, total $10m$ qubits) with arbitrary Pauli operators applied on a single qubit in each pair and gate G conditionally applied on one special qubit, where m in the number of state transitions where input from interval S_{pos} is read. Note that each qubit is labeled by its state and level.

Now an evaluator gets as input N encodings, one for each index $i \leq N$. Recall that all the N intervals $S_1, \ldots, S_N$ are mutually exclusive and all even-numbered transitions read input bits from these intervals. Now, the evaluator *re-arranges* all these qubits according to their levels and states. In other words, for every branching program state and level, the evaluator uniquely associates a qubit in the encodings. Next, it labels the input qubit ρ to be at level 1 and state 1, i.e. starting state. Let x_{N+1} be the evaluator's input string. For applying the transform, it performs Bell measurements between qubits at level $2\ell - 1$ and 2ℓ as per the permutation specified for state transition at level $2\ell - 1$ and corresponding input bit in x_{N+1}. And, it stores all the measurement outcomes along with the location of associated qubits as the auxiliary information. In other words, at all odd state transitions the input bit is read from interval S_{N+1}, thus depending on the string x_{N+1} the evaluator performs successive Bell measurements between qubits associated with odd state transitions and stores the outcomes. This process continues until the input qubit is teleported to either an accepting or rejecting state in the last level. Now note that since the evaluator does not know the output of circuit C, thus it could not deterministically set the output qubit. In order to work around this issue, we use the idea from [DSS16] to apply an inverse branching program for circuit C afterwards. That is, now the encoder will start by generating the interval-alternating BP and then compute a new branching program which will be twice the size by appending the inverse of original program. This results in the invariant that the final state of the new program on each input will always be 1 (i.e., same as the starting state). Thus, the evaluator always sets the qubit in state 1 at the last level as the output qubit.[34]

The decoding procedure takes as input the decoding key which contains the information about entangled qubit pairs and their corresponding Pauli coefficients, and the auxiliary information which consists of the measurement outcomes. The decoder simply traces the path of the qubit from the starting level to last level, and generates the key updates accordingly. Below we describe our scheme is detail.

8.1 Construction

Let $\{\mathcal{C}_{d(n)}\}_n$ denote the class of all depth $d(n)$ circuits on n input bits with 1-bit output. Below we describe our scheme $\mathsf{COQT} = (\mathsf{Encode}, \mathsf{Apply}, \mathsf{Decode})$ for circuit class $\{\mathcal{C}_{d(n)}\}_n$ such that $d(n) = O(\log n)$, and gate set $\{\mathsf{P}^\dagger\}$.[35] For notational convenience, let $d = d(n)$.

- $\mathsf{Encode}(C, (S_1, \ldots, S_{N+1}), \mathsf{pos}, x, \mathsf{G})$: The encoding algorithm takes as input a classical circuit $C \in \mathcal{C}_d$, $(N + 1)$ set intervals $\{S_i\}_{i \leq N+1}$ of set $\{1, \ldots, n\}$, index $\mathsf{pos} \leq N$, a bit string $x \in \{0,1\}^{|S_i|}$, and description of a single-qubit gate G.

  Let $n_i = |S_i|$ for $i \leq N + 1$, and $\mathbf{S} = (S_1, \ldots, S_{N+1})$. The encoder generates an $\mathbf{S}$-interval-alternating branching program BP for circuit C.[36] By Corollary 3.1, we know that branching program BP has length $L = 2 \cdot N \cdot 4^d$ and width 5, and can be represented as

$$\mathsf{BP} = \Big(\mathsf{inp} : [L] \to [n], \{\pi_{i,b} : [5] \to [5]\}_{i \leq L, b \in \{0,1\}}, \mathsf{acc} \in [5], \mathsf{rej} \in [5]\Big).$$

[34]Technically, the qubit that the evaluator sets as the output bit is not always in the starting state, but the evaluator always knows which qubit to be set as the output.

[35]We would like to point out that the current construction could be used to give COQTs for Clifford group as well.

[36]Throughout this section we assume that the interval-alternating branching program constructed for any circuit C and set intervals $\mathbf{S}$ will always be the same branching program. In other words, constructing an interval-alternating branching program is a deterministic procedure. Note that this is *not* an extra assumption, and is already satisfied.

Let $\widetilde{\mathsf{BP}}$ denote a length $2L$ branching program of width 5 on n bits with accepting and rejecting states $\widetilde{\mathsf{acc}}, \widetilde{\mathsf{rej}} = 1, 2$, and the input selection function $\widetilde{\mathsf{inp}}$ and permutations $\widetilde{\pi}_{i,b}$ defined as follows

$$\widetilde{\mathsf{inp}}(i) = \begin{cases} \mathsf{inp}(i) & \text{if } i \leq L \\ \mathsf{inp}(2L+1-i) & \text{otherwise.} \end{cases} \qquad \widetilde{\pi}_{i,b} = \begin{cases} \pi_{i,b} & \text{if } i \leq L \\ \pi^{-1}_{2L+1-i,b} & \text{otherwise.} \end{cases}$$

Let T denote the set of state transitions in $\widetilde{\mathsf{BP}}$ where the input bit read is from interval S_{pos}, i.e. $T = \left\{ i \in \{1, \ldots, 2L\} : \widetilde{\mathsf{inp}}(i) \in S_{\mathsf{pos}} \right\}$. Let m denote the number of times an input bit from interval S_{pos} is read (i.e., $m = |T|$), and t_i denote the i^{th} element in T for $i \leq m$. Also, let x_i denote the $(i - i_{\mathsf{pos}})^{th}$ bit of string x (where $i_{\mathsf{pos}} = \sum_{i<\mathsf{pos}} n_i$), and let $\mathsf{next}(j, w) = \widetilde{\pi}_{j, x_{\widetilde{\mathsf{inp}}(j)}}(w)$.[37]

Let $p_{\ell,w} = 10(\ell - 1) + w$ and $q_{\ell,w} = 10\ell - 5 + \mathsf{next}(t_\ell, w)$ for all $\ell \leq m$, $w \leq 5$. The encoded state σ will be defined as

$$\bigotimes_{\ell=1}^{m} \bigotimes_{w=1}^{5} \left(\mathsf{X}^{a_{\ell,w}} \mathsf{Z}^{b_{\ell,w}} \mathsf{G}^{c_{\ell,w}} \right) \left| \Phi^+ \middle\rangle\!\middle\langle \Phi^+ \right|_{p_{\ell,w}, q_{\ell,w}} \left(\mathsf{X}^{a_{\ell,w}} \mathsf{Z}^{b_{\ell,w}} \mathsf{G}^{c_{\ell,w}} \right)^\dagger,$$

where all single-qubit gates are applied to first qubit (i.e., qubit $p_{\ell,w}$), and $|\Phi^+\rangle\langle\Phi^+|_{p_{\ell,w},q_{\ell,w}}$ denotes entanglement between qubits $p_{\ell,w}, q_{\ell,w}$, and $a_{\ell,w}, b_{\ell,w}$ are random bits, and $c_{\ell,w}$ is defined as

$$c_{\ell,w} = \begin{cases} 1 & \text{if } t_\ell = L \text{ and } \mathsf{next}(L, w) = \mathsf{acc} \\ 0 & \text{otherwise.} \end{cases}$$

In other words, the encoder sets the quantum encoded state σ as a collection of $10m$ qubits as follows

- For $\ell \leq m, w \leq 5$, $p^{th}_{\ell,w}$ qubit is (maximally) entangled with $q^{th}_{\ell,w}$ qubit.
- If $t_\ell = L$ and $\mathsf{next}(t_\ell, w) = \mathsf{acc}$ for some ℓ, w, then gate G is applied on $p^{th}_{\ell,w}$ qubit.
- For $\ell \leq m, w \leq 5$, random Pauli matrices are applied to qubit $p^{th}_{\ell,w}$.

Next, the decoding key will contain $5m$ 6-tuples as described below

$$\mathsf{key} = \left((t_\ell, w, \mathsf{next}(t_\ell, w), a_{\ell,w}, b_{\ell,w}, c_{\ell,w})_{\ell \leq m, w \leq 5} \right).$$

The encoding algorithm outputs the quantum encoding state σ and decoding key key.

- $\mathsf{Apply}(C, (S_1, \ldots, S_{N+1}), \sigma_1, \ldots, \sigma_N, x, \rho) \to (\rho', \mathsf{aux})$. The apply algorithm takes as input a classical circuit $C \in \mathcal{C}_d$, $(N+1)$ set intervals $\{S_i\}_{i \leq N+1}$, N encoded states σ_i, bit string $x \in \{0,1\}^{|S_{N+1}|}$, and a single-qubit state ρ.

  It computes branching program $\widetilde{\mathsf{BP}}$ corresponding to circuit C as during the encoding procedure. $\widetilde{\mathsf{BP}}$ is a length $2L$ branching program of width 5 on n bits with accepting and rejecting states $\widetilde{\mathsf{acc}} = 1, \widetilde{\mathsf{rej}} = 2$, and the input selection function $\widetilde{\mathsf{inp}}$ and permutations $\widetilde{\pi}_{i,b}$.

  Let $n_i = |S_i|$, $T_i = \left\{ j \in \{1, \ldots, 2L\} : \widetilde{\mathsf{inp}}(j) \in S_i \right\}$, $m_i = |T_i|$ for $i \leq N+1$, and $\sigma_i^{(j)}$ denote the j^{th} qubit of encoding σ_i for $i \leq N$ and $j \leq 10m_i$. Also, let x_i denote the $(i - i_{N+1})^{th}$ bit of string x, where $i_{N+1} = \sum_{i<N+1} n_i$.

  Let $\mathsf{pos} : [2L] \to [N+1]$ be a function such that $\mathsf{pos}(t) = i$ iff $\widetilde{\mathsf{inp}}(t) \in S_i$, and $\mathsf{ind} : [2L] \to [L]$ be a function such that $\mathsf{ind}(t) = j$ iff $|T_{\mathsf{pos}(t)} \cap \{1, \ldots, t\}| = j$. Also, let $\mathsf{next}(j, w) = \widetilde{\pi}_{j, x_{\widetilde{\mathsf{inp}}(j)}}(w)$ for $j \leq 2L$, $w \leq 5$.

  The apply algorithm performs bell measurements as follows and stores the corresponding measurement outcomes as part of the auxiliary information:

[37] Note that since BP is $\mathbf{S}$-interval-alternating BP, we know that $m = L/N$ and $T = \{2iN + 2\mathsf{pos} - 2N\}_{i \leq L/2N} \cup \{L + 2iN - 2\mathsf{pos} + 1\}_{i \leq L/2N}$.

- It performs bell measurements on qubits ρ and $\sigma_1^{(\mathsf{next}(1,1))}$. Let the outcomes be x, y.
- For $t \in T_{N+1} \setminus \{1, 2L\}$ and $w \leq 5$, it performs bell measurements on qubits $\sigma_{\mathsf{pos}(t-1)}^{(10\mathsf{ind}(t-1)+w-5)}$ and $\sigma_{\mathsf{pos}(t+1)}^{(10\mathsf{ind}(t+1)+\mathsf{next}(t,w)-10)}$. Let the outcomes be $x_{t,w}, y_{t,w}$.
- For $w \leq 5$ and $\ell = \mathsf{ind}(L)$, it performs bell measurements on qubits $\sigma_N^{(10\ell+w-5)}$ and $\sigma_N^{(10\ell+w)}$. Let the outcomes be x'_w, y'_w.

Finally, qubit $\sigma_1^{(10m_1+\mathsf{next}(1,1)-5)}$ will be set as the output qubit ρ'. And, the auxiliary information will contain $(5L-4)$ 5-tuples as described below.

$$\mathsf{aux} = \Big((1, 1, \mathsf{next}(1,1), x, y), (\perp, w, w, x'_w, y'_w)_{w\leq 5}, (t, w, \mathsf{next}(t,w), x_{t,w}, y_{t,w})_{t\in T', w\leq 5}\Big),$$

where $T' = T_{N+1} \setminus \{1, 2L\}$.

- $\mathsf{Decode}(\mathsf{key}_1, \ldots, \mathsf{key}_N, \mathsf{aux}) \to (a, b)$. The decoding algorithm takes as input N decoding keys key_i and auxiliary information aux, and outputs two bits (a, b).

  Let $5L$ be the total number of 6-tuples contained in the all the N decoding keys combined, and key denote the combined decoding key (i.e., all these $5L$ 6-tuples).

  The decoding algorithm sets $a = 0, b = 0$ and $\mathsf{st} = 1, t = 1$. It proceeds as follows.

  – For $1 \leq t \leq L$:
    * If t is odd, search $(t, \mathsf{st}, w, x, y)$ in aux and update $a = a + x, b = b + y, \mathsf{st} = w, t = t + 1$.
    * Otherwise, search $(t, \mathsf{st}, w, x, y, z)$ in key and update $a = a+x, b = a\cdot z+b+y, \mathsf{st} = w, t = t+1$.

  – For $t = L + 1$, search tuple $(\perp, \mathsf{st}, \mathsf{st}, x, y)$ in aux and $(t, \mathsf{st}, w, \widetilde{x}, \widetilde{y}, \widetilde{z})$ in key.
    Update $a = a + x + \widetilde{x}, b = (a + x) \cdot \widetilde{z} + b + y + \widetilde{y}, \mathsf{st} = w, t = t + 1$.

  – For $L + 1 < t \leq 2L - 1$:
    * If t is even, search $(t, \mathsf{st}, w, x, y)$ in aux and update $a = a + x, b = b + y, \mathsf{st} = w, t = t + 1$.
    * Otherwise, search $(t, \mathsf{st}, w, x, y, z)$ in key and update $a = a+x, b = a\cdot z+b+y, \mathsf{st} = w, t = t+1$.

  Finally, it outputs bits $(a \bmod 2, b \bmod 2)$.

8.2 Correctness

We will prove that the conditional oblivious quantum transform scheme described above satisfies the correctness property. Consider any classical circuit $C \in \mathcal{C}_d$, $(N+1)$ set intervals $\{S_i\}_{i\leq N+1}$ of set $\{1, \ldots, n\}$, $(N+1)$ inputs $x_i \in \{0,1\}^{|S_i|}$, let $\widetilde{\mathsf{BP}}$ denote the length $2L$ branching program as defined in the construction, and st_i denote the state of branching program $\widetilde{\mathsf{BP}}$ on input y after i steps, where $y = x_1 \,||\, x_2 \ldots ||\, x_{N+1}$.

We know that $\mathsf{st}_0 = 1$ and $\mathsf{st}_{2L} = 1$ (by construction). Also, the first L permutations of $\widetilde{\mathsf{BP}}$ represent circuit C, thus we know that $\mathsf{st}_L = \mathsf{acc}$ if $C(y) = 1$, and $\mathsf{st}_L = \mathsf{rej}$ otherwise. Now the encoding of ith input x_i (for $i \leq N$) with circuit C and gate $\mathsf{P}^\dagger$ is of the following form

$$\sigma_i = \bigotimes_{\ell=1}^{m_i} \bigotimes_{w=1}^{5} \left(\mathsf{X}^{a_{\ell,w}^{(i)}} \mathsf{Z}^{b_{\ell,w}^{(i)}} \mathsf{G}^{c_{\ell,w}^{(i)}}\right) |\Phi^+\rangle\langle\Phi^+|_{p_{\ell,w}^{(i)}, q_{\ell,w}^{(i)}} \left(\mathsf{X}^{a_{\ell,w}^{(i)}} \mathsf{Z}^{b_{\ell,w}^{(i)}} \mathsf{G}^{c_{\ell,w}^{(i)}}\right)^\dagger,$$

$$\mathsf{key}_i = \left(\left(t_\ell^{(i)}, w, \mathsf{next}(t_\ell^{(i)}, w), a_{\ell,w}^{(i)}, b_{\ell,w}^{(i)}, c_{\ell,w}^{(i)}\right)_{\ell\leq m_i, w\leq 5}\right),$$

where $T_i = \left\{j \in \{1, \ldots, 2L\} : \widetilde{\mathsf{inp}}(j) \in S_i\right\}$, $m_i = |T_i|$, $t_\ell^{(i)}$ denotes ℓ^{th} element in T_i, $\mathsf{next}(j, w) = \widetilde{\pi}_{j, y_{\widetilde{\mathsf{inp}}(j)}}(w)$ where y_i denotes i^{th} bit of y, and $p_{\ell,w}^{(i)} = 10(\ell-1)+w$, $q_{\ell,w}^{(i)} = 10\ell-5+\mathsf{next}(t_\ell^{(i)}, w)$ and $a_{\ell,w}^{(i)}, b_{\ell,w}^{(i)}$ are random bits for $\ell \leq m_i$, $w \leq 5$. Also, if $i = N$ and $t_\ell^{(i)} = L$, then $c_{\ell,w}^{(i)} = 1$, otherwise $c_{\ell,w}^{(i)} = 0$.

Next, consider any single-qubit state ρ. For ease of exposition, assume that $\rho = |\psi\rangle\langle\psi|$ (i.e., a pure state $|\psi\rangle$). We would like to show that applying the transform on $|\psi\rangle$ with these N encodings σ_i results in a state $\left|\widetilde{\psi}\right\rangle = (\mathsf{X}^a\mathsf{Z}^b(\mathsf{P}^\dagger)^{C(y)})\,|\psi\rangle$ where $(\rho', \mathsf{aux}) \leftarrow \mathsf{Apply}(C, (S_1, \ldots, S_{N+1}), \sigma_1, \ldots, \sigma_N, x, \rho)$, $(a, b) = \mathsf{Decode}(\mathsf{key}_1, \ldots, \mathsf{key}_N, \mathsf{aux})$ and $\rho' = \left|\widetilde{\psi}\right\rangle\!\left\langle\widetilde{\psi}\right|$.

We prove correctness of our construction by induction on the levels/branching program steps. If the measurements during apply procedure are sequentially analyzed, then we show that the decoding procedure correctly updates Pauli coefficients a, b and location of qubit ρ after each teleportation step. In other words, to prove correctness we trace the path of qubit during apply phase and verify a, b updates step-by-step.

Throughout the rest of the proof, whenever we say that the qubit is at level $2j$ and state w, then we mean it is $(10 \cdot \mathsf{ind}(2j) + w - 5)^{th}$ qubit in $\sigma_{\mathsf{pos}(2j)}$ (if $j \leq L/2$), and $(10 \cdot \mathsf{ind}(2j-1) + w - 5)^{th}$ qubit in $\sigma_{\mathsf{pos}(2j-1)}$ (otherwise).[38] Similarly, a qubit at level $2j-1$ and state w corresponds to $(10 \cdot \mathsf{ind}(2j) + w - 10)^{th}$ qubit in $\sigma_{\mathsf{pos}(2j)}$ (if $j \leq L/2$), and $(10 \cdot \mathsf{ind}(2j-1) + w - 10)^{th}$ qubit in $\sigma_{\mathsf{pos}(2j-1)}$ (otherwise). Also, we say that the input qubit ρ is at level 0 and state 1.

Below we prove (by induction) that after j^{th} set of measurements, the qubit will be teleported to level $2j$ and state st_{2j} (if $j \leq L/2$), otherwise it will be teleported to level $2j$ and state st_{2j-1}. Also, it will be in state $\left(\mathsf{X}^a\mathsf{Z}^b(\mathsf{P}^\dagger)^c\right)|\psi\rangle$, where values a, b will match those determined by partial execution of decoding procedure, and $c = 1$ if $j \geq L/2$ and $C(y) = 1$, else $c = 0$.

Base Case. Initially (i.e., before any measurement) the qubit is in state $|\psi\rangle$. Thus, by definition it is at level 0 and state $\mathsf{st}_0 = 1$, and a, b, c are all 0. Also, the decoding algorithm sets a, b to be 0 initially. This completes the proof for base case. For induction step, we show that if the above invariant holds till j^{th} set of measurements, then it also holds after $(j+1)^{th}$ set of measurements.

Induction Step. (Case 1: $j < L/2$) After j^{th} set of measurements (i.e., between levels $2(j-1)$ and $2j-1$), we have that the qubit is in state $\left(\mathsf{X}^a\mathsf{Z}^b(\mathsf{P}^\dagger)^c\right)|\psi\rangle$ where $c = 0$ and a, b match those determined by partial execution of decoding procedure. We know that the qubit will be at level $2j$ and state st_{2j}.

First, note that the $(j+1)^{th}$ set of measurements denotes bell measurements between qubits at level $2j$ and state w, and level $2j+1$ and state $\mathsf{next}(2j+1, w)$ (for $w \leq 5$). Also, we have that qubits at level $2j+1$ and state w, and level $2(j+1)$ and state $\mathsf{next}(2j+2, w)$ (for $w \leq 5$) are entangled. Therefore, we know that $(j+1)^{th}$ set of measurements will teleport qubit at level $2j$ and state st_{2j} to qubit at level $2j+2$ and state st_{2j+2}.

Let x, y be the outcome of the bell measurement, and $\widetilde{x}, \widetilde{y}, \widetilde{z} = a^{(i)}_{\ell,w}, b^{(i)}_{\ell,w}, c^{(i)}_{\ell,w}$ where $\ell = \mathsf{ind}(2j+2), w = \mathsf{st}_{2j+1}, i = \mathsf{pos}(2j+2)$. First, the measurement changes the state to

$$\left(\mathsf{X}^x\mathsf{Z}^y\right)\left(\mathsf{X}^a\mathsf{Z}^b(\mathsf{P}^\dagger)^c\right)|\psi\rangle = \left(\mathsf{X}^{a+x}\mathsf{Z}^{b+y}(\mathsf{P}^\dagger)^c\right)|\psi\rangle\,.$$

Next, the qubit is teleported through the entangled pair and since $\left(\mathsf{X}^{\widetilde{x}}\mathsf{Z}^{\widetilde{y}}(\mathsf{P}^\dagger)^{\widetilde{z}}\right)$ is applied to the first qubit in the entangled pair, thus teleportation changes the current state to

$$\begin{aligned}\left(\mathsf{X}^{\widetilde{x}}\mathsf{Z}^{\widetilde{y}}(\mathsf{P}^\dagger)^{\widetilde{z}}\right)\left(\mathsf{X}^{a+x}\mathsf{Z}^{b+y}(\mathsf{P}^\dagger)^c\right)|\psi\rangle &= \left(\mathsf{X}^{\widetilde{x}}\mathsf{Z}^{\widetilde{y}}\right)\left(\mathsf{X}^{a+x}\mathsf{Z}^{(a+x)\cdot\widetilde{z}+b+y}(\mathsf{P}^\dagger)^{c+\widetilde{z}}\right)|\psi\rangle\,. \\ &= \left(\mathsf{X}^{a+x+\widetilde{x}}\mathsf{Z}^{(a+x)\cdot\widetilde{z}+b+y+\widetilde{y}}(\mathsf{P}^\dagger)^{c+\widetilde{z}}\right)|\psi\rangle\,.\end{aligned}$$

Thus, the a, b pair is updated to $a', b' = a + x, b + y$ first, and then to $a' + x, a' \cdot \widetilde{z} + b' + \widetilde{y}$. Note that during the decoding phase, the same update procedure is carried out in two steps. Since the keys key_i and auxiliary information aux contains the correct key values and measurement outcomes (respectively), thus the decoding procedure correctly updates a, b pair.

Also, we know that $c = 0$ (by inductive hypothesis) and $\widetilde{z} = 1$ only if $2j + 2 = L$ and $\mathsf{st}_L = \mathsf{acc}$, thus $c + \widetilde{z} = 1$ only if $\mathsf{st}_L = \mathsf{acc}$ (i.e., if $C(y) = 1$), otherwise $c + \widetilde{z} = 0$. This maintains the invariant on c as well,

[38] Here ind and pos are as defined in the construction.

thereby completing the proof if $j < L/2$.

Induction Step. (Case 2: $j \geq L/2$) The proof in this case is identical to that for *Case 1*, except in the $(L/2+1)^{th}$ set of measurements, the apply algorithm performs measurements between qubits at level L and state w, and level $L+1$ and state w. In other words, it applies an identity permutation. Since qubits at level $2j+1$ and state w are entangled with qubits at level $2(j+1)$ and state $\mathsf{next}(2j+1, w)$, thus we know that $(L/2+1)^{th}$ set of measurements will teleport qubit to level $L+2$ and state st_{L+1}. Thus, the invariant after $(L/2+1)^{th}$ measurements will also be maintained. The remaining proof is same as before.

This completes the proof of our claim that after j^{th} set of measurements, the qubit will be in state $\left(\mathsf{X}^a\mathsf{Z}^b(\mathsf{P}^\dagger)^c\right)|\psi\rangle$ at level $2j$ and state st_{2j} (if $j \leq L/2$), or at level $2j$ and state st_{2j-1} (if $j > L/2$), where values a, b will match those determined by partial execution of decoding procedure, and $c = 1$ if $j \geq L/2$, else $c = 0$.

This implies that after L^{th} set of measurements (i.e., at the end of apply procedure), the qubit will be in state $\left(\mathsf{X}^a\mathsf{Z}^b(\mathsf{P}^\dagger)^c\right)|\psi\rangle$ at level $2L$ and state st_{2L-1} where $(a, b) = \mathsf{Decode}(\mathsf{key}_1, \ldots, \mathsf{key}_N, \mathsf{aux})$ and $c = 1$ iff $C(y) = 1$. This completes the correctness proof.

8.3 Security

We will now show that the scheme described above is secure as per Definition 4.1. Formally, we prove the following.

Theorem 8.1. The scheme $\mathsf{COQT} = (\mathsf{Encode}, \mathsf{Apply}, \mathsf{Decode})$ (described in Section 8.1) is a secure conditional oblivious quantum transform for circuit class $\mathcal{C}_d$ and gate set $\left\{\mathsf{P}^\dagger\right\}$ as per Definition 4.1.

The above statement holds unconditionally, and is not based on any cryptographic assumption. We prove a slightly stronger theorem by showing that even a computationally unbounded quantum adversary can not distinguish encodings of two different inputs $x^{(0)}, x^{(1)}$ for index i and circuit C. We show that encodings look like a completely mixed state to any adversary who does not know the corresponding key. Below we discuss this in more detail.

First, note that for any circuit $C \in \mathcal{C}_d$, intervals $\{S_i\}_{i \leq N+1}$ of set $\{1, \ldots, n\}$, index $i \leq N$, a bit string $x \in \{0,1\}^{|S_i|}$, its encoding σ is of the following form

$$\bigotimes_{\ell=1}^{m}\bigotimes_{w=1}^{5}\left(\mathsf{X}^{a_{\ell,w}}\mathsf{Z}^{b_{\ell,w}}(\mathsf{P}^\dagger)^{c_{\ell,w}}\right)|\Phi^+\rangle\langle\Phi^+|_{p_{\ell,w},q_{\ell,w}}\left(\mathsf{X}^{a_{\ell,w}}\mathsf{Z}^{b_{\ell,w}}(\mathsf{P}^\dagger)^{c_{\ell,w}}\right)^\dagger,$$

where $a_{\ell,w}, b_{\ell,w}$ are random bits, and single-qubit gates are applied to first qubit in the entangled pair. Now, note that for any c, we have

$$\frac{1}{4}\left(\sum_{a,b\in\{0,1\}}\left(\mathsf{X}^a\mathsf{Z}^b(\mathsf{P}^\dagger)^c\right)|\Phi^+\rangle\langle\Phi^+|\left(\mathsf{X}^a\mathsf{Z}^b(\mathsf{P}^\dagger)^c\right)^\dagger\right) = \frac{\mathbb{I}_4}{4}.$$

Since σ consists of $5m$ such pairs of qubits, therefore it is also a completely mixed state. More formally, we can write that

$$\frac{1}{4^{5m}}\left(\sum_{\substack{\forall \ell\leq m, w\leq 5:\\ a_{\ell,w},b_{\ell,w}\in\{0,1\}}}\bigotimes_{\ell=1}^{m}\bigotimes_{w=1}^{5}\left(\mathsf{X}^{a_{\ell,w}}\mathsf{Z}^{b_{\ell,w}}(\mathsf{P}^\dagger)^{c_{\ell,w}}\right)|\Phi^+\rangle\langle\Phi^+|_{p_{\ell,w},q_{\ell,w}}\left(\mathsf{X}^{a_{\ell,w}}\mathsf{Z}^{b_{\ell,w}}(\mathsf{P}^\dagger)^{c_{\ell,w}}\right)^\dagger\right) = \frac{\mathbb{I}_{4^{5m}}}{4^{5m}}.$$

Thus, for every input string x, its encoding looks like a completely mixed state on $10m$ qubits to any adversary that does not know the decoding key. This concludes the security proof.

9 Bootstrapping Conditional Oblivious Quantum Transform

In this section, we show how to bootstrap a conditional oblivious quantum transform scheme for a class of $\mathbf{NC}^1$ circuits to $\mathbf{P}/\mathsf{poly}$. Concretely, we show that using a (classical) multi-key homomorphic encryption scheme with log-depth decryption circuit, we can construct COQTs for poly-sized circuits from COQTs for log-depth circuits. Below we describe the construction in detail.

Outline. The main idea is to encrypt the inputs to be encoded under MLHE public keys and create COQT for the MLHE decryption circuit with input the corresponding secret key. Now during evaluation, one could simply evaluate the circuit homomorphically on the encrypted inputs and later apply the transform on the input qubit with the ciphertext encrypting the circuit output as the input. The decoding procedure will be identical to that of the underlying scheme.

The encoder gets as input a string x, circuit C, set intervals $\{S_i\}_i$, index pos and gate G. It first chooses an MLHE key pair and computes ct as the encryption of string x. Next, it computes a COQT for MLHE decryption circuit with the MLHE secret key as the input string, index pos and gate G. The encoder outputs the quantum encoding, MLHE public-evaluation keys and ciphertext ct as the encoded state, and the decoding key will simply be the same decoding keys output by the underlying encoding procedure.

Now an evaluator gets as input N encodings, one for each index $i \leq N$. It starts by homomorphically evaluating the circuit $C(\ldots, y)$ on the N MLHE ciphertexts contained in the encodings, where y is the input used by the evaluator. Let ct be the evaluated ciphertext. Next, it evaluates the COQTs with input ct on the input qubit and sets the output state and auxiliary information appropriately. Using the correctness of MLHE scheme, we can argue that ct is an encryption $C(x_1, \ldots, x_N, y)$, and by correctness of the underlying COQT scheme, we can argue the correctness of our bootstrapped scheme.

The security proof here is fairly simple. Using the security of the underlying COQT, we first switch the encodings of MLHE secret key to encodings of all-zeros string. Once the underlying quantum encoding is switched to be an encoding of all-zeros string, we no longer need the MLHE secret key. Therefore, we can also replace the MLHE encryption of string x with encryption of zeros, thereby erasing all the information about input x except its size. Below we describe our scheme in detail.

9.1 Construction

Let $\mathsf{MLHE} = (\mathsf{MLHE.Setup}, \mathsf{MLHE.KeyGen}, \mathsf{MLHE.Enc}, \mathsf{MLHE.Expand}, \mathsf{MLHE.Eval}, \mathsf{MLHE.Dec})$ be a multi-keyleveled homomorphic encryption scheme for 1-bit messages with log-depth decryption circuit, expanded ciphertexts of length $p(\lambda, N)$ and secret keys of length $s(\lambda, N)$. Also, let $\mathsf{COQT}_{\mathbf{NC}^1} = (\mathsf{Encode}_{\mathbf{NC}^1}, \mathsf{Apply}_{\mathbf{NC}^1}, \mathsf{Decode}_{\mathbf{NC}^1})$ be a conditional oblivious quantum transform for $\mathbf{NC}^1$ and gate set GS. Below we describe our scheme $\mathsf{COQT} = (\mathsf{Setup}, \mathsf{Encode}, \mathsf{Apply}, \mathsf{Decode})$[39] for circuit class $\mathcal{C}_d$ (i.e., the class of depth $d(n)$ circuits on n input bits with 1-bit output) and gate set GS, where d is any polynomial. For notational convenience, let $p = p(\lambda, N), s = s(\lambda, N)$ and $d = d(n)$.

- $\mathsf{Setup}(1^n, 1^d)$: The setup algorithm takes as input the circuit input length n and depth bound d. It runs $\mathsf{MLHE.Setup}$ to generate public parameters as $\mathsf{params} \leftarrow \mathsf{MLHE.Setup}(1^n, 1^d)$, and outputs params as the public parameters.

- $\mathsf{Encode}(\mathsf{params}, C, (S_1, \ldots, S_{N+1}), \mathsf{pos}, x, \mathsf{G})$: The encoding algorithm takes as input the public parameters params, a classical circuit $C \in \mathcal{C}_d$, $(N+1)$ set intervals $\{S_i\}_{i \leq N+1}$ of set $\{1, \ldots, n\}$, index $\mathsf{pos} \leq N$, a bit string $x \in \{0,1\}^{|S_{\mathsf{pos}}|}$, and description of a single-qubit gate $\mathsf{G} \in \mathsf{GS}$.

  It generates classical MLHE key pair as $(\mathsf{pk}, \mathsf{ek}, \mathsf{sk}) \leftarrow \mathsf{MLHE.KeyGen}(\mathsf{params})$. Consider set intervals $T_i = \{s \cdot (i-1) + 1, \ldots, s \cdot i\}$ for $i \leq N$, and $T_{N+1} = \{s \cdot N + 1, \ldots, s \cdot N + p\}$. Let $\mathbf{T} = (T_1, \ldots, T_{N+1})$. It computes a COQT as $(\sigma', \mathsf{key}') \leftarrow \mathsf{Encode}_{\mathbf{NC}^1}(\mathsf{MLHE.Dec}, \mathbf{T}, \mathsf{pos}, \mathsf{sk}, \mathsf{G})$. Next,

[39]Note that here we have an additional setup algorithm. This slightly departs from the definition in Section 4.2. We would like to point out that this constraint is due to the underlying MLHE scheme. If the underlying MLHE scheme has an empty setup algorithm, then our COQT scheme can be defined without setup as well.

it encrypts input x under public key pk as $\mathsf{ct} \leftarrow \mathsf{MLHE.Enc}(\mathsf{pk}, x)$. Finally, it outputs the encoded state and decoding key as

$$\sigma = \rho\left(\mathsf{pk}, \mathsf{ek}, \mathsf{ct}\right) \otimes \sigma', \quad \mathsf{key} = \mathsf{key}'.$$

- $\mathsf{Apply}(C, (S_1, \ldots, S_{N+1}), \sigma_1, \ldots, \sigma_N, x, \rho)$: The apply algorithm takes as input a classical circuit $C \in \mathcal{C}_d$, $(N+1)$ set intervals $\{S_i\}_{i \leq N+1}$, N encoded states σ_i, bit string $x \in \{0,1\}^{|S_{N+1}|}$, and a single-qubit state ρ. Let $\sigma_i = \rho\left(\mathsf{pk}_i, \mathsf{ek}_i, \mathsf{ct}_i\right) \otimes \sigma'_i$ for $i \leq N$.

  The apply algorithm expands all N ciphertexts ct_i as $\widehat{\mathsf{ct}}_i \leftarrow \mathsf{Expand}((\mathsf{pk}_1, \ldots, \mathsf{pk}_N), i, \mathsf{ct}_i)$ for $i \leq N$. Let $\widetilde{C}$ denote the circuit C with last $|S_{N+1}|$ bits hardwired to be x, i.e. $\widetilde{C}(y_1, \ldots, y_N) = C(y_1, \ldots, y_N, x)$ where $|y_i| = |S_i|$. It homomorphically evaluates circuit $\widetilde{C}$ on expanded ciphertexts $\widehat{\mathsf{ct}}_i$ as $\widetilde{\mathsf{ct}} \leftarrow \mathsf{MLHE.Eval}(\widetilde{C}, (\mathsf{ek}_1, \ldots, \mathsf{ek}_N), (\widehat{\mathsf{ct}}_1, \ldots, \widehat{\mathsf{ct}}_N))$.

  Consider set intervals $T_i = \{s \cdot (i-1)+1, \ldots, s \cdot i\}$ for $i \leq N$, and $T_{N+1} = \{s \cdot N + 1, \ldots, s \cdot N + p\}$. Let $\mathbf{T} = (T_1, \ldots, T_{N+1})$. Finally, it applies the COQT on ρ as

  $$(\rho', \mathsf{aux}) \leftarrow \mathsf{Apply}_{\mathbf{NC}^1}(\mathsf{MLHE.Dec}, \mathbf{T}, \sigma'_1, \ldots, \sigma'_N, \widetilde{\mathsf{ct}}, \rho)$$

  and outputs (ρ', aux) as the transformed state and auxiliary information.

- $\mathsf{Decode}(\mathsf{key}_1, \ldots, \mathsf{key}_N, \mathsf{aux})$: The decoding algorithm takes as input N decoding keys key_i and auxiliary information aux. It outputs (a, b) where $(a, b) \leftarrow \mathsf{Decode}_{\mathbf{NC}^1}(\mathsf{key}_1, \ldots, \mathsf{key}_N, \mathsf{aux})$.

9.2 Correctness

Consider any classical circuit $C \in \mathcal{C}_d$, $(N+1)$ set intervals $\{S_i\}_{i \leq N+1}$ of set $\{1, \ldots, n\}$, $(N+1)$ inputs $x_i \in \{0,1\}^{|S_i|}$, public parameters $\mathsf{params} \leftarrow \mathsf{MLHE.Setup}(1^n, 1^d)$ and gate $\mathsf{G} \in \mathsf{GS}$. Let z denote the string $x_1 \,||\, x_2 \ldots ||\, x_{N+1}$, and $\mathbf{T} = (T_1, \ldots, T_{N+1})$ be $(N+1)$ set intervals defined as $T_i = \{s \cdot (i-1)+1, \ldots, s \cdot i\}$ for $i \leq N$, and $T_{N+1} = \{s \cdot N + 1, \ldots, s \cdot N + p\}$.

For any index $i \leq N$, the encoding of i^{th} input x_i with circuit C, gate G and index i is of the following form $\rho\left(\mathsf{pk}_i, \mathsf{ek}_i, \mathsf{ct}_i\right) \otimes \sigma_i$, where $(\mathsf{pk}_i, \mathsf{ek}_i, \mathsf{sk}_i) \leftarrow \mathsf{MLHE.KeyGen}(\mathsf{params})$, $\mathsf{ct}_i \leftarrow \mathsf{MLHE.Enc}(\mathsf{pk}_i, x_i)$ and $(\sigma_i, \mathsf{key}_i) \leftarrow \mathsf{Encode}_{\mathbf{NC}^1}(\mathsf{MLHE.Dec}, \mathbf{T}, i, \mathsf{sk}_i, \mathsf{G})$. Also, it corresponding decoding key is simply key_i.

For correctness we need to argue that the apply algorithm applies gate G on state ρ iff $C(z) = 1$, and the decoding algorithm correctly computes the Pauli coefficients. First, note that (by correctness of MLHE expansion) we have that $\widehat{\mathsf{ct}}_i$ is an encryption of x_i under keys $(\mathsf{pk}_1, \ldots, \mathsf{pk}_N)$, where $\widehat{\mathsf{ct}}_i \leftarrow \mathsf{Expand}((\mathsf{pk}_1, \ldots, \mathsf{pk}_N), i, \mathsf{ct}_i)$. In other words, $\mathsf{MLHE.Dec}(\mathsf{sk}_1, \ldots, \mathsf{sk}_N, \widehat{\mathsf{ct}}_i) = x_i$ for $i \leq N$. Next, by correctness of $\mathsf{COQT}_{\mathbf{NC}^1}$, we have that

$$\rho' = \left(\mathsf{X}^a \mathsf{Z}^b \mathsf{G}^c\right) \rho \left(\mathsf{X}^a \mathsf{Z}^b \mathsf{G}^c\right)^\dagger,$$

where $(\rho', \mathsf{aux}) \leftarrow \mathsf{Apply}_{\mathbf{NC}^1}(\mathsf{MLHE.Dec}, \mathbf{T}, \sigma_1, \ldots, \sigma_N, \widetilde{\mathsf{ct}}, \rho)$, $(a, b) \leftarrow \mathsf{Decode}_{\mathbf{NC}^1}(\mathsf{key}_1, \ldots, \mathsf{key}_N, \mathsf{aux})$, $c = \mathsf{MLHE.Dec}(\mathsf{sk}_1, \ldots, \mathsf{sk}_N, \widetilde{\mathsf{ct}})$, $\widetilde{\mathsf{ct}} \leftarrow \mathsf{MLHE.Eval}(\widetilde{C}, (\mathsf{ek}_1, \ldots, \mathsf{ek}_N), (\widehat{\mathsf{ct}}_1, \ldots, \widehat{\mathsf{ct}}_N))$ and $\widetilde{C}(y_1, \ldots, y_N) = C(y_1, \ldots, y_N, x_{N+1})$. As before, equality here means zero trace distance between the associated states.

By correctness of MLHE evaluation, we have that $\widetilde{\mathsf{ct}}$ is an encryption of $C(z)$. Formally, we have that $\mathsf{MLHE.Dec}(\mathsf{sk}_1, \ldots, \mathsf{sk}_N, \widetilde{\mathsf{ct}}) = \widetilde{C}(x_1, \ldots, x_N) = C(x_1, \ldots, x_N, x_{N+1}) = C(z)$. Therefore, we know that

$$\rho' = \left(\mathsf{X}^a \mathsf{Z}^b \mathsf{G}^{C(z)}\right) \rho \left(\mathsf{X}^a \mathsf{Z}^b \mathsf{G}^{C(z)}\right)^\dagger.$$

This completes the proof of correctness.

9.3 Security

We will now show that the scheme described above is secure as per Definition 4.1.[40] Formally, we prove the following.

[40]The security property for a COQT scheme with Setup is analogously defined. The only difference is that the adversary is also given the public parameters.

Theorem 9.1. If $\mathsf{MLHE} = (\mathsf{MLHE.Setup}, \mathsf{MLHE.KeyGen}, \mathsf{MLHE.Enc}, \mathsf{MLHE.Expand}, \mathsf{MLHE.Eval}, \mathsf{MLHE.Dec})$ is a q-IND-CPA secure multi-key leveled homomorphic encryption scheme for 1-bit messages satisfying Definition 3.1, and $\mathsf{COQT}_{\mathbf{NC}^1} = (\mathsf{Encode}_{\mathbf{NC}^1}, \mathsf{Apply}_{\mathbf{NC}^1}, \mathsf{Decode}_{\mathbf{NC}^1})$ is a secure conditional oblivious quantum transform for $\mathbf{NC}^1$ and gate set GS satisfying Definition 4.1, then the scheme COQT (described in Section 9.1) is a secure conditional oblivious quantum transform for $\mathbf{P}/\mathsf{poly}$ and gate set GS as per Definition 4.1.

Our proof proceeds via a sequence of hybrid games. Each game is played between the challenger and attacker $\mathcal{A}$. Let $\mathcal{A}$ be any quantum PPT adversary that wins the security game with non-negligible advantage. We argue that such an adversary must break security of at least one underlying primitive. The first game corresponds to the security game as described in Definition 4.1. In the next game, we switch the COQTs to an empty transform (i.e., encoding of all-zeros string instead of sk). Indistinguishability of this step follows directly from COQT security. Next, we could argue (using q-IND-CPA security) that since the adversary has no information about the MLHE secret key, it can not distinguish between encryptions of challenge inputs $x^{(0)}, x^{(1)}$. Below we describe the proof in detail.

Throughout the hybrids, the set intervals $\mathbf{T} = (T_1, \ldots, T_{N+1})$ are defined as in the construction, i.e. $T_i = \{s \cdot (i-1)+1, \ldots, s \cdot i\}$ for $i \leq N$ and $T_{N+1} = \{s \cdot N + 1, \ldots, s \cdot N + p\}$. Also, we will use $\mathbf{0}$ to denote the all-zeros string of appropriate length. We would like point out that in the hybrid games the adversary and the challenger are both given a circuit $C \in \mathcal{C}_d$ and gate $\mathsf{G} \in \mathsf{GS}$. And, the indistinguishability proofs hold irrespective of the choice of C, or G.

Game 1: This game is same as the original security game.

1. **Setup Phase.** The challenger sets up by sampling the MLHE public parameters $\mathsf{params} \leftarrow \mathsf{MLHE.Setup}(1^n, 1^d)$. It sends the public parameters to $\mathcal{A}$.

2. **Challenge.** $\mathcal{A}$ sends a sequence of $(N+1)$ set intervals $\{S_i\}_{i \leq N+1}$ of set $\{1, \ldots, n\}$, index $i \leq N$, and two bit strings $x^{(0)}, x^{(1)} \in \{0,1\}^{|S_i|}$.

   The challenger chooses a random bit $b \leftarrow \{0,1\}$. It generates classical MLHE key pair as $(\mathsf{pk}, \mathsf{ek}, \mathsf{sk}) \leftarrow \mathsf{MLHE.KeyGen}(\mathsf{params})$. It computes a COQT as $(\sigma, \mathsf{key}) \leftarrow \mathsf{Encode}_{\mathbf{NC}^1}(\mathsf{MLHE.Dec}, \mathbf{T}, i, \mathsf{sk}, \mathsf{G})$. Next, it encrypts input $x^{(b)}$ under public key pk as $\mathsf{ct} \leftarrow \mathsf{MLHE.Enc}(\mathsf{pk}, x^{(b)})$. Finally, it sends the encoded state as $\rho\left(\mathsf{pk}, \mathsf{ek}, \mathsf{ct}\right) \otimes \sigma$ to $\mathcal{A}$.

3. **Guess.** $\mathcal{A}$ outputs it guess b' and wins if $b' = b$.

Game 2: This game is same as Game 1, except the challenger computes COQT (σ, key) as transforms of $\mathbf{0}$ instead of sk.

2. **Challenge.** $\mathcal{A}$ sends a sequence of $(N+1)$ set intervals $\{S_i\}_{i \leq N+1}$ of set $\{1, \ldots, n\}$, index $i \leq N$, and two bit strings $x^{(0)}, x^{(1)} \in \{0,1\}^{|S_i|}$.

   The challenger chooses a random bit $b \leftarrow \{0,1\}$. It generates classical MLHE key pair as $(\mathsf{pk}, \mathsf{ek}, \mathsf{sk}) \leftarrow \mathsf{MLHE.KeyGen}(\mathsf{params})$. It computes a COQT as $(\sigma, \mathsf{key}) \leftarrow \mathsf{Encode}_{\mathbf{NC}^1}(\mathsf{MLHE.Dec}, \mathbf{T}, i, \mathbf{0}, \mathsf{G})$. Next, it encrypts input $x^{(b)}$ under public key pk as $\mathsf{ct} \leftarrow \mathsf{MLHE.Enc}(\mathsf{pk}, x^{(b)})$. Finally, it sends the encoded state as $\rho\left(\mathsf{pk}, \mathsf{ek}, \mathsf{ct}\right) \otimes \sigma$ to $\mathcal{A}$.

9.3.1 Analysis

Let $\mathsf{Adv}^i_{\mathcal{A}} = |\Pr[b' = b] - 1/2|$ denote the advantage of adversary $\mathcal{A}$ in guessing the bit b in Game i. To complete the proof, we establish via a sequence of lemmas that no quantum PPT adversary $\mathcal{A}$ can distinguish between Games 1 and 2 with non-negligible probability, and the advantage of every quantum PPT adversary in Game 2 is also negligible. Below we discuss our lemmas in detail.

Lemma 9.1. If $\mathsf{COQT}_{\mathbf{NC}^1} = (\mathsf{Encode}_{\mathbf{NC}^1}, \mathsf{Apply}_{\mathbf{NC}^1}, \mathsf{Decode}_{\mathbf{NC}^1})$ is a secure conditional oblivious quantum transform, then for every quantum PPT adversary $\mathcal{A}$, $|\mathsf{Adv}^1_{\mathcal{A}} - \mathsf{Adv}^2_{\mathcal{A}}|$ is negligible in n.

Proof. Suppose there exists an adversary $\mathcal{A}$ such that $|\mathsf{Adv}^1_{\mathcal{A}} - \mathsf{Adv}^2_{\mathcal{A}}|$ is non-negligible. We construct an algorithm $\mathcal{B}$ that can distinguish a COQT of input sk from COQT of all zeros strings with circuit $\mathsf{MLHE.Dec}$, set intervals $\mathbf{T}$, position i and gate G, therefore break security of the COQT scheme.

$\mathcal{B}$ generates public parameters params as $\mathsf{params} \leftarrow \mathsf{MLHE.Setup}(1^n, 1^d)$ and sends params to $\mathcal{A}$. $\mathcal{A}$ sends set intervals $\{S_i\}_{i\leq N+1}$, index $i \leq N$, and two bit strings $x^{(0)}, x^{(1)} \in \{0,1\}^{|S_i|}$ to $\mathcal{B}$. $\mathcal{B}$ samples MLHE key pair as $(\mathsf{pk}, \mathsf{ek}, \mathsf{sk}) \leftarrow \mathsf{MLHE.KeyGen}(\mathsf{params})$. It sends circuit $\mathsf{MLHE.Dec}$, set intervals $\mathbf{T}$, index i, input strings sk and $\mathbf{0}$, and gate G to the COQT challenger. The COQT challenger chooses a random bit β, encodes either sk or $\mathbf{0}$, and sends σ^* as the corresponding challenge encoding. $\mathcal{B}$ sets $\sigma = \sigma^*$. Next, it chooses a random bit $b \leftarrow \{0,1\}$, and encrypts $x^{(b)}$ as $\mathsf{ct} \leftarrow \mathsf{MLHE.Enc}(\mathsf{pk}, x^{(b)})$. Finally, $\mathcal{B}$ sends the encoded state as $\rho(\mathsf{pk}, \mathsf{ek}, \mathsf{ct}) \otimes \sigma$ to the adversary $\mathcal{A}$. Finally, $\mathcal{A}$ outputs its guess b'. If $b = b'$, then $\mathcal{B}$ sends 0 as its guess (i.e., sk was encoded), otherwise it sends 1 as its guess (i.e., all-zeros strings was encoded) to the COQT challenger.

First, note that $\mathcal{B}$ does not need to know the secret key (i.e., decoding key corresponding to σ^*) in the above reduction. Also, if the COQT challenger encoded sk (i.e., $\beta = 0$), then $\mathcal{B}$ perfectly simulates Game 1 for adversary $\mathcal{A}$. Otherwise it simulates Game 2 for $\mathcal{A}$. As a result, if $|\mathsf{Adv}^1_{\mathcal{A}} - \mathsf{Adv}^2_{\mathcal{A}}|$ is non-negligible, then $\mathcal{B}$ breaks the COQT's security with non-negligible advantage. ∎

Lemma 9.2. If $\mathsf{MLHE} = (\mathsf{MLHE.Setup}, \mathsf{MLHE.KeyGen}, \mathsf{MLHE.Enc}, \mathsf{MLHE.Expand}, \mathsf{MLHE.Eval}, \mathsf{MLHE.Dec})$ is a q-IND-CPA secure multi-key leveled homomorphic encryption scheme, then for every quantum PPT adversary $\mathcal{A}$, $\mathsf{Adv}^2_{\mathcal{A}}$ is negligible in n.

Proof. Suppose there exists an adversary $\mathcal{A}$ such that $\mathsf{Adv}^2_{\mathcal{A}}$ is non-negligible. We construct an algorithm $\mathcal{B}$ that can distinguish between encryptions of $x^{(0)}, x^{(1)}$ under public key pk, therefore breakq-IND-CPA security of the MLHE scheme.

The MLHE challenger generates public parameters params and a public-evaluation key pair $(\mathsf{pk}^*, \mathsf{ek}^*)$, and sends these to $\mathcal{B}$. $\mathcal{B}$ sets $\mathsf{pk}, \mathsf{ek} = \mathsf{pk}^*, \mathsf{ek}^*$, and sends params to $\mathcal{A}$. $\mathcal{A}$ sends set intervals $\{S_i\}_{i\leq N+1}$, index $i \leq N$, and two bit strings $x^{(0)}, x^{(1)} \in \{0,1\}^{|S_i|}$ to $\mathcal{B}$. $\mathcal{B}$ sends $x^{(0)}, x^{(1)}$ as its challenge messages to MLHE challenger. The MLHE challenger flips a random bit β and encrypts either $x^{(0)}$ or $x^{(1)}$, and sends the corresponding ciphertext ct^* to $\mathcal{B}$. $\mathcal{B}$ sets $\mathsf{ct} = \mathsf{ct}^*$, and computes COQT as $(\sigma, \mathsf{key}) \leftarrow \mathsf{Encode}_{\mathbf{NC}^1}(\mathsf{MLHE.Dec}, \mathbf{T}, i, \mathbf{0}, \mathsf{G})$. Finally, $\mathcal{B}$ sends the encoded state as $\rho(\mathsf{pk}, \mathsf{ek}, \mathsf{ct}) \otimes \sigma$ to the adversary $\mathcal{A}$. Finally, $\mathcal{A}$ outputs its guess b'. $\mathcal{B}$ simply forwards b' as its own guess to the MLHE challenger.

First, note that $\mathcal{B}$ does not need to know the secret sk^* (i.e., secret key corresponding to pk^*) in the above reduction. And, since $\mathcal{B}$ perfectly simulates Game 2 for adversary $\mathcal{A}$, therefore if $\mathsf{Adv}^2_{\mathcal{A}}$ is non-negligible, then $\mathcal{B}$ breaks the MLHE scheme's security with non-negligible advantage. ∎

10 A Template for On-the-Fly Multi-Party Quantum Computation

In this section, we provide an on-the-fly multi-party quantum computation (MPQC) protocol using quantum multi-key homomorphic encryption. The same protocol can be used for multi-party delegation by providing the circuit description along the input qubits and asking the server to run the universal circuit. Below we briefly mention the target ideal functionalities, and then describe our construction.

On-the-fly MPQC Functionality. For an on-the-fly N-party MPQC protocol, the ideal functionality $\mathcal{F}^{\mathsf{MPQC}}$ interacts with $N+1$ parties where the first N parties correspond to the N parties running the MPQC protocol, and the $(N+1)^{th}$ party corresponds to the server S. Now the first N parties supply their quantum inputs for computation, but server S does not provide any input. Finally, the ideal functionality $\mathcal{F}^{\mathsf{MPQC}}$ sends the appropriate quantum registers containing the final output to the respective N users, and server S does not receive any output. Here the functionality $\mathcal{F}^{\mathsf{MPQC}}$ can also be parameterized by the quantum circuit C that parties wish to compute. Intuitively, this captures the functionality that we would ideally expect as each party only learns its final output state whereas the server S does not learn anything.

"Blind" MPQC Functionality. If we extend the above functionality such that each party also supplies the description of the quantum circuit C that they wish to jointly evaluate, and now we also restrict that the server does not learn any information about the circuit C then that gives us the functionality for blind MPQC.[41]

10.1 Protocol

Let $\mathsf{QMLHE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Expand}, \mathsf{Eval}, \mathsf{Dec}, \mathsf{PartDec}, \mathsf{FinDecPre}, \mathsf{FinDecPost}, \mathsf{ReRand})$ be a quantum multi-key leveled homomorphic encryption scheme that supports threshold decryption and allows ciphertext re-randomizability. Let $\mathcal{M} = \mathbb{C}^{\{0,1\}}$, and let $C : \mathcal{M}^{\otimes n} \to \mathcal{M}^{\otimes n}$ be the n-qubit quantum circuit that the server/parties want to compute. Let k be the number of T-gates in the circuit C, and (for $i \in [N]$) let n_i, n'_i denote the number of qubits provided by i^{th} user as input and received by i^{th} user as output. Without loss of generality, we assume that input and output qubits (for circuit C) are arranged as per the natural ordering among the users, i.e. as per their indices.

In case the QMLHE scheme has a non-empty setup algorithm, the server S samples global parameters as $\mathsf{params} \leftarrow \mathsf{Setup}(1^\lambda, 1^k)$ and publishes params.[42] Otherwise, params is set to be the security parameter 1^λ and T-gate bound k. Now the protocol π is defined as follows:

Round I. For $\ell \in [N]$, party P_ℓ on input $\rho_\ell \in \mathcal{M}^{\otimes n_\ell}$ proceeds as follows:

1. It samples a QMLHE key tuple as $(\mathsf{pk}_\ell, \rho_{\mathsf{ek}_\ell}, \mathsf{sk}_\ell) \leftarrow \mathsf{KeyGen}(\mathsf{params})$.
2. It encrypts the input state ρ_ℓ as $\sigma_\ell \leftarrow \mathsf{Enc}(\mathsf{pk}_\ell, \rho_\ell)$.[43]
3. It sends the public-evaluation key pair and the cipherstate $(\mathsf{pk}_\ell, \rho_{\mathsf{ek}_\ell}, \sigma_\ell)$ to the server.[44]

Round II. The server S on receiving keys and cipherstates as $\{(\mathsf{pk}_\ell, \rho_{\mathsf{ek}_\ell}, \sigma_\ell)\}_{\ell \in [N]}$ proceeds as follows:

1. It evaluates circuit C homomorphically by first expanding the cipherstates and then running homomorphic evaluation as follows:

$$\forall \ell \in [N], \quad \widehat{\sigma}_\ell \leftarrow \mathsf{Expand}((\mathsf{pk}_1, \ldots, \mathsf{pk}_N), \ell, \sigma_\ell)$$
$$\widehat{\sigma}' \leftarrow \mathsf{Eval}(C, (\rho_{\mathsf{ek}_1}, \ldots, \rho_{\mathsf{ek}_\ell}), \widehat{\sigma})$$

2. It re-randomizes the evaluated cipherstates $\widehat{\sigma}'$ as $\widehat{\sigma}^* \leftarrow \mathsf{ReRand}((\mathsf{pk}_1, \ldots, \mathsf{pk}_N), \widehat{\sigma}')$.
3. Let $\widehat{\sigma}^* = \otimes_{\ell \in [N]} \widehat{\sigma}^*_\ell$ where $\widehat{\sigma}^*_\ell$ denotes the cipherstate corresponding to the ℓ^{th} party's output. By definition, we have that each cipherstate can be divided into quantum and classical components as $\widehat{\sigma}^*_\ell = \rho(\widehat{\mathsf{ct}}^*_\ell) \otimes \widetilde{\sigma}^*_\ell$. Finally, the server S sends $\widetilde{\sigma}^*_\ell$ to party P_ℓ, and broadcasts all classical cipherstate components $\widehat{\mathsf{ct}}^*_\ell$ (for $\ell \in [N]$).

Round III. For $\ell \in [N]$, party P_ℓ on messages $\widetilde{\sigma}^*_\ell$ and $\left\{\widehat{\mathsf{ct}}^*_\ell\right\}_{\ell \in [N]}$ proceeds as follows:

1. For $i \in [N]$, it computes partial decryption as $\mathsf{sh}^{(i)}_\ell \leftarrow \mathsf{PartDec}((\mathsf{pk}_1, \ldots, \mathsf{pk}_N), \ell, \mathsf{sk}_\ell, \widehat{\mathsf{ct}}^*_i)$.
2. For $i \in [N] \setminus \{\ell\}$, it encrypts (classical) partial decryptions $\mathsf{sh}^{(i)}_\ell$ as $\mathsf{ct}^{(i)}_\ell \leftarrow \mathsf{Enc}(\mathsf{pk}_i, \mathsf{sh}^{(i)}_\ell)$, and broadcasts (classical) ciphertexts $\mathsf{ct}^{(i)}_\ell$.[45]

[41] We would like to point out that our MPQC construction can not be made fully blind. The functionality will always leak an upper bound on the number of T-gates present in the circuit being evaluated.

[42] Parameters params could be sampled using a CRS if the Setup is public-coin.

[43] Recall that a multi-qubit message state is encrypted qubit-by-qubit. However, for notational convenience we overload the encryption algorithm to encrypt multi-qubit message states as well.

[44] Here the classical components could be broadcasted and only the quantum information needs to be sent to the server via an authenticated quantum channel.

[45] Here we assume for simplicity that the QMLHE also supports directly encrypting classical plaintexts as well. Note that this is not an additional requirement as QMLHE is strictly stronger than MLHE.

Output. For $\ell \in [N]$, party P_ℓ on broadcast messages $\left\{\mathsf{sh}_\ell^{(i)}\right\}_{i \in [N] \setminus \{\ell\}}$ proceeds as follows:

1. It computes reconstruction key as $\mathsf{rk}_\ell \leftarrow \mathsf{FinDecPre}(\mathsf{sh}_\ell^{(1)}, \ldots, \mathsf{sh}_\ell^{(N)})$.
2. Finally, it decrypts cipherstate $\widetilde{\sigma}_\ell^*$ to obtain the output state ρ_ℓ^* as $\rho_\ell^* \leftarrow \mathsf{FinDecPost}(\widetilde{\sigma}_\ell^*, \mathsf{rk}_\ell)$.

NOTE. We would like to stress that the server S can be choose the circuit C (with at most k T-gates) anytime after round I. Thus, the resulting protocol is an on-the-fly MPQC protocol.

The correctness of the above protocol follows directly from the correctness of the underlying QMLHE scheme. Below we briefly sketch our intuition for security. We break down the security analysis in two parts — (1) Server S is honest, (2) Server S is dishonest but all delegating parties are honest. If S is dishonest but all delegating parties are honest, then security would simply follow from q-IND-CPA security of the QMHE scheme.

And if S is honest, we could expect the simulator to work as follows. First, it receives the encrypted input states from the adversary after round I. Then the simulator extracts the adversary's input from cipherstates by running the QMHE decryption algorithm with the adversary's secret key. Here we assume that the simulator gets the secret keys from the special witness tape (as defined in the semi-malicious security model). Next, the simulator gives the extracted input states to the ideal functionality and gets the corresponding output states. It then encrypts the output states under the public key of appropriate party, expands the cipherstates, and sends the quantum component of the cipherstates to the appropriate corrupt party and broadcasts the classical component of the cipherstates. Finally, it simulates the partial decryptions of the classical ciphertext components on the behalf of honest parties.[46] The main proof ideas will be — (1) simulated partial decryptions are indistinguishable from honest partial decryptions due to partial decryption simulation security, (2) re-randomized homomorphically evaluated cipherstate is indistinguishable from a fresh encryption of the quantum circuit output, and (3) honest parties' input states (which are encrypted under their public keys) are hidden by q-IND-CPA security.

Now one might expect the above protocol to achieve a quantum analogue of semi-malicious security since our basic template is inspired by those used in the classical setting [LATV12, AJL+12, MW16]. However, the notion of semi-malicious quantum adversary does not seem to be well defined (again) because of unclonability of quantum states. To understand the issue, let us first recall the notion of classical semi-malicious adversaries. Such classical adversaries are restricted to explain their messages sent on the behalf of the corrupt parties during each round by writing to a special witness tape a classical string (x, r) consisting of the input x and randomness r used. Now the behaviour of such adversaries could be verified at each step of the protocol execution. However in the quantum setting, we can not expect the adversary to explain its messages completely by simply providing its quantum inputs (due to no-cloning). Thus the notion of semi-malicious security does not seem to translate well to the quantum setting. We leave further analysis for future work.

Acknowledgements

We thank the anonymous reviewers for helpful comments on our definitions, and especially for pointing out an error in our formalization of the notion of semi-malicious quantum adversaries in a previous version of the manuscript.

[46]Note that for simulating the partial decryptions, the simulator must be able to extract the associated reconstruction key. Now the simulator can extract this key efficiently because of the unique reconstruction property of the underlying QMHE scheme.

References

[ABOE08] Dorit Aharonov, Michael Ben-Or, and Elad Eban. Interactive proofs for quantum computations. *arXiv preprint arXiv:0810.5375*, 2008.

[ABOEM17] Dorit Aharonov, Michael Ben-Or, Elad Eban, and Urmila Mahadev. Interactive proofs for quantum computations. *arXiv preprint arXiv:1704.04487*, 2017.

[AJL$^+$12] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, 2012.

[AJW11] Gilad Asharov, Abhishek Jain, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold fhe. Cryptology ePrint Archive, Report 2011/613, 2011. http://eprint.iacr.org/2011/613.

[AMMR13] Matthew Amy, Dmitri Maslov, Michele Mosca, and Martin Roetteler. A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2013.

[AMTdW00] Andris Ambainis, Michele Mosca, Alain Tapp, and Ronald de Wolf. Private quantum channels. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA*, 2000.

[AS06] Pablo Arrighi and Louis Salvail. Blind quantum computation. *International Journal of Quantum Information*, 4(05):883–898, 2006.

[Bar86] D A Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in nc1. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, STOC '86, 1986.

[BDFP86] Allan Borodin, Danny Dolev, Faith E. Fich, and Wolfgang J. Paul. Bounds for width two branching programs. *SIAM J. Comput.*, 15(2):549–560, 1986.

[BFK09] Anne Broadbent, Joseph Fitzsimons, and Elham Kashefi. Universal blind quantum computation. In *Foundations of Computer Science, 2009. FOCS'09. 50th Annual IEEE Symposium on*, pages 517–526, 2009.

[BFSS13] Harry Buhrman, Serge Fehr, Christian Schaffner, and Florian Speelman. The garden-hose model. In *Innovations in Theoretical Computer Science, ITCS '13, Berkeley, CA, USA, January 9-12, 2013*, 2013.

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In *TCC '05*, volume 3378 of LNCS, pages 325—341. Springer, 2005.

[BGS13] Anne Broadbent, Gus Gutoski, and Douglas Stebila. Quantum one-time programs. In *Advances in Cryptology–CRYPTO 2013*, pages 344–360. Springer, 2013.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *ITCS*, 2012.

[BHP17] Zvika Brakerski, Shai Halevi, and Antigoni Polychroniadou. Four round secure computation without setup. *IACR Cryptology ePrint Archive*, 2017:386, 2017.

[BJ15] Anne Broadbent and Stacey Jeffery. Quantum homomorphic encryption for circuits of low t-gate complexity. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, 2015.

[BOCG$^+$06] Michael Ben-Or, Claude Crépeau, Daniel Gottesman, Avinatan Hassidim, and Adam Smith. Secure multiparty quantum computation with (only) a strict honest majority. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 249–260. IEEE, 2006.

[BP16] Zvika Brakerski and Renen Perlman. Lattice-based fully dynamic multi-key fhe with short ciphertexts. In *Annual Cryptology Conference*, pages 190–213. Springer, 2016.

[Bra12] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In *CRYPTO*, pages 868–886, 2012.

[Bra18] Zvika Brakerski. Quantum fhe (almost) as secure as classical. Cryptology ePrint Archive, Report 2018/338, 2018.

[Bro15] Anne Broadbent. Delegating private quantum computations 1 2. *Canadian Journal of Physics*, 93(9):941–946, 2015.

[BV11a] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. In *FOCS*, pages 97–106, 2011.

[BV11b] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. In *CRYPTO*, 2011.

[CGKS95] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *Foundations of Computer Science, 1995. Proceedings., 36th Annual Symposium on*, pages 41–50. IEEE, 1995.

[CGS02] Claude Crépeau, Daniel Gottesman, and Adam Smith. Secure multi-party quantum computation. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 643–652. ACM, 2002.

[Chi05] Andrew M Childs. Secure assisted quantum computation. *Quantum Information & Computation*, 5(6):456–466, 2005.

[CKGS98] Benny Chor, Eyal Kushilevitz, Oded Goldreich, and Madhu Sudan. Private information retrieval. *Journal of the ACM (JACM)*, 45(6):965–981, 1998.

[CM15] Michael Clear and Ciarán McGoldrick. Multi-identity and multi-key leveled fhe from learning with errors. In *Annual Cryptology Conference*, pages 630–656. Springer, 2015.

[CMNT11] Jean-Sébastien Coron, Avradip Mandal, David Naccache, and Mehdi Tibouchi. Fully homomorphic encryption over the integers with shorter public keys. In *CRYPTO*, pages 487–504, 2011.

[CNT12] Jean-Sébastien Coron, David Naccache, and Mehdi Tibouchi. Public key compression and modulus switching for fully homomorphic encryption over the integers. In *EUROCRYPT*, volume 7237, pages 446–464. Springer, 2012.

[CO17] Wutichai Chongchitmate and Rafail Ostrovsky. Circuit-private multi-key fhe. In *IACR International Workshop on Public Key Cryptography*, pages 241–270. Springer, 2017.

[DFPR14] Vedran Dunjko, Joseph F Fitzsimons, Christopher Portmann, and Renato Renner. Composable security of delegated quantum computation. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 406–425. Springer, 2014.

[DHRW16] Yevgeniy Dodis, Shai Halevi, Ron D Rothblum, and Daniel Wichs. Spooky encryption and its applications. In *Annual Cryptology Conference*, pages 93–122. Springer, 2016.

[DSS16] Yfke Dulek, Christian Schaffner, and Florian Speelman. Quantum homomorphic encryption for polynomial-sized circuits. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part III*, 2016.

[Elg84] Taher Elgamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *CRYPTO '84*, pages 10–18, 1984.

[FBS$^+$14] KAG Fisher, Anne Broadbent, LK Shalm, Z Yan, J Lavoie, R Prevedel, T Jennewein, and KJ Resch. Quantum computing on encrypted data. *Nature communications*, 5:3074, 2014.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 169–178, 2009.

[Gen10] Craig Gentry. Toward basing fully homomorphic encryption on worst-case hardness. In *CRYPTO*, volume 6223, pages 116–137. Springer, 2010.

[GH11a] Craig Gentry and Shai Halevi. Fully homomorphic encryption without squashing using depth-3 arithmetic circuits. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 107–109. IEEE, 2011.

[GH11b] Craig Gentry and Shai Halevi. Implementing gentry's fully-homomorphic encryption scheme. In *EUROCRYPT*, volume 6632, pages 129–148. Springer, 2011.

[GHS12a] Craig Gentry, Shai Halevi, and Nigel P Smart. Fully homomorphic encryption with polylog overhead. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 465–482. Springer, 2012.

[GHS12b] Craig Gentry, Shai Halevi, and Nigel P Smart. Homomorphic evaluation of the aes circuit. In *Advances in Cryptology–CRYPTO 2012*, pages 850–867. Springer, 2012.

[GM84] S. Goldwasser and S. Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *STOC '87*, pages 218–229, 1987.

[Got98] Daniel Gottesman. Theory of fault-tolerant quantum computation. *Physical Review A*, 57(1):127, 1998.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, pages 75–92, 2013.

[IP07] Yuval Ishai and Anat Paskin. Evaluating branching programs on encrypted data. *Theory of Cryptography*, 2007.

[KO97] Eyal Kushilevitz and Rafail Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval. In *Foundations of Computer Science, 1997. Proceedings., 38th Annual Symposium on*, pages 364–373. IEEE, 1997.

[KP16] Elham Kashefi and Anna Pappa. Blind multiparty quantum computing. *arXiv preprint arXiv:1606.09200*, 2016.

[LATV12] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *STOC*, pages 1219–1234, 2012.

[Lia13] Min Liang. Symmetric quantum fully homomorphic encryption with perfect security. *Quantum information processing*, 12(12):3675–3687, 2013.

[Mah17] Urmila Mahadev. Classical homomorphic encryption for quantum circuits. *arXiv preprint arXiv:1708.02130*, 2017.

[MW16] Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key fhe. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 2016.

[NRS01] Gabriele Nebe, Eric M Rains, and Neil JA Sloane. The invariants of the clifford groups. *Designs, Codes and Cryptography*, 24(1):99–122, 2001.

[OTF15] Yingkai Ouyang, Si-Hui Tan, and Joseph Fitzsimons. Quantum homomorphic encryption from quantum codes. *arXiv preprint arXiv:1508.00938*, 2015.

[Pai99] Pascal Paillier. Public key cryptosystems based on composite degree residuosity classes. In *Proceedings of Eurocrypt '99*, volume 1592 of LNCS, pages 223–238, 1999.

[PS16] Chris Peikert and Sina Shiehian. Multi-key fhe from lwe, revisited. In *Theory of Cryptography Conference*, pages 217–238. Springer, 2016.

[RAD78] Ron Rivest, Leonard Adleman, and Michael L. Dertouzos. On data banks and privacy homomorphisms. In *Foundations of Secure Computation*, pages 169–180, 1978.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 84–93, 2005.

[RFG12] Peter P Rohde, Joseph F Fitzsimons, and Alexei Gilchrist. Quantum walks with encrypted data. *Physical review letters*, 109(15):150501, 2012.

[Spe16] Florian Speelman. Instantaneous non-local computation of low t-depth quantum circuits. In *11th Conference on the Theory of Quantum Computation, Communication and Cryptography, TQC 2016, September 27-29, 2016, Berlin, Germany*, 2016.

[SYY99] Tomas Sander, Adam Young, and Moti Yung. Non-interactive cryptocomputing for nc1. In *FOCS '99*, page 554, Washington, DC, USA, 1999. IEEE Computer Society.

[TKO$^+$16] Si-Hui Tan, Joshua A Kettlewell, Yingkai Ouyang, Lin Chen, and Joseph F Fitzsimons. A quantum approach to homomorphic encryption. *Scientific reports*, 6, 2016.

[vDGHV10] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In *EUROCRYPT*, pages 24–43, 2010.

[Yao82] Andrew C Yao. Protocols for secure computations. In *Foundations of Computer Science, 1982. SFCS'08. 23rd Annual Symposium on*, pages 160–164, 1982.

[Yao86] Andrew Yao. How to generate and exchange secrets. In *FOCS*, pages 162–167, 1986.

[YPDF14] Li Yu, Carlos A Pérez-Delgado, and Joseph F Fitzsimons. Limitations on information-theoretically-secure quantum homomorphic encryption. *Physical Review A*, 90(5):050303, 2014.