**Classical Homomorphic Encryption for Quantum Circuits**

# 1 Functionality Description

Delegated Computation is the task of assigning quantum computation to an untrusted device while maintaining privacy of the computation. It can be done via classical online/offline and quantum online/offline communication. Following description deals with a method which involves fully classical offline and no quantum communication, which uses only classical Homomorphic Encryption (FHE) to evaluate quantum circuits homomorphically for classical input/input. It allows a fully classical Client to hide her data such that Server can carry out any arbitrary quantum computation on the encrypted data without having any knowledge about Client’s inputs. It hides the output and input of the computation while Server is allowed to choose the unitary operation for required computation. Quantum offline communication would be required if Client’s input and output is quantum.

**Tag** Two Party, Quantum Functionality, Universal Task, Delegated Quantum computation, Superposition.

# 2 Outline

FHE presents a classical protocol with the help of which a completely classical Client could assign Server a quantum computation for her encrypted (hidden) input/output. Similar to any classical HE this scheme is divided into four steps: Key Generation generates keys for encryption, decryption and evaluation of the circuit, Encryption encodes the input into a ciphertext using encryption key, Homomorphic Evauation performs operations (imlpements the circuit) on the encrypted input using evaluation key and Decryption transforms result of the ciphertext to actual outcome of the circuit using decryption key. Following the stages of **Delegated Quantum Computation**, in preparation stage, Client encrypts (hides) her inputs from the Server who, in the computation stage, performs quantum computation by a completely classical evaluation step where applying **Clifford gates** remains a simple step as it leaves the state with only Pauli corrections which are easy to handle by QOTP, but when applying **Toffoli Gates**, it leaves the state with some Pauli corrections and Clifford gate corrections depending on the one pad key used for (QOTP) by Client. QOTP cannot deal with Clifford gate errors and hence it needs to be corrected before the operation of next gate. These Clifford gate corrections are a combination of CNOT corrections dependent on pad key and Hadamard correction independent of pad key. Applying Hadamard requires no extra information but CNOT gate errors require revelation of one pad keys. FHE deals with this problem via **Encrypted CNOT operation** using **TCF** which only needs client to prepare one-time padded superposition states. Sever thus, updates the Pauli keys accordingly and at the end of computation, sends encrypted output to the Client with updated Pauli keys. Client decrypts sent states and gets correct output in Output Correction stage. Following is an outline of the steps involved in the scheme, assuming depth of circuit (see notations used) equal to L.

* **Key Generation** Client generates L+1 classical homomorphic key sets consisting of public key, evaluation key, secret key, trapdoor information using HE.KeyGen (classical HE step). Evaluation key consists of first L pairs of secret key-trapdoor information encrypted with last L public keys such that secret key-trapdoor key pair and public key do not belong to the same key set. Evaluation key also contains this public key used to encrypt the pair.
* **Encryption** Client uses one time pad to hide her input and encrypts the pad key using a public key not used to encrypt the trapdoors and secret in the previous step. She then sends the hidden input with encypted pad key and classical evaluation key to the Server. In case of classical input Client uses the public key to encrypt her classical message and send it to the Server over classical channel.
* **Circuit Evaluation**For a quantum inputs, Server starts with the quantum one time padded state from the Client, while in case of a classical Client, Server prepare quantum states for the encrypted input. For each gate of the circuit that Server applies, he updates the encrypted Pauli encryption. In case of Toffoli gate operation, he also corrects the extra Clifford group error performing encrypted CNOT operation and then Hadamard operations on the target qubit. Operation of encrypted CNOT operation is performed using evaluation key as follows.

**Encrypted CNOT operation** This operation uses **Trapdoor Claw Free** function pairs which have the same image (output) for different pre-images(inputs) called random claw pair. Given the image it is rendered hard to find corresponding random claw without a trapdoor (inverse function). For this protocol, the HE Encryption function under a particular public key (provided in the evaluation key) is taken as one of the functions whose distribution is shifted from the other function by a natural (homomorphic) XOR operation of encrypted key bit. The functions have a common range and hence, any element in this range set would have a pre-image in the domain set of each function, together called random claw. Any pre-image pair (random claw) thus, obtained hides the pad key used for CNOT by a XOR operation. This is implied from the properties of homomorphic XOR. In simple language, if the functions are separated by encrypted pad key via a homomorphic XOR operation, so their inputs for a common output (random claw) would be separated by the (not encrypted) pad key bit. Thus, Server creates a superposition of inputs for the functions over some distribution. Next, in case of a classical input he creates a superposition of one time padded quantum state using the encrypted key. After applying the gates on qubits, for correction of CNOT gates he has two one time padded qubits as quantum state and a pad key for CNOT. The encrypted pad key was sent to the Server by the Client. For each correction, Server thus creates three registers. First has the superposition of quantum states to be operated, second has the superposition of inputs while third register has the output of the function, where function is chosen depending on the first qubit of quantum state register and input is taken from the second qubit. Hence these registers are entangled. Server, now measures the third register which reduces second register to a random claw pair as discussed before, hiding the pad key. Now, after some calculations it can be shown that if one performs Hadamard operation on the second register and then measures it, the first register is reduced to CNOT error corrected quantum state with some extra Pauli corrections. These final Pauli corrections require trapdoor information and measurement outcome of the second register (if this outcome is zero the Z correction is zero). To perform the above operation one needs ciphertext to be same throughtout the protocol and existence of a natural XOR operation. This is not known to have been achieved by a single HE together. Hence, one uses AltHE which can operate XOR for encrypted CNOT operation and HE for updation of Pauli keys. In order to do this, HE provides a conversion of ciphertext under HE to ciphertext under AltHE and vice versa. Thus, after encrypted CNOT operation, encrypted pad key bit and other measurement outcomes recrypted using public key of the evaluation key under HE. Hence, after using trapdoor functions and other information, he finds Pauli corrections encrypted under the same public key.

Server repeats the same procedure for each layer and finally sends the updated Pauli encryption and quantum one time padded output of the circuit to Client. Client uses the encrypted pad keys of both qubits related to the CNOT error and converts it to another ciphertext using AltHE.

* **Decryption** The sent Pauli corrections are updated with public key of the last evaluation key used. This is the (*L* + 1)*th* public key and hence, Client uses (*L* + 1)*th* secret key (which was not included in the evaluation keys) to decrypt the updated encryption of pad key sent by the Server. Thus, she uses the resulting pad key to undo the quantum one time pad on the sent output state.

# 3 Figure 4 Notations

* *x*˜ encryption of x
* *Lc*, depth of a layer of circuit where each layer contains clifford gates and Toffoli gates
* L, depth of the circuit (no. of layers in the circuit)
* {*pki,ski,evki,tski*}, *ith* homomorphic key set generated from HE.KeyGen(). Public key for encryption, secret key for decryption, evaluation function key, trapdoor information required for randomness recovery from ciphertexts.
* y, measurement outcome of third register
* (*µ*0*,r*0)(*µ*1*,r*1) random claw for TCF pair, for given y
* d, measurement outcome of the second register

# 5 Properties

## Parameters

* k, security parameter
* *Lc*, depth of the circuit/ no. of layers

**Adversarial Assumption**

* This protocol is secure against honest but curious adversary setting.

## Setup Assumptions

* HE is a classical leveled fully homomorphic encryption scheme which is quantum capable for depth *Lc*.
* A BQP Server can generate a superposition of inputs for the encryption function over some distribution given the public key used for encryption. The protocol takes the learning with errors assumption.

## Security Claim/ Theorems

* **Quantum Capable** A classical HE is quantum capable i.e. can perform quantum computation efficiently if there exists AltHE which can execute natural XOR operations.
* **Indistinguishability under Chosen Plaintext Attacks by adversary(IND-CPA)** The presented classical FHE scheme is CPA secure i.e. it is not possible for any polynomial time adversary to distinguish between the encrypted classical message bits 0 and 1, by **learning with errors**.
* **Compactnes** This protocol is compact i.e. decryption does not depend on the complexity of the quantum circuit.
* **Correctness** Correctness is implied from the correctness of encrypted CNOT operation.
* **Circuit Privacy** This protocol is not circuit private as both Client and Server know the quantum circuit used for performing the computation.
* **Full Homomorphism** This protocol is fully homomorphic i.e. Server can operate any quantum circuit using this protocol.
* **Circular security** This orotocol has a stronger notion of circular security where not only the secret key but also the trapdoor functions are encrypted when provided to the Server.

# 6 Pseudo-Code

**Stage 1** Client’s Preparation

* **Input:** k, L, *Lc*, classical message m, ( and Quantum Input |*ψ*i in case of quantum inputs)
* **Output:** Homomorphic key sets (*pki,evki,ski,tski*), encrypted pad key ˜*a,*˜*b* (and Quantum One time Padded Output State *XaZb* |*ψ*i in case of quantum output)

## • Key Generation (FHE.KeyGen(

1. For 1 ≤ *i* ≤ *L* + 1,
2. Client generates homomorphic key set, (*pki,evki,ski,tski*) =HE.Keygen(1*λ,*1*Lc*) The public key *pk* is *pk*1 and the secret key *sk* is *skL*+1.

The evaluation key *evk* consists of (*evk*1*,...,evkL*+1) as well as (*pki*+1*,*HE.Enc*pki*+1(*ski*), HE.Enc*pki*+1(*tski*)) for 1 ≤ *i* ≤ *L*.

## • Encryption (FHE.Enc*pk*(*m*))

1. Client chooses pad key for each message bit *a,b* ∈ {0*,*1}*λ*.
2. She then encrypts this pad key and sends it to the Server with the evaluation keys. HE.Enc*pk*1(*a,b*)),
3. She sends encrypted classical message *XaZb* |*l*i which can be represented as the classical string *a*⊕*m*. In case of quantum output Client uses pad key to hide her quantum state using QOTP (*XaZb* |*ψ*i) and then sends this hidden state to the Server alongwith the encrypted pad key.

**Stage 2** Server’s Computation

* **Input:** evaluation key (*evki*), encrypted pad key ˜*a,*˜*b* concatenation (c), one time padded message l (and Quantum One time Padded Output State in case of quantum output)
* **Output:** updated encryption of pad key ˜*a,*˜*b* (and Quantum One time Padded Output State

*Xa*˜*Z*˜*bC* |*ψ*i in case of quantum output, where C is the quantum circuit)

## • Circuit Evaluation (FHE.Eval())

1. Server creates a superposition state for the encrypted classical message and Pauli one time pads it using encrypted pad key. He applies the circuit on it as follows: Let the Circuit be denoted by C and the gates be *ci*
2. For all i, *ci* gate is applied on qubit l and the *lth* bits of pad key (˜*a*[*l*]*,*˜*b*[*l*]) are updated to

(*a*˜0[*l*]*,*˜*b*0[*l*]) as follows.

* 1. If *ci* = {*P,H,CNOT*}, a Clifford gate then //(*ciXa*[*l*]*Zb*[*l*]*ψ* = *Xa*0[*l*]*Zb*0[*l*]*ciψ*)
		1. if *ci* =H then //Hadamard Gate

(*a*˜[*l*]*,*˜*b*[*l*]) → (˜*b*[*l*]*,a*˜[*l*]) (Hadamard tranforms X gate into Z and Z into X) ii. if *ci* =P then //Pauli Gate

(*a*˜[*l*]*,*˜*b*[*l*]) → (*a*˜[*l*]*,a*˜[*l*] ⊕˜*b*[*l*])

iii. if *ci* =CNOT with m as target bit and n as control bit then (CNOT)

(*a*˜[*l*]*,*˜*b*[*l*];˜*a*[*n*]*,*˜*b*[*n*]) → (*a*˜[*l*]*,*˜*b*[*l*] ⊕˜*b*[*n*];˜*a*[*l*] ⊕ *a*˜[*n*]*,*˜*b*[*n*])

* 1. If *ci* = *T* gate then //Toffoli Gate on *lth,nth,oth* key bits
		1. The Toffoli gate is applied to the Pauli one time padded state and the state is reduced

to combination of Clifford C and Pauli P corrections as follows:

*TXa*[*l*]*Zb*[*l*]*Xa*[*n*]*Zb*[*n*]*Xa*[*o*]*Zb*[*o*] |*ψ*i

= *TXa*[*l*]*Zb*[*l*]*Xa*[*n*]*Zb*[*n*]*Xa*[*o*]*Zb*[*o*]*T* † *T* |*ψ*i

= *CNOTl,oa*[*n*]*CNOTn,oa*[*l*]*CZl,nb*[*o*]*Xa*[*l*]*Zb*[*l*]*T* |*ψ*i

= *CNOTl,oa*[*n*]*CNOTn,oa*[*l*]*HnCNOTl,nb*[*o*]*HnXa*[*l*]*Zb*[*l*]*T* |*ψ*i

= *CabPabT* |*ψ*i, where *C*{CNOT,H} and ii. The Pauli key encryptions are homomorphically updated according to *Pab*. iii. Three encrypted CNOTs are used to correct *Cab* as follows.

* + - 1. The server applies encrypted CNOT operation to the two qubit state *ZzXx* |*ψ*i using the ciphertext ˆ*c* =HE.Convert(*c*).
			2. Server generates following superposition sampled over random distribution D for the

TCF function pairs (*f*0 =AltHE.Enc*pk*()*,f*1) based on the condition *f*0 ⊕*H f*1 = *c*ˆ



* + - 1. Servers generates three register for quantum input, function input, function output and entangles them as follows:



* + - 1. Server measures the last register to get a ciphertext *y* =AltHE.Enc*pk*(*µ*0*,r*0), where *µ*0 ⊕ *µ*1 = *s*.
			2. Server performs Hadamard on second register and measures it to get a string d such that first register of input quantum state is reduced to: the following ideal state:

 (*Zd*·((*µ*0*,r*0)⊕(*µ*1*,r*1)) ⊗ *Xµ*0)CNOT (1)

where AltHE.Enc*pk*(*µ*0;*r*0) = AltHE.Enc*pk*(*µ*1;*r*1) ⊕*H c*ˆ and ⊕*H* is the homomorphic XOR operation.

* + - 1. The server uses *pki*+1 to compute HE.Enc*pki*+1(*ca,b,pki*) and HE.Enc*pki*+1(*c,y,d*ˆ ).
			2. The server computes the encryption of *a,b* under *pki*+1 by homomorphically running the decryption circuit on inputs HE*.*Enc*pki*+1(*ski*) and HE.Enc*pki*+1(*ca,b,pki*) .
			3. The server homomorphically computes (*µ*0*,r*0) and (*µ*1*,r*1), using the ciphertexts encrypting *tski,ski,c,y,d*ˆ (all encrypted with HE under public key *pki*+1). The server then uses this result, along with the ciphertexts encrypting *a,b,d*, to homomorphically compute ˜*b* = *b* + (*d* · ((*µ*0*,r*0) ⊕ (*µ*1*,r*1))*,*0) and ˜*a* = *a* + (0*,µ*0).
1. Server sends updated encryptions of Pauli corrections ˜*a,*˜*b* and the classical outcome after measurement of the output state (or Quantum one time padded state in case of quantum output) to Client.

**Stage 3** Client’s Output Correction

**Input**: Classical output state, *l* ∈ {0*,*1}*λ* (or Quantum One time padded state in case of Quantum output), encrypted Pauli corrections ˜*a,*˜*b*

**Output**: Decrypted classical message *x* ⊕ *m* (or final quantum output of computation *ZzXx* |*ψ*i)

## Decryption (FHE.Dec*sk*)

1. Client decrypts ˜*a,*˜*b* using *skL*+1 to obtain *a,b*.
2. She then uses the decrypted Pauli corrections to get the output *XaZb* |*l*i, which can be represented as *a* ⊕ *l*.

She operates *XaZb* on quantum output to get C|*ψ*i, in case of quantum output.

# 7 References 8 Requirements 9 Use Case