Quantum Fully Homomorphic Encryption with Verification

Gorjan Alagic[1,2(✉)], Yfke Dulek[3], Christian Schaffner[3], and Florian Speelman[4]

[1] Joint Center for Quantum Information and Computer Science, University of Maryland, College Park, MD, USA
galagic@gmail.com
[2] National Institute of Standards and Technology, Gaithersburg, MD, USA
[3] CWI, QuSoft, and University of Amsterdam, Amsterdam, Netherlands
[4] QMATH, Department of Mathematical Sciences, University of Copenhagen, Copenhagen, Denmark

Abstract. Fully-homomorphic encryption (FHE) enables computation on encrypted data while maintaining secrecy. Recent research has shown that such schemes exist even for quantum computation. Given the numerous applications of classical FHE (zero-knowledge proofs, secure two-party computation, obfuscation, etc.) it is reasonable to hope that quantum FHE (or QFHE) will lead to many new results in the quantum setting. However, a crucial ingredient in almost all applications of FHE is *circuit verification*. Classically, verification is performed by checking a transcript of the homomorphic computation. Quantumly, this strategy is impossible due to no-cloning. This leads to an important open question: can quantum computations be delegated and verified in a non-interactive manner?

In this work, we answer this question in the affirmative, by constructing a scheme for QFHE with verification (vQFHE). Our scheme provides authenticated encryption, and enables arbitrary polynomial-time quantum computations without the need of interaction between client and server. Verification is almost entirely classical; for computations that start and end with classical states, it is completely classical. As a first application, we show how to construct quantum one-time programs from classical one-time programs and vQFHE.

1 Introduction

The 2009 discovery of fully-homomorphic encryption (FHE) in classical cryptography is widely considered to be one of the major breakthroughs of the field. Unlike standard encryption, FHE enables non-interactive computation on encrypted data even by parties that do not hold the decryption key. Crucially, the input, output, and all intermediate states of the computation remain encrypted, and thus hidden from the computing party. While FHE has some obvious applications (e.g., cloud computing), its importance in cryptography stems from its wide-ranging applications to other cryptographic scenarios. For instance, FHE can be used to construct

T. Takagi and T. Peyrin (Eds.): ASIACRYPT 2017, Part I, LNCS 10624, pp. 438–467, 2017.
https://doi.org/10.1007/978-3-319-70694-8_16

secure two-party computation, efficient zero-knowledge proofs for NP, and indistinguishability obfuscation [4,14]. In fact, the breadth of its usefulness has led some to dub FHE "the swiss army knife of cryptography" [4].

Recent progress on constructing quantum computers has led to theoretical research on "cloud-based" quantum computing. In such a setting, it is natural to ask whether users can keep their data secret from the server that performs the quantum computation. A recently-constructed quantum fully-homomorphic encryption (QFHE) scheme shows that this can be done in a single round of interaction [12]. This discovery raises an important question: do the numerous classical applications of FHE have suitable quantum analogues? As it turns out, most of the classical applications require an additional property which is simple classically, but non-trivial quantumly. That property is *verification*: the ability of the user to check that the final ciphertext produced by the server is indeed the result of a particular computation, homomorphically applied to the initial user-generated ciphertext. In the classical case, this is a simple matter: the server makes a copy of each intermediate computation step, and provides the user with all these copies. In the quantum case, such a "transcript" would appear to violate no-cloning. The user simply checks a transcript generated by the server. In the quantum case, this would violate no-cloning. In fact, one might suspect that the no-cloning theorem prevents non-interactive quantum verification *in principle*.

In this work, we show that verification of homomorphic quantum computations is in fact possible. We construct a new QFHE scheme which allows the server to generate a "computation log" which can certify to the user that a particular homomorphic quantum computation was performed on the ciphertext. The computation log itself is purely classical, and most (in some cases, all) of the verification can be performed on a classical computer. Unlike in all previous quantum homomorphic schemes, the underlying encryption is now authenticated.

Verification immediately yields new applications of QFHE, e.g., allowing users of a "quantum cloud service" to certify the server's computations. Verified QFHE (or vQFHE) also leads to a simple construction of quantum one-time programs (qOTPs) [9]. In this construction, the qOTP for a functionality Φ consists of an evaluation key and a classical OTP which performs vQFHE verification for Φ only. Finding other applications of vQFHE (including appropriate analogues of all classical applications) is the subject of ongoing work.

Related Work. Classical FHE was first constructed by Gentry in 2009 [15]. For us, the scheme of Brakerski and Vaikuntanathan [5] is of note: it has decryption in $\mathbf{NC}^1$ and is believed to be quantum-secure. Quantumly, partially-homomorphic (or partially-compact) schemes were proposed by Broadbent and Jeffery [6]. The first fully-homomorphic (leveled) scheme was constructed by Dulek, Schaffner and Speelman [12]. Recently, Mahadev proposed a scheme, based on classical indistinguishability obfuscation, in which the user is completely classical [17]. A parallel line of work has attempted to produce QFHE with information-theoretic security [18,19,21,23]. There has also been significant research on delegating quantum computation interactively (see, e.g., [1,8,11]). Another notable inter-

active approach is quantum computation on authenticated data (QCAD), which was used to construct quantum one-time programs from classical one-time programs [9] and zero-knowledge proofs for QMA [10].

Summary of Results. Our results concern a new primitive: verified QFHE. A standard QFHE scheme consists of four algorithms: KeyGen, Enc, Eval and Dec [6,12]. We define vQFHE similarly, with two changes: (i) Eval provides an extra classical "computation log" output; (ii) decryption is now called VerDec, and accepts a ciphertext, a circuit description C, and a computation log. Informally, correctness then demands that, for all keys k and circuits C acting on plaintexts,

$$\mathsf{VerDec}_k^C \circ \mathsf{Eval}_{\mathsf{evk}}^C \circ \mathsf{Enc}_k = \Phi_C. \tag{1}$$

A crucial parameter is the relative difficulty of performing C and VerDec_k^C. In a nontrivial scheme, the latter must be simpler. In our case, C is an arbitrary poly-size quantum circuit and VerDec_k^C is almost entirely classical.

Security of verified QFHE. Informally, security should require that, if a server deviates significantly from the map Eval_k^C in (1), then VerDec_k^C will reject.

1. **Semantic security (SEM-VER).** Consider a QPT adversary $\mathcal{A}$ which manipulates a ciphertext (and side info) and declares a circuit, as in Fig. 1 (top). This defines a channel $\Phi_{\mathcal{A}} := \mathsf{VerDec} \circ \mathcal{A} \circ \mathsf{Enc}$. A simulator $\mathcal{S}$ does not receive or output a ciphertext, but does declare a circuit; this defines a channel $\Phi_{\mathcal{S}}$ which first runs $\mathcal{S}$ and then runs a circuit on the plaintext based on the outputs of $\mathcal{S}$. We say that a vQFHE scheme is semantically secure (SEM-VER) if for all adversaries $\mathcal{A}$ there exists a simulator $\mathcal{S}$ such that the channels $\Phi_{\mathcal{A}}$ and $\Phi_{\mathcal{S}}$ are computationally indistinguishable.
2. **Indistinguishability (IND-VER).** Consider the following security game. Based on a hidden coin flip b, $\mathcal{A}$ participates in one of two protocols. For $b = 0$, this is normal vQFHE. For $b = 1$, this is a modified execution, where we secretly swap out the plaintext $\rho_{\mathcal{A}}$ to a private register (replacing it with a fixed state), apply the desired circuit to $\rho_{\mathcal{A}}$, and then swap $\rho_{\mathcal{A}}$ back in; we then discard this plaintext if VerDec rejects the outputs of $\mathcal{A}$. Upon receiving the final plaintext of the protocol, $\mathcal{A}$ must guess the bit b. A vQFHE scheme is IND-VER if, for all $\mathcal{A}$, the success probability is at most $1/2 + \mathrm{negl}(n)$.
3. **New relations between security definitions.** If we restrict SEM-VER to empty circuit case, we recover (the computational version of) the definition of quantum authentication [7,13]. SEM-VER (resp., IND-VER) generalizes computational semantic security SEM (resp., indistinguishability IND) for quantum encryption [2,6]. We generalize SEM $\Leftrightarrow$ IND [2] as follows.

Theorem 1. *A vQFHE scheme satisfies SEM-VER iff it satisfies IND-VER.*

A scheme for vQFHE for poly-size quantum circuits. Our main result is a vQFHE scheme which admits verification of arbitrary polynomial-size quantum circuits. The verification in our scheme is almost entirely classical. In fact, we

can verify classical input/output computations using purely classical verification. The main technical ingredients are (i) classical FHE with $\mathbf{NC}^1$ decryption [5], (ii) the trap code for computing on authenticated quantum data [7,9,20], and (iii) the "garden-hose gadgets" from the first QFHE scheme [12]. The scheme is called TrapTP; a brief sketch is as follows.

1. **Key Generation (KeyGen).** We generate keys for the classical FHE scheme, as well as some encrypted auxiliary states (see evaluation below). This procedure requires the generation of single-qubit and two-qubit states from a small fixed set, performing Bell measurements and Pauli gates, and executing the encoding procedure of a quantum error-correcting code on which the trap code is based.
2. **Encryption (Enc).** We encrypt each qubit of the plaintext using the trap code, and encrypt the trap code keys using the FHE scheme. This again requires the ability to perform Paulis, execute an error-correcting encoding, and the generation of basic single-qubit states.
3. **Evaluation (Eval).** Paulis and CNOT are evaluated as in the trap code; keys are updated via FHE evaluation. To measure a qubit, we measure all ciphertext qubits and place the outcomes in the log. To apply P or H, we use encrypted magic states (from the eval key) plus the aforementioned gates. Applying T requires a magic state and an encrypted "garden-hose gadget" (because the T-gate magic state circuit applies a P-gate conditioned on a measurement outcome). In addition to all of the measurement outcomes, the log also contains a transcript of all the classical FHE computations.
4. **Verified decryption (VerDec).** We check the correctness and consistency of the classical FHE transcript, the measurement outcomes, and the claimed circuit. The result of this computation is a set of keys for the trap code, which are correct provided that Eval was performed honestly. We decrypt using these keys and output either a plaintext or reject. In terms of quantum capabilities, decryption requires executing the decoding procedure of the error-correcting code, computational-basis and Hadamard-basis measurements, and Paulis.

Our scheme is *compact*: the number of elementary quantum operations performed by VerDec scales only with the size of the plaintext, and *not* with the size of the circuit performed via Eval. We do require that VerDec performs a classical computation which can scale with the size of the circuit; this is reasonable since VerDec must receive the circuit as input. Like the other currently-known schemes for QFHE, our scheme is leveled, in the sense that pre-generated auxiliary magic states are needed to perform the evaluation procedure.

Theorem 2 (Main result, informal). *Let* TrapTP *be the scheme outlined above, and let* $\mathsf{VerDec}^{\equiv}$ *be* VerDec *for the case of verifying the empty circuit.*

1. *The vQFHE scheme* TrapTP *satisfies IND-VER security.*
2. *The scheme* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{VerDec}^{\equiv})$ *is authenticating* [13] *and IND-CPA* [6].

Application: quantum one-time programs. A one-time program (or OTP) is a device which implements a circuit, but self-destructs after the first use. OTPs are impossible without hardware assumptions, even with quantum states; OTPs that implement quantum circuits (qOTP) can be built from classical OTPs (cOTP) [9]. As a first application of vQFHE, we give another simple construction of qOTPs. Our construction is weaker, since it requires a computational assumption. On the other hand, it is conceptually very simple and serves to demonstrates the power of verification. In our construction, the qOTP for a quantum circuit C is simply a (vQFHE) encryption of C together with a cOTP for verifying the universal circuit. To use the resulting qOTP, the user attaches their desired input, homomorphically evaluates the universal circuit, and then plugs their computation log into the cOTP to retrieve the final decryption keys.

Preliminaries. Our exposition assumes a working knowledge of basic quantum information and the associated notation. As for the particular notation of quantum gates, the gates $(\mathsf{H}, \mathsf{P}, \mathsf{CNOT})$ generate the so-called Clifford group (which can also be defined as the normalizer of the Pauli group); it includes the Pauli gates X and Z. In order to implement arbitrary unitary operators, it is sufficient to add the T gate (also known as the $\pi/8$ gate). Finally, we can reach universal quantum computation by adding single-qubit measurements in the computational basis.

We will frequently make use of several standard cryptographic ingredients, as follows. The quantum one-time pad (QOTP) will be used for information-theoretically secret one-time encryption. In its encryption phase, two bits $a, b \in \{0, 1\}$ are selected at random, and the map $\mathsf{X}^a\mathsf{Z}^b$ is applied to the input, projecting it to the maximally-mixed state. We will also need the computational security notions for quantum secrecy, including indistinguishability (IND, IND-CPA) [6] and semantic security (SEM) [2]. For quantum authentication, we will refer to the security definition of Dupuis, Nielsen and Salvail [13]. We will also make frequent use of the trap code for quantum authentication, described below in Sect. 3. For a security proof and methods for interactive computation on this code, see [9]. Finally, we will also use classical fully-homomorphic encryption (FHE). In brief, an FHE scheme consists of classical algorithms $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ for (respectively) generating keys, encrypting plaintexts, homomorphically evaluating circuits on ciphertexts, and decrypting ciphertexts. We will use FHE schemes which are quantum-secure and whose Dec circuits are in $\mathbf{NC}^1$ (see, e.g., [5]).

2 A New Primitive: Verifiable QFHE

We now define verified quantum fully-homomorphic encryption (or vQFHE), in the symmetric-key setting. The public-key case is a straightforward modification.

Basic Definition. The definition has two parameters: the class $\mathcal{C}$ of circuits which the user can verify, and the class $\mathcal{V}$ of circuits which the user needs to perform in order to verify. We are interested in cases where $\mathcal{C}$ is stronger than $\mathcal{V}$.

Definition 1 (vQFHE). *Let $\mathcal{C}$ and $\mathcal{V}$ be (possibly infinite) collections of quantum circuits. A $(\mathcal{C},\mathcal{V})$-vQFHE scheme is a set of four QPT algorithms:*

- $\mathsf{KeyGen} : \{1\}^\kappa \to \mathcal{K} \times \mathfrak{D}(\mathcal{H}_E)$ *(security parameter → private key, eval key);*
- $\mathsf{Enc} : \mathcal{K} \times \mathfrak{D}(\mathcal{H}_X) \to \mathfrak{D}(\mathcal{H}_C)$ *(key, ptext → ctext);*
- $\mathsf{Eval} : \mathcal{C} \times \mathfrak{D}(\mathcal{H}_{CE}) \to \mathcal{L} \times \mathfrak{D}(\mathcal{H}_C)$ *(circuit, eval key, ctext → log, ctext);*
- $\mathsf{VerDec} : \mathcal{K} \times \mathcal{C} \times \mathcal{L} \times \mathfrak{D}(\mathcal{H}_C) \to \mathfrak{D}(\mathcal{H}_X) \times \{\mathsf{acc},\mathsf{rej}\}$

such that (i) the circuits of VerDec *belong to the class $\mathcal{V}$, and (ii) for all $(sk,\rho_{\mathsf{evk}}) \leftarrow \mathsf{KeyGen}$, all circuits $c \in \mathcal{C}$, and all $\rho \in \mathfrak{D}(\mathcal{H}_{XR})$,*

$$\left\|\mathsf{VerDec}_{sk}(c,\mathsf{Eval}(c,\mathsf{Enc}_k(\rho),\rho_{\mathsf{evk}})) - \Phi_c(\rho)\otimes|\mathsf{acc}\rangle\langle\mathsf{acc}|)\right\|_1 \leq \mathrm{negl}(\kappa),$$

where R is a reference and the maps implicitly act on appropriate spaces.

We will refer to condition (ii) as *correctness*. It is implicit in the definition that the classical registers $\mathcal{K},\mathcal{L}$ and the quantum registers E,X,C are really infinite families of registers, each consisting of $\mathrm{poly}(\kappa)$-many (qu)bits. In some later definitions, it will be convenient to use a version of VerDec which also outputs a copy of the (classical) description of the circuit c.

Compactness. We note that there are trivial vQFHE schemes for some choices of $(\mathcal{C},\mathcal{V})$ (e.g., if $\mathcal{C}\subset\mathcal{V}$, then the user can simply authenticate the ciphertext and then perform the computation during decryption). Earlier work on quantum and classical homomorphic encryption required compactness, meaning that the size of the decrypt circuit should not scale with the size of the homomorphic circuit.

Definition 2 (Compactness of QFHE). *A QFHE scheme S is compact if there exists a polynomial $p(\kappa)$ such that for any circuit C with n_{out} output qubits, and for any input ρ_X, the complexity of applying $S.\mathsf{Dec}$ to $S.\mathsf{Eval}^C(S.\mathsf{Enc}_{sk}(\rho_X),\rho_{evk})$ is at most $p(n_{\mathsf{out}},\kappa)$.*

When considering QFHE *with* verification, however, some tension arises. On one hand, trivial schemes like the above still need to be excluded. On the other hand, verifying that a circuit C has been applied requires reading a description of C, which violates Definition 2. We thus require a more careful consideration of the relationship between the desired circuit $C\in\mathcal{C}$ and the verification circuit $V\in\mathcal{V}$. In our work, we will allow the number of classical gates in V to scale with the size of C. We propose a new definition of compactness in this context.

Definition 3 (Compactness of vQFHE (informal)). *A vQFHE scheme S is compact if $S.\mathsf{VerDec}$ is divisible into a classical verification procedure $S.\mathsf{Ver}$ (outputting only an accept/reject flag), followed by a quantum decryption procedure $S.\mathsf{Dec}$. The running time of $S.\mathsf{Ver}$ is allowed to depend on the circuit size, but the running time of $S.\mathsf{Dec}$ is not.*

The procedure $S.\mathsf{Dec}$ is not allowed to receive and use any other information from $S.\mathsf{Ver}$ than whether or not it accepts or rejects. This prevents the classical procedure $S.\mathsf{Ver}$ from de facto performing part of the decryption work (e.g., by computing classical decryption keys). In Sect. 3, we will see a scheme that does not fulfill compactness for this reason.

Definition 4 (Compactness of vQFHE (formal)). *A vQFHE scheme S is compact if there exists a polynomial p such that $S.\mathsf{VerDec}$ can be written as $S.\mathsf{Dec} \circ S.\mathsf{Ver}$, and the output ciphertext space $\mathfrak{D}(\mathcal{H}_C)$ can be written as a classical-quantum state space $\mathcal{A} \times \mathfrak{D}(\mathcal{H}_B)$, where (i.) $S.\mathsf{Ver} : \mathcal{K} \times \mathcal{C} \times \mathcal{L} \times \mathcal{A} \to \{\mathsf{acc}, \mathsf{rej}\}$ is a classical polynomial-time algorithm, and (ii.) $S.\mathsf{Dec} : \{\mathsf{acc}, \mathsf{rej}\} \times \mathcal{K} \times \mathfrak{D}(\mathcal{H}_C) \to \mathfrak{D}(\mathcal{H}_X) \times \{\mathsf{acc}, \mathsf{rej}\}$ is a quantum algorithm such that for any circuit C with n_{out} output qubits and for any input ρ_X, $\mathcal{S}.\mathsf{Dec}$ runs in time $p(n_{\mathsf{out}}, \kappa)$ on the output of $S.\mathsf{Eval}^C(S.\mathsf{Enc}(\rho_X), \rho_{evk})$.*

Note that in the above definition, the classical registers $\mathcal{K}$ and $\mathcal{A}$ are copied and fed to both $S.\mathsf{Dec}$ and $S.\mathsf{Ver}$.

For privacy, we say that a vQFHE scheme is private if its ciphertexts are indistinguishable under chosen plaintext attack (IND-CPA) [6,12].

Secure Verifiability. In this section, we formalize the concept of verifiability. Informally, one would like the scheme to be such that whenever VerDec accepts, the output can be trusted to be close to the desired output. We will consider two formalizations of this idea: a semantic one, and an indistinguishability-based one.

Our semantic definition will state that every adversary with access to the ciphertext can be simulated by a simulator that only has access to an ideal functionality that simply applies the claimed circuit. It is inspired by quantum authentication [7,13] and semantic secrecy [2].

The real-world scenario (Fig. 1, top) begins with a state $\rho_{XR_1R_2}$ prepared by a QPT ("message generator") $\mathcal{M}$. The register X (plaintext) is subsequently encrypted and sent to the adversary $\mathcal{A}$. The registers R_1 and R_2 contain side information. The adversary acts on the ciphertext and R_1, producing some output ciphertext $C_{X'}$, a circuit description c, and a computation log *log*. These outputs are then sent to the verified decryption function. The output, along with R_2, is sent to a distinguisher $\mathcal{D}$, who produces a bit 0 or 1.

In the ideal-world scenario (Fig. 1, bottom), the plaintext X is not encrypted or sent to the simulator $\mathcal{S}$. The simulator outputs a circuit c and chooses whether to accept or reject. The channel Φ_c implemented by c is applied to the input register X directly. If reject is chosen, the output register X' is traced out and replaced by the fixed state Ω; this controlled-channel is denoted $\overline{\mathsf{ctrl}}\text{-}\oslash$.

Definition 5 (κ-SEM-VER). *A vQFHE scheme* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{VerDec})$ *is* semantically *κ-verifiable if for any QPT adversary $\mathcal{A}$, there exists a QPT $\mathcal{S}$ such that for all QPTs $\mathcal{M}$ and $\mathcal{D}$,*

$$\left| \Pr\left[\mathcal{D}\Big(\mathsf{Real}^{\mathcal{A}}_{sk}(\mathcal{M}(\rho_{evk}))\Big) = 1\right] - \Pr\left[\mathcal{D}\Big(\mathsf{Ideal}^{\mathcal{S}}_{sk}(\mathcal{M}(\rho_{evk}))\Big) = 1\right] \right| \leq \mathrm{negl}(\kappa),$$

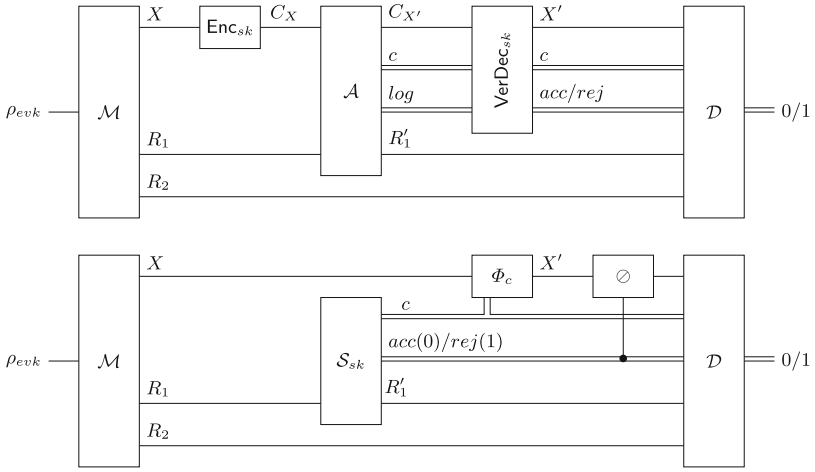


Fig. 1. The real-world (top) and ideal-world (bottom) for SEM-VER.

where $\mathsf{Real}^{\mathcal{A}}_{sk} = \mathsf{VerDec}_{sk} \circ \mathcal{A} \circ \mathsf{Enc}_{sk}$ *and* $\mathsf{Ideal}^{\mathcal{S}}_{sk} = \overline{\mathsf{ctrl}}\text{-}\oslash \circ \Phi_c \circ \mathcal{S}_{sk}$*, and the probability is taken over* $(\rho_{\mathsf{evk}}, sk) \leftarrow \mathsf{KeyGen}(1^\kappa)$ *and all QPTs above.*

Note that the simulator (in the ideal world) gets the secret key sk. We believe that this is necessary, because the actions of an adversary may depend on superficial properties of the ciphertext. In order to successfully simulate this, the simulator needs to be able to generate (authenticated) ciphertexts. He cannot do so with a fresh secret key, because the input plaintext may depend on the correlated evaluation key ρ_{evk}. Fortunately, the simulator does not become too powerful when in possession of the secret key, because he does not receive any relevant plaintexts or ciphertexts to encrypt or decrypt: the input register X is untouchable for the simulator.

Next, we present an alternative definition of verifiability, based on a security game motivated by indistinguishability.

Game 1. *For an adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$*, a scheme* S*, and a security parameter* κ*, the* $\mathsf{VerGame}_{\mathcal{A},S}(\kappa)$ *game proceeds as depicted in Fig. 2.*

The game is played in several rounds. Based on the evaluation key, the adversary first chooses an input (and some side information in R). Based on a random bit b this input is either encrypted and sent to $\mathcal{A}_2$ (if $b = 0$), or swapped out and replaced by a dummy input $|0^n\rangle\langle 0^n|$ (if $b = 1$). If $b = 1$, the ideal channel Φ_c is applied by the challenger, and the result is swapped back in right before the adversary (in the form of $\mathcal{A}_3$) has to decide on its output bit b'. If $\mathcal{A}_2$ causes a reject, the real result is also erased by the channel $\oslash$. We say that the adversary *wins* (expressed as $\mathsf{VerGame}_{\mathcal{A},S}(\kappa) = 1$) whenever $b' = b$.

Definition 6. (κ-IND-VER). *A vQFHE scheme* S *has* κ*-indistinguishable verification if for any QPT adversary* $\mathcal{A}$*,* $\Pr[\mathsf{VerGame}_{\mathcal{A},S}(\kappa) = 1] \leq \frac{1}{2} + \mathrm{negl}(\kappa)$.

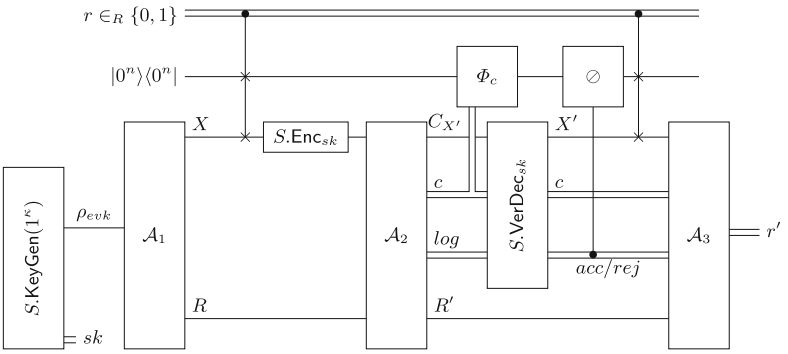


Fig. 2. The indistinguishability game $\mathsf{VerGame}_{\mathcal{A},S}(\kappa)$, as used in the definition of κ-IND-VER.

Theorem 3. *A vQFHE scheme is κ-IND-VER iff it is κ-SEM-VER.*

Proof (sketch). The forward direction is shown by contraposition. Given an adversary $\mathcal{A}$, define a simulator $\mathcal{S}$ that encrypts a dummy 0-state, then runs $\mathcal{A}$, and then VerDec. For this simulator, there exist $\mathcal{M}$ and $\mathcal{D}$ such that the difference in acceptance probability between the real and the ideal scenario is nonnegligible. The triple $(\mathcal{M}, \mathcal{A}, \mathcal{D})$ forms an adversary for the VER indistinguishability game.

For the reverse direction, we use the following approach. From an arbitrary adversary $\mathcal{A}$ for the IND-VER indistinguishability game, we define a semantic adversary, message generator, and distinguisher, that together simulate the game for $\mathcal{A}$. The fact that S is κ-SEM-VER allows us to limit the advantage of the semantic adversary over any simulator, and thereby the winning probability of $\mathcal{A}$.

For a detailed proof, see the full version [3]. □

3 TC: A partially-homomorphic scheme with verification

We now present a partially-homomorphic scheme with verification, which will serve as a building block for the fully-homomorphic scheme in Sect. 4. It is called TC (for "trap code"), and is homomorphic only for CNOT, (classically controlled) Paulis, and measurement in the computational and Hadamard basis. It does not satisfy compactness: as such, it performs worse than the trivial scheme where the client performs the circuit at decryption time. However, TC lays the groundwork for the vQFHE scheme we present in Sect. 4, and as such is important to understand in detail. It is a variant of the trap-code scheme presented in [9] (which requires classical interaction for T gates), adapted to our vQFHE framework. A variation also appears in [10], and implicitly in [20].

Setup and Encryption. Let CSS be a (public) self-dual $[[m, 1, d]]$ CSS code, so that H and CNOT are transversal. CSS can correct d_c errors, where $d = 2d_c + 1$.

We choose $m = \text{poly}(d)$ and large enough that $d_c = \kappa$ where κ is the security parameter. The concatenated Steane code satisfies all these requirements.

We generate the keys as follows. Choose a random permutation $\pi \in_R S_{3m}$ of $3m$ letters. Let n be the number of qubits that will be encrypted. For each $i \in \{1, \ldots, n\}$, pick bit strings $x[i] \in_R \{0,1\}^{3m}$ and $z[i] \in_R \{0,1\}^{3m}$. The secret key sk is the tuple $(\pi, x[1], z[1], \ldots, x[n], z[n])$, and ρ_{evk} is left empty.

Encryption is per qubit: (i) the state σ is encoded using CSS, (ii) m computational and m Hadamard 'traps' ($|0\rangle$ and $|+\rangle$ states, see [9]) are added, (iii) the resulting $3m$ qubits are permuted by π, and (iv) the overall state is encrypted with a quantum one-time pad (QOTP) as dictated by $x = x[i]$ and $z = z[i]$ for the ith qubit. We denote the ciphertext by $\widetilde{\sigma}$.

Evaluation. First, consider Pauli gates. By the properties of CSS, applying a logical Pauli is done by applying the same Pauli to all physical qubits. The application of Pauli gates (X and/or Z) to a state encrypted with a quantum one-time pad can be achieved without touching the actual state, by updating the keys to QOTP in the appropriate way. This is a classical task, so we can postpone the application of the Pauli to VerDec (recall it gets the circuit description) without giving up compactness for TC. So, formally, the evaluation procedure for Pauli gates is the identity map. Paulis conditioned on a classical bit b which will be known to VerDec at execution time (e.g., a measurement outcome) can be applied in the same manner.

Next, we consider CNOT. To apply a CNOT to encrypted qubits σ_i and σ_j, we apply CNOT transversally between the $3m$ qubits of $\widetilde{\sigma_i}$ and the $3m$ qubits of $\widetilde{\sigma_j}$. Ignoring the QOTP for the moment, the effect is a transversal application of CNOT on the pysical data qubits (which, by CSS properties, amounts to logical CNOT on $\sigma_i \otimes \sigma_j$), and an application of CNOT between the $2m$ pairs of trap qubits. Since $\mathsf{CNOT}|00\rangle = |00\rangle$ and $\mathsf{CNOT}|{+}{+}\rangle = |{+}{+}\rangle$, the traps are unchanged. Note that CNOT commutes with the Paulis that form the QOTP. In particular, for all $a, b, c, d \in \{0,1\}$, $\mathsf{CNOT}(\mathsf{X}_1^a \mathsf{Z}_1^b \otimes \mathsf{X}_2^c \mathsf{Z}_2^d) = (\mathsf{X}_1^a \mathsf{Z}_1^{b \oplus d} \otimes \mathsf{X}_2^{a \oplus c} \mathsf{Z}_2^d)\mathsf{CNOT}$. Thus, updating the secret-key bits (a, b, c, d) to $(a, b \oplus d, a \oplus c, d)$ finishes the job. The required key update happens in $\mathsf{TC.VerDec}$ (see below).

Next, consider computational-basis measurements. For CSS, logical measurement is performed by measurement of all physical qubits, followed by a classical decoding procedure [9]. In $\mathsf{TC.Eval}$, we measure all $3m$ ciphertext qubits. During $\mathsf{TC.VerDec}$, the contents of the measured qubits (now a classical string $a \in \{0,1\}^{3m}$) will be interpreted into a logical measurement outcome.

Finally, we handle Hadamard-basis measurements. A transversal application of H to all $3m$ relevant physical qubits precedes the evaluation procedure for the computational basis measurement. Since CSS is self-dual, this applies a logical H. Since $\mathsf{H}|0\rangle = |+\rangle$ and $\mathsf{H}|+\rangle = |0\rangle$, all computational traps are swapped with the Hadamard traps. This is reflected in the way $\mathsf{TC.VerDec}$ checks the traps (see the full version [3] for details). Note that this is a classical procedure (and thus its accept/reject output flag is classical).

Verification and Decryption. If a qubit is unmeasured after evaluation (as stated in the circuit), TC.VerDecQubit is applied. This removes the QOTP, undoes the permutation, checks all traps, and decodes the qubit. See the full version [3] for a specification of this algorithm.

If a qubit is measured during evaluation, TC.VerDec receives a list $\widetilde{w}$ of $3m$ physical measurement outcomes for that qubit. These outcomes are classically processed (removing the QOTP by flipping bits, undoing π, and decoding CSS) to produce the plaintext measurement outcome. Note that we only check the $|0\rangle$ traps in this case. Intuitively, this should not affect security, since any attack that affects only $|+\rangle$ but not $|0\rangle$ will be canceled by computational basis measurement.

The complete procedure TC.VerDec updates the QOTP keys according to the gates in the circuit description, and then decrypts all qubits and measurement results as described above (see the full version [3] for details).

Correctness, Compactness, and Privacy. For honest evaluation, TC.VerDec accepts with probability 1. Correctness is straightforward to check by following the description in Sect. 3. For privacy, note that the final step in the encryption procedure is the application of a (information-theoretically secure) QOTP with fresh, independent keys. If IND-CPA security is desired, one could easily extend TC by using a pseudorandom function for the QOTP, as in [2].

TC is not compact in the sense of Definition 4, however. In order to compute the final decryption keys, the whole gate-by-gate key update procedure needs to be executed, aided by the computation log and information about the circuit. Thus, we cannot break TC.VerDec up into two separate functionalities, Ver and Dec, where Dec can successfully retrieve the keys and decrypt the state, based on only the output ciphertext and the secret key.

Security of Verification. The trap code is proven secure in its application to one-time programs [9]. Broadbent and Wainewright proved authentication security (with an explicit, efficient simulator) [7]. One can use similar strategies to prove κ-IND-VER for TC. In fact, TC satisfies a stronger notion of verifiability, where the adversary is allowed to submit plaintexts in multiple rounds (letting the choice of the next plaintext depend on the previous ciphertext), which are either all encrypted or all swapped out. Two rounds (κ-IND-VER-2) are sufficient for us; the definitions and proof (see the full version [3]) extend straightforwardly to the general case κ-IND-VER-i for $i \in \mathbb{N}_+$.

Theorem 4. TC *is κ-IND-VER-2 for the above circuit class.*

4 TrapTP: Quantum FHE With Verification

In this section, we introduce our candidate scheme for verifiable quantum fully homomorphic encryption (vQFHE). In this section, we will define the scheme prove correctness, compactness, and privacy. We will show verifiability in Sect. 5.

Let $\kappa \in \mathbb{N}$ be a security parameter, and let $t, p, h \in \mathbb{N}$ be an upper bound on the number of T, P, and H gates (respectively) that will be in the circuit which is to be homomorphically evaluated. As in Sect. 3, we fix a self-dual $[[m, 1, d]]$ CSS code CSS which has $m = \mathrm{poly}(d)$ and can correct $d_c := \kappa$ errors (e.g., the concatenated Steane code). We also fix a classical fully homomorphic public-key encryption scheme HE with decryption in $\mathsf{LOGSPACE}$ (see, e.g., [5]). Finally, fix a message authentication code $\mathsf{MAC} = (\mathsf{Tag}, \mathsf{Ver})$ that is existentially unforgeable under adaptive chosen message attacks (EUF-CMA [16]) from a quantum adversary; for example, one may take the standard pseudorandom-function construction with a post-quantum PRF. This defines an authentication procedure $\mathsf{MAC.Sign}_k : m \mapsto (m, \mathsf{MAC.Tag}_k(m))$.

Key Generation and Encryption. The evaluation key will require a number of auxiliary states, which makes the key generation algorithm $\mathsf{TrapTP.KeyGen}$ somewhat involved (see Algorithms 1 and 2). Note that non-evaluation keys are generated first, and then used to encrypt auxiliary states which are included in the evaluation key (see $\mathsf{TrapTP.Enc}$ below). Most states are encrypted using the same 'global' permutation π, but all qubits in the error-correction gadget (except first and last) are encrypted using independent permutations π_i (see line 15). The T-gate gadgets are prepared by Algorithm 2, making use of garden-hose gadgets from [12].

Algorithm 1. $\mathsf{TrapTP.KeyGen}(1^\kappa, 1^t, 1^p, 1^h)$

1: $k \leftarrow \mathsf{MAC.KeyGen}(1^\kappa)$
2: $\pi \leftarrow_R S_{3m}$ ▷ S_{3m} is the permutation group on $3m$ elements
3: **for** i = 0, ..., t **do**
4: $\quad (sk_i, pk_i, evk_i) \leftarrow \mathsf{HE.KeyGen}(1^\kappa)$
5: $sk \leftarrow (\pi, k, sk_0, ..., sk_t, pk_0)$
6: **for** i = 1, ..., p **do** ▷ Magic-state generation for P
7: $\quad \mu_i^{\mathsf{P}} \leftarrow \mathsf{TrapTP.Enc}(sk, \mathsf{P}|+\rangle)$ ▷ See Algorithm 3 for $\mathsf{TrapTP.Enc}$
8: **for** i = 1, ..., t **do** ▷ Magic-state generation for T
9: $\quad \mu_i^{\mathsf{T}} \leftarrow \mathsf{TrapTP.Enc}(sk, \mathsf{T}|+\rangle)$
10: **for** i = 1, ..., h **do** ▷ Magic-state generation for H
11: $\quad \mu_i^{\mathsf{H}} \leftarrow \mathsf{TrapTP.Enc}(sk, \frac{1}{\sqrt{2}}(\mathsf{H} \otimes \mathbb{I})(|00\rangle + |11\rangle))$
12: **for** i = 1, ..., t **do** ▷ Gadget generation for T
13: $\quad \pi_i \leftarrow_R S_{3m}$
14: $\quad (g_i, \gamma_i^{\mathsf{in}}, \gamma_i^{\mathsf{mid}}, \gamma_i^{\mathsf{out}}) \leftarrow \mathsf{TrapTP.GadgetGen}(sk_{i-1})$ ▷ See Algorithm 2
15: $\quad \Gamma_i \leftarrow \mathsf{MAC.Sign}(\mathsf{HE.Enc}_{pk_i}(g_i, \pi_i)) \otimes \mathsf{TrapTP.Enc}((\pi_i, k, sk_0, ..., sk_t, pk_i), \gamma_i^{\mathsf{mid}}) \otimes \mathsf{TrapTP.Enc}(sk, \gamma_i^{\mathsf{in}}, \gamma_i^{\mathsf{out}})$
16: $keys \leftarrow \mathsf{MAC.Sign}(evk_0, ..., evk_t, pk_0, ..., pk_t, \mathsf{HE.Enc}_{pk_0}(\pi))$
17: $\rho_{evk} \leftarrow (keys, \mu_0^{\mathsf{P}}, ..., \mu_p^{\mathsf{P}}, \mu_0^{\mathsf{T}}, ..., \mu_t^{\mathsf{T}}, \mu_0^{\mathsf{H}}, ..., \mu_h^{\mathsf{H}}, \Gamma_1, ..., \Gamma_t)$
18: **return** (sk, ρ_{evk})

Algorithm 2. $\mathsf{TrapTP.GadgetGen}(sk_i)$

1: $g_i \leftarrow g(sk_i)$ ▷ classical description of the garden-hose gadget, see [12], p. 13
2: $(\gamma_i^{\mathsf{in}}, \gamma_i^{\mathsf{mid}}, \gamma_i^{\mathsf{out}}) \leftarrow$ generate $|\Phi^+\rangle$ states and arrange them as described by g_i. Call the first qubit γ_i^{in} and the last qubit γ_i^{out}. The rest forms the state γ_i^{mid}.
3: **return** $(g_i, \gamma_i^{\mathsf{in}}, \gamma_i^{\mathsf{mid}}, \gamma_i^{\mathsf{out}})$

The encryption of a quantum state is similar to $\mathsf{TC.Enc}$, only the keys to the QOTP are now chosen during encryption (rather than during key generation) and appended in encrypted and authenticated form to the ciphertext (see Algorithm 3). Note that the classical secret keys sk_0 through sk_t are not used.

Algorithm 3. $\mathsf{TrapTP.Enc}((\pi, k, sk_0, ..., sk_t, pk), \sigma)$

1: $\widetilde{\sigma} \leftarrow \sum_{x,z\in\{0,1\}^{3m}} \left(\mathsf{TC.Enc}((\pi, x, z), \sigma) \otimes \mathsf{MAC.Sign}_k(\mathsf{HE.Enc}_{pk}(x, z))\right)$ ▷ Algorithm 13
2: **return** $\widetilde{\sigma}$

Evaluation. Evaluation of gates is analogous to the interactive evaluation scheme using the trap code [9], except the interactions are replaced by classical homomorphic evaluation. Evaluation of a circuit c is done gate-by-gate, as follows.

In general, we will use the notation $\widetilde{\cdot}$ to denote encryptions of classical and quantum states. For example, in the algorithms below, $\widetilde{\sigma}$ is the encrypted input qubit for a gate and $\widetilde{x}$ and $\widetilde{z}$ are classical encryptions of the associated QOTP keys. We will assume throughout that $\mathsf{HE.Enc}$ and $\mathsf{HE.Eval}$ produce, apart from their actual output, a complete *computation log* describing all randomness used, all computation steps, and all intermediate results.

Measurements. Computational basis measurement is described in Algorithm 4. Recall that $\mathsf{TC.VerDecMeasurement}$ is a completely classical procedure that decodes the list of $3m$ measurement outcomes into the logical outcome and checks the relevant traps. Hadamard-basis measurement is performed similarly, except the qubits of $\widetilde{\sigma}$ are measured in the Hadamard basis and $\mathsf{HE.Enc}_{pk}(\times)$ is given as the last argument for the evaluation of $\mathsf{TC.VerDecMeasurement}$.

Algorithm 4. $\mathsf{TrapTP.EvalMeasure}(\widetilde{\sigma}, \widetilde{x}, \widetilde{z}, \widetilde{\pi}, pk, evk)$

1: $a = (a_1, ..., a_{3m}) \leftarrow$ measure qubits of $\widetilde{\sigma}$ in the computational basis
2: $(\widetilde{a}, \widetilde{log_1}) \leftarrow \mathsf{HE.Enc}_{pk}(a)$
3: $(\widetilde{b}, \widetilde{flag}, \widetilde{log_2}) \leftarrow \mathsf{HE.Eval}_{evk}^{\mathsf{TC.VerDecMeasurement}}((\widetilde{\pi}, \widetilde{x}, \widetilde{z}), \widetilde{a}, \mathsf{HE.Enc}_{pk}(+))$
4: **return** $(\widetilde{b}, \widetilde{flag}, \widetilde{log_1}, \widetilde{log_2})$ ▷ $b \in \{0, 1\}$ represents the output of the measurement

Pauli gates. A logical Pauli-X is performed by (homomorphically) flipping the X-key bits of the QOTP (see Algorithm 5). Since this is a classical operation, the functionality extends straightforwardly to a classically controlled Pauli-X (by specifying an additional bit b encrypted into $\widetilde{b}$ that indicates whether or not X should be applied; see Algorithm 6). The (classically controlled) evaluation of a Pauli-Z works the same way, only the relevant bits in $\widetilde{z}$ are flipped.

Algorithm 5. $\mathsf{TrapTP.EvalX}(\widetilde{\sigma}, \widetilde{x}, \widetilde{\pi}, pk, evk)$

1: $(\widetilde{x}, log_1) \leftarrow \mathsf{HE.Eval}_{evk}^{\mathsf{unpermute}}(\widetilde{\pi}, \widetilde{x})$
2: $(\widetilde{x}, log_2) \leftarrow \mathsf{HE.Eval}_{\mathsf{evk}}^{\oplus}(\widetilde{x}, \mathsf{HE.Enc}_{pk}(1^m 0^{2m}))$ ▷ this flips the first m bits
3: $(\widetilde{x}, log_3) \leftarrow \mathsf{HE.Eval}_{evk}^{\mathsf{permute}}(\widetilde{\pi}, \widetilde{x})$
4: **return** $(\widetilde{\sigma}, \widetilde{x}, log_1, log_2, log_3)$

Algorithm 6. $\mathsf{TrapTP.EvalCondX}(\widetilde{b}, \widetilde{\sigma}, \widetilde{x}, \widetilde{z}, \widetilde{\pi}, pk, evk)$

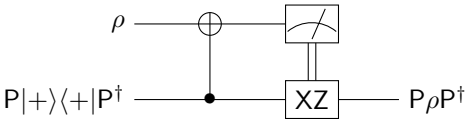
1: $(\widetilde{x}, log_1) \leftarrow \mathsf{HE.Eval}_{evk}^{\mathsf{unpermute}}(\widetilde{\pi}, \widetilde{x})$
2: $\widetilde{s} \leftarrow \mathsf{HE.Eval}_{evk}^{y \mapsto y^m 0^{2m}}(\widetilde{b})$
3: $(\widetilde{x}, log_2) \leftarrow \mathsf{HE.Eval}_{evk}^{\oplus}(\widetilde{x}, \widetilde{s})$ ▷ this conditionally flips the first m bits
4: $(\widetilde{x}, log_3) \leftarrow \mathsf{HE.Eval}_{evk}^{\mathsf{permute}}(\widetilde{\pi}, \widetilde{x})$
5: **return** $(\widetilde{\sigma}, \widetilde{x}, \widetilde{z}, log_1, log_2, log_3)$

CNOT *gates.* The evaluation of CNOT in TrapTP is analogous to TC, only the key updates are performed homomorphically during evaluation (see Algorithm 7).

Algorithm 7. $\mathsf{TrapTP.EvalCNOT}(\widetilde{\sigma_1}, \widetilde{\sigma_2}, \widetilde{x_1}, \widetilde{x_2}, \widetilde{z_1}, \widetilde{z_2}, \widetilde{\pi}, pk, evk)$

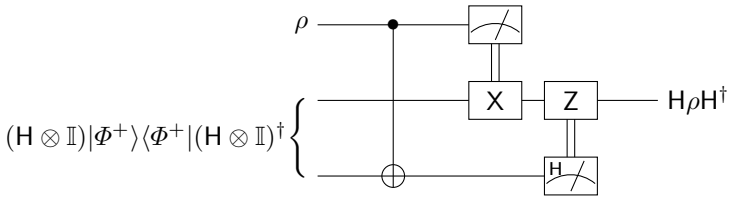
1: $(\widetilde{\sigma_1}, \widetilde{\sigma_2}) \leftarrow$ apply CNOT on all physical qubit pairs of $\widetilde{\sigma_1}, \widetilde{\sigma_2}$
2: $(\widetilde{x_1}, \widetilde{x_2}, \widetilde{z_1}, \widetilde{z_2}, log_1) \leftarrow \mathsf{HE.Eval}_{evk}^{\mathsf{CNOT-key-update}}(\widetilde{x_1}, \widetilde{x_2}, \widetilde{z_1}, \widetilde{z_2})$ ▷ for commutation rules, see Sect. 3
3: **return** $(\widetilde{\sigma_1}, \widetilde{\sigma_2}, \widetilde{x_1}, \widetilde{x_2}, \widetilde{z_1}, \widetilde{z_2}, log_1, log_2)$

Phase gates. Performing a P gate requires homomorphic evaluation of all the above gates: (classically controlled) Paulis, CNOTs, and measurements. We also consume the state μ_i^{P} (an encryption of the state $\mathsf{P}|+\rangle$) for the ith phase gate in the circuit. The circuit below applies P to the data qubit (see, e.g., [9]).

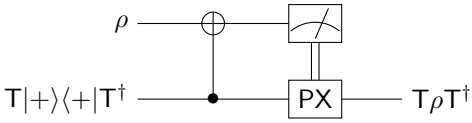


We define TrapTP.EvalP to be the concatenation of the corresponding gate evaluations. The overall computation log is just a concatenation of the logs.

Hadamard gate. The Hadamard gate can be performed using the same ingredients as the phase gate [9]. The ith gate consumes μ_i^{H}, an encryption of $(\mathsf{H}\otimes\mathbb{I})|\Phi^+\rangle$.



The T *gate.* A magic-state computation of T uses a similar circuit to that for P, using μ_i^{T} (an encryption of $\mathsf{T}|+\rangle$) as a resource for the ith T gate:



The evaluation of this circuit is much more complicated, since it requires the application of a classically-controlled phase correction P. We will accomplish this using the error-correction gadget Γ_i.

First, we remark on some subtleties regarding the encrypted classical information surrounding the gadget. Since the structure of Γ_i depends on the classical secret key sk_{i-1}, the classical information about Γ_i is encrypted under the (independent) public key pk_i (see Algorithm 1). This observation will play a crucial role in our proof that TrapTP satisfies IND-VER, in Sect. 5.

The usage of two different key sets also means that, at some point during the evaluation of a T gate, all classically encrypted information needs to be recrypted from the $(i-1)$st into the ith key set. This can be done because $\widetilde{sk}_{i-1}$ is included in the classical information g_i in Γ_i. The recryption is performed right before the classically-controlled phase gate is applied (see Algorithm 8).

Algorithm 8. $\mathsf{TrapTP.EvalT}(\widetilde{\sigma}, \widetilde{x}, \widetilde{z}, \widetilde{\pi}, \mu_i^{\mathsf{T}}, \Gamma_i, pk_{i-1}, evk_{i-1}, pk_i, evk_i)$

1: $(\widetilde{\sigma_1}, \widetilde{\sigma_2}, \widetilde{x_1}, \widetilde{z_1}, \widetilde{x_2}, \widetilde{z_2}, log_1) \leftarrow \mathsf{TrapTP.EvalCNOT}(\mu_i^{\mathsf{T}}, \widetilde{\sigma}, \widetilde{x}, \widetilde{z}, \widetilde{\pi}, pk_{i-1}, evk_{i-1})$
2: $(\widetilde{b}, log_2) \leftarrow \mathsf{TrapTP.EvalMeasure}(\widetilde{\sigma_2}, \widetilde{x_2}, \widetilde{z_2}, \widetilde{\pi}, pk_{i-1}, evk_{i-1})$
3: $log_3 \leftarrow$ recrypt *all* classically encrypted information (except $\widetilde{b}$) from key set $i-1$ into key set i.
4: $(\widetilde{\sigma}, log_4) \leftarrow \mathsf{TrapTP.EvalCondP}(\widetilde{b}, \widetilde{\sigma_1}, \widetilde{x_1}, \widetilde{z_1}, \Gamma_i, \widetilde{\pi}, pk_i, evk_i)$
5: **return** $(\widetilde{\sigma}, log_1, log_2, log_3, log_4)$

Algorithm 9 shows how to use Γ_i to apply logical P on an encrypted quantum state $\widetilde{\sigma}$, conditioned on a classical bit b for which only the encryption $\widetilde{b}$ is available. When $\mathsf{TrapTP.EvalCondP}$ is called, b is encrypted under the $(i-1)$st classical HE-key, while all other classical information (QOTP keys x and z, permutations π and π_i, classical gadget description g_i) is encrypted under the ith key. Note that we can evaluate Bell measurements using only evaluation of CNOT,

computational-basis measurements, and H-basis measurements. In particular, no magic states are needed to perform a Bell measurement. After this procedure, the data is in qubit $\widetilde{\gamma_i^{\mathsf{out}}}$. The outcomes a_1, a_2, a of the Bell measurements determine how the keys to the QOTP must be updated.

Algorithm 9. $\mathsf{TrapTP.EvalCondP}(\widetilde{b}, \widetilde{\sigma}, \widetilde{x}, \widetilde{z}, \Gamma_i = (\widetilde{g_i}, \widetilde{\pi_i}, \widetilde{\gamma_i^{\mathsf{in}}}, \widetilde{\gamma_i^{\mathsf{mid}}}, \widetilde{\gamma_i^{\mathsf{out}}}), \widetilde{\pi}, pk_i, evk_i)$

1: $(\widetilde{a_1}, \widetilde{a_2}, log_1) \leftarrow$ evaluate Bell measurement between $\widetilde{\sigma}$ and $\widetilde{\gamma_i^{\mathsf{in}}}$ ▷ $a_1, a_2 \in \{0, 1\}$
2: $(\widetilde{a}, log_2) \leftarrow$ evaluate Bell measurements in $\widetilde{\gamma_i^{\mathsf{mid}}}$ as dictated by the ciphertext $\widetilde{b}$ and the garden-hose protocol for $\mathsf{HE.Dec}$
3: $(\widetilde{x}, \widetilde{z}, log_3) \leftarrow \mathsf{HE.Eval}_{evk_i}^{\mathsf{T-key-update}}(\widetilde{x}, \widetilde{z}, \widetilde{a_1}, \widetilde{a_2}, \widetilde{a}, \widetilde{g_i})$
4: **return** $(\widetilde{\gamma_i^{\mathsf{out}}}, \widetilde{x}, \widetilde{z}, log_1, log_2, log_3)$

Verified Decryption. The decryption procedure (Algorithm 10) consists of two parts. First, we perform several classical checks. This includes MAC-verification of all classically authenticated messages, and checking that the gates listed in the log match the circuit description. We also check the portions of the log which specify the (purely classical, FHE) steps taken during $\mathsf{HE.Enc}$ and $\mathsf{HE.Eval}$; this is the standard transcript-checking procedure for FHE, which we call $\mathsf{TrapTP.CheckLog}$. Secondly, we check all unmeasured traps and decode the remaining qubits. We reject if $\mathsf{TrapTP.CheckLog}$ rejects, or if the traps have been triggered.

Algorithm 10. $\mathsf{TrapTP.VerDec}(sk, \widetilde{\sigma}, (\widetilde{x[i]})_i, (\widetilde{z[i]})_i, log, c)$

1: Verify classically authenticated messages (in log) using k (contained in sk). If one of these verifications rejects, **reject**.
2: Check whether all claimed gates in log match the structure of c. If not, **return** $(\Omega, |\text{rej}\rangle)$. ▷ Recall that Ω is a dummy state.
3: $flag \leftarrow \mathsf{TrapTP.CheckLog}(log)$ If $flag = \text{rej}$, **return** $(\Omega, |\text{rej}\rangle)$.
4: Check whether the claimed final QOTP keys in the log match $\widetilde{x}$ and $\widetilde{z}$. If not, **return** $(\Omega, |\text{rej}\rangle)$.
5: **for all** gates G of c **do**
6: **if** G is a measurement **then**
7: $\widetilde{x'}, \widetilde{z'} \leftarrow$ encrypted QOTP keys right before measurement (listed in log)
8: $\widetilde{w} \leftarrow$ encrypted measurement outcomes (listed in log)
9: $x', z', w \leftarrow \mathsf{HE.Dec}_{sk_t}(\widetilde{x'}, \widetilde{z'}, \widetilde{w})$
10: Execute $\mathsf{TC.VerDecMeasurement}((\pi, x', z'), w, basis)$, where $basis$ is the appropriate basis for the measurement, and store the (classical) outcome.
11: **if** a trap is triggered **then**
12: **return** $(\Omega, |\text{rej}\rangle)$.
13: **for all** unmeasured qubits $\widetilde{\sigma_i}$ in $\widetilde{\sigma}$ **do**
14: $x[i], z[i] \leftarrow \mathsf{HE.Dec}_{sk_t}(\widetilde{x[i]}, \widetilde{z[i]})$
15: $\sigma_i \leftarrow \mathsf{TC.VerDec}_{(\pi, x[i], z[i])}(\widetilde{\sigma_i})$. If $\mathsf{TC.VerDec}$ rejects, **return** $(\Omega, |\text{rej}\rangle)$.
16: $\sigma \leftarrow$ the list of decrypted qubits (and measurement outcomes) that are part of the output of c
17: **return** $(\sigma, |\text{acc}\rangle)$

4.1 Correctness, Compactness, and Privacy

If all classical computation was unencrypted, checking correctness of TrapTP can be done by inspecting the evaluation procedure for the different types of gates, and comparing them to the trap code construction in [9]. This suffices, since HE and the MAC authentication both satisfy correctness.

Compactness as defined in Definition 4 is also satisfied: verifying the computation log and checking all intermediate measurements (up until line 12 in Algorithm 10) is a completely classical procedure and runs in polynomial time in its input. The rest of TrapTP.VerDec (starting from line 13) only uses the secret key and the ciphertext $(\widetilde{\sigma}, \widetilde{x}, \widetilde{z})$ as input, not the log or the circuit description. Thus, we can separate TrapTP.VerDec into two algorithms Ver and Dec as described in Definition 4, by letting the second part (Dec, lines 13 to 17) reject whenever the first part (Ver, lines 1 to 12) does. It is worth noting that, because the key-update steps are performed homomorphically during the evaluation phase, skipping the classical verification step yields a QFHE scheme without verification that satisfies Definition 2 (and is authenticating). This is not the case for the scheme TC, where the classical computation is necessary for the correct decryption of the output state.

In terms of privacy, TrapTP satisfies IND-CPA (see Sect. 2). This is shown by reduction to IND-CPA of HE. This is non-trivial since the structure of the error-correction gadgets depends on the classical secret key. The reduction is done in steps, where first the security of the encryptions under pk_t is applied (no gadget depends on sk_t), after which the quantum part of the gadget Γ_t (which depends on sk_{t-1}) looks completely mixed from the point of view of the adversary. We then apply indistinguishability of the classical encryptions under pk_{t-1}, and repeat the process. After all classical encryptions of the quantum one-time pad keys are removed, the encryption of a state appears fully mixed. Full details of this proof can be found in Lemma 1 of [12], where IND-CPA security of an encryption function very similar to TrapTP.Enc is proven.

5 Proof of Verifiability for TrapTP

In this section, we will prove that TrapTP is κ-IND-VER. By Theorem 3, it then follows that TrapTP is also verifiable in the semantic sense. We will define a slight variation on the VER indistinguishability game, followed by several hybrid schemes (variations of the TrapTP scheme) that fit into this new game. We will argue that for any adversary, changing the game or scheme does not significantly affect the winning probability. After polynomially-many such steps, we will have reduced the adversary to an adversary for the somewhat homomorphic scheme TC, which we already know to be IND-VER. This will complete the argument that TrapTP is IND-VER. The IND-VER game is adjusted as follows.

Definition 7 (Hybrid game $\mathsf{Hyb}_{\mathcal{A},S}(\kappa)$). *For an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$, a scheme S, and security parameter κ, $\mathsf{Hyb}_{\mathcal{A},S}(\kappa)$ is the game in Fig. 3.*

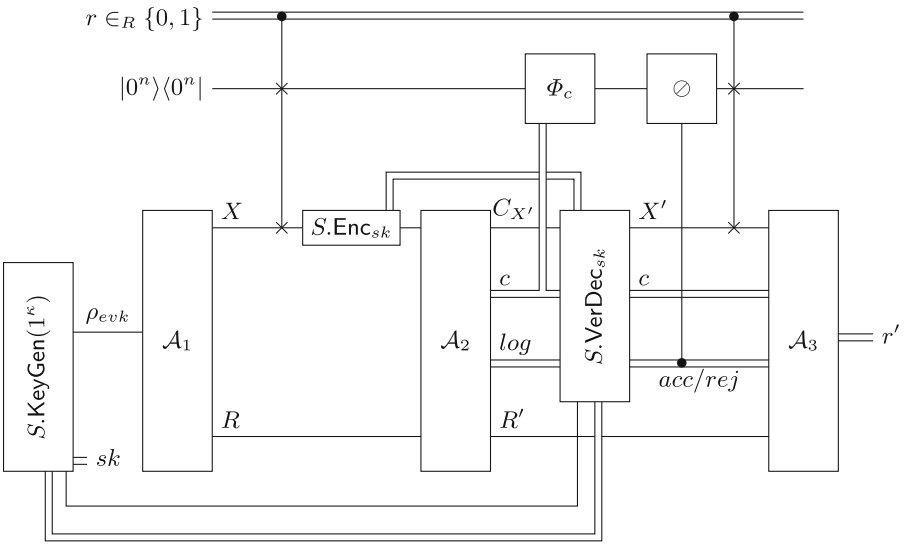


Fig. 3. The hybrid indistinguishability game $\mathsf{Hyb}_{\mathcal{A},S}(\kappa)$, which is a slight variation on $\mathsf{VerGame}_{\mathcal{A},S}(\kappa)$ from Fig. 2.

Comparing to Definition 1, we see that three new wires are added: a classical wire from $S.\mathsf{Enc}$ to $S.\mathsf{VerDec}$, and a classical and quantum wire from $S.\mathsf{KeyGen}$ to $S.\mathsf{VerDec}$. We will later adjust TrapTP to use these wires to bypass the adversary; TrapTP as defined in the previous section does not use them. Therefore, for any adversary, $\Pr[\mathsf{VerGame}_{\mathcal{A},\mathsf{TrapTP}}(\kappa) = 1] = \Pr[\mathsf{Hyb}_{\mathcal{A},\mathsf{TrapTP}}(\kappa) = 1]$.

Hybrid 1: Removing Classical MAC. In TrapTP, the initial keys to the QOTP can only become known to VerDec through the adversary. We thus use MAC to make sure these keys cannot be altered. Without this authentication, the adversary could, e.g., homomorphically use $\widetilde{\pi}$ to flip only those bits in $\widetilde{x}$ that correspond to non-trap qubits, thus applying X to the plaintext. In fact, all classical information in the evaluation key must be authenticated.

In the first hybrid, we argue that the winning probability of a QPT $\mathcal{A}$ in $\mathsf{Hyb}_{\mathcal{A},\mathsf{TrapTP}}(\kappa)$ is at most negligibly higher than in $\mathsf{Hyb}_{\mathcal{A},\mathsf{TrapTP}'}(\kappa)$, where TrapTP' is a modified version of TrapTP where the initial keys are sent directly from KeyGen and Enc to VerDec (via the extra wires above). More precisely, in $\mathsf{TrapTP}'.\mathsf{KeyGen}$ and $\mathsf{TrapTP}'.\mathsf{Enc}$, whenever $\mathsf{MAC.Sign}(\mathsf{HE.Enc}(x))$ or $\mathsf{MAC.Sign}(x)$ is called, the message x is also sent directly to $\mathsf{TrapTP}'.\mathsf{VerDec}$. Moreover, instead of decrypting the classically authenticated messages sent by the adversary, $\mathsf{TrapTP}'.\mathsf{VerDec}$ uses the information it received directly from $\mathsf{TrapTP}'.\mathsf{KeyGen}$ and $\mathsf{TrapTP}'.\mathsf{Enc}$. It still check whether the computation log provided by the adversary contains these values at the appropriate locations and whether the MAC signature is correct. The following fact is then a straightforward consequence of the EUF-CMA property of MAC.

Recall that all adversaries are QPTs, i.e., quantum polynomial-time uniform algorithms. Given two hybrid games H_1, H_2, and a QPT adversary $\mathcal{A}$, define

$$\mathsf{AdvHyb}^{H_2}_{H_1}(\mathcal{A},\kappa) := \left|\Pr[\mathsf{Hyb}_{\mathcal{A},\mathsf{H}_1}(\kappa)=1] - \Pr[\mathsf{Hyb}_{\mathcal{A},\mathsf{H}_2}(\kappa)=1]\right|.$$

Lemma 1. *For any QPT $\mathcal{A}$,* $\mathsf{AdvHyb}^{\mathsf{TrapTP}'}_{\mathsf{TrapTP}}(\mathcal{A},\kappa) \leq \mathrm{negl}(\kappa)$.

Hybrid 2: Removing Computation Log. In TrapTP and TrapTP', the adversary (homomorphically) keeps track of the keys to the QOTP and stores encryptions of all intermediate values in the computation log. Whenever VerDec needs to know the value of a key (for example to check a trap or to decrypt the final output state), the relevant entry in the computation log is decrypted.

In TrapTP', however, the plaintext initial values to the computation log are available to VerDec, as they are sent through the classical side channels. This means that whenever VerDec needs to know the value of a key, instead of decrypting an entry to the computation log, it can be computed by "shadowing" the computation log in the clear.

For example, suppose the log contains the encryptions $\widetilde{b_1}, \widetilde{b_2}$ of two initial bits, and specifies the homomorphic evaluation of XOR, resulting in $\widetilde{b}$ where $b = b_1 \oplus b_2$. If one knows the plaintext values b_1 and b_2, then one can compute $b_1 \oplus b_2$ directly, instead of decrypting the entry $\widetilde{b}$ from the computation log.

We now define a second hybrid, TrapTP'', which differs from TrapTP' exactly like this: VerDec still verifies the authenticated parts of the log, checks whether the computation log matches the structure of c, and checks whether it is syntactically correct. However, instead of decrypting values from the log (as it does in $\mathsf{TrapTP.VerDec}$, Algorithm 10, on lines 9 and 14), it computes those values from the plaintext initial values, by following the computation steps that are claimed in the log. By correctness of classical FHE, we then have the following.

Lemma 2. *For any QPT $\mathcal{A}$,* $\mathsf{AdvHyb}^{\mathsf{TrapTP}''}_{\mathsf{TrapTP}'}(\mathcal{A},\kappa) \leq \mathrm{negl}(\kappa)$.

Proof. Let s be the (plaintext) classical information that forms the input to the classical computations performed by the adversary: initial QOTP keys, secret keys and permutations, measurement results, etc. Let f be the function that the adversary computes on it in order to arrive at the final keys and logical measurement results. By correctness of HE, we have that

$$\Pr[\mathsf{HE.Dec}_{sk_t}(\mathsf{HE.Eval}^{f}_{evk_0,\ldots,evk_t}(\mathsf{HE.Enc}_{pk_0}(s))) \neq f(s)] \leq \mathrm{negl}(\kappa).$$

In the above expression, we slightly abuse notation and write $\mathsf{HE.Eval}_{evk_0,\ldots,evk_t}$ to include the t recryption steps that are performed during $\mathsf{TrapTP.Eval}$. As long as the number of T gates, and thus the number of recryptions, is polynomial in κ, the expression holds.

Thus, the probability that $\mathsf{TrapTP}'.\mathsf{VerDec}$ and $\mathsf{TrapTP}''.\mathsf{VerDec}$ use different classical values (decrypting from the log vs. computing from the initial values) is negligible. Since this is the only place where the two schemes differ, the output

of the two VerDec functions will be identical, except with negligible probability. Thus $\mathcal{A}$ will either win in both $\mathsf{Hyb}_{\mathcal{A},\mathsf{TrapTP}'}(\kappa)$ and $\mathsf{Hyb}_{\mathcal{A},\mathsf{TrapTP}''}(\kappa)$, or lose in both, again except with negligible probability. □

More Hybrids: Removing Gadgets. We continue by defining a sequence of hybrid schemes based on TrapTP''. In $4t$ steps, we will move all error-correction functionality from the gadgets to VerDec. This will imply that the adversary has no information about the classical secret keys (which are involved in constructing these gadgets). This will allow us to eventually reduce the security of TrapTP to that of TC.

We remove the gadgets back-to-front, starting with the final gadget. Every gadget is removed in four steps. For all $1 \leq \ell \leq t$, define the hybrids $\mathsf{TrapTP}_1^{(\ell)}$, $\mathsf{TrapTP}_2^{(\ell)}$, $\mathsf{TrapTP}_3^{(\ell)}$, and $\mathsf{TrapTP}_4^{(\ell)}$ (with $\mathsf{TrapTP}_4^{(t+1)} := \mathsf{TrapTP}''$) as follows:

1. $\mathsf{TrapTP}_1^{(\ell)}$ is the same as $\mathsf{TrapTP}_4^{(\ell+1)}$ (or, in the case that $\ell = t$, the same as TrapTP''), except for the generation of the state Γ_ℓ (see Algorithm 1, line 15). In $\mathsf{TrapTP}_1^{(\ell)}$, all classical information encrypted under pk_ℓ is replaced with encryptions of zeros. In particular, for $i \geq \ell$, line 15 is adapted to

$$\begin{aligned}\Gamma_i \leftarrow &\mathsf{MAC.Sign}(\mathsf{HE.Enc}_{pk_i}(00\cdots0)) \\ &\otimes \mathsf{TrapTP}''.\mathsf{Enc}'(sk', \gamma_i^{\mathsf{mid}}) \otimes \mathsf{TrapTP.Enc}(sk, \gamma_i^{\mathsf{in}} \otimes \gamma_i^{\mathsf{out}})\end{aligned}$$

where $\mathsf{TrapTP}''.\mathsf{Enc}'$ also appends a signed encryption of zeros, effectively replacing line 1 in Algorithm 3 with

$$\widetilde{\sigma} \leftarrow \sum_{x,z\in\{0,1\}^{3m}} \Big(\mathsf{TC.Enc}((\pi,x,z),\sigma) \otimes \mathsf{MAC.Sign}_k(\mathsf{HE.Enc}_{pk}(00\cdots0))\Big)$$

It is important to note that in both KeyGen and Enc', the information that is sent to VerDec through the classical side channel is *not* replaced with zeros. Hence, the structural and encryption information about Γ_ℓ is kept from the adversary, and instead is directly sent (only) to the verification procedure. Whenever VerDec needs this information, it is taken directly from this trusted source, and the all-zero string sent by the adversary will be ignored.

2. $\mathsf{TrapTP}_2^{(\ell)}$ is the same as $\mathsf{TrapTP}_1^{(\ell)}$, except that for the ℓth gadget, the procedure TrapTP.PostGadgetGen is called instead of TrapTP.GadgetGen:

Algorithm 11. $\mathsf{TrapTP.PostGadgetGen}(sk_i)$

1: $g_i \leftarrow 0^{|g(sk_i)|}$
2: $(\gamma^{\mathsf{in}}, \gamma^{\mathsf{mid}}, \gamma^{\mathsf{out}}) \leftarrow$ halves of EPR pairs (send other halves to VerDec)
3: **return** $(g_i, \gamma_i^{\mathsf{in}}, \gamma_i^{\mathsf{mid}}, \gamma_i^{\mathsf{out}})$

This algorithm produces a 'gadget' in which all qubits are replaced with halves of EPR pairs. These still get encrypted in line 15 of Algorithm 1. All other halves of these EPR pairs are sent to VerDec through the provided quantum channel. $\mathsf{TrapTP}_2^{(\ell)}.\mathsf{VerDec}$ has access to the structural information g_ℓ (as

this is sent via the classical side information channel from KeyGen to VerDec) and performs the necessary Bell measurements to recreate $\gamma_\ell^{\mathsf{in}}$, $\gamma_\ell^{\mathsf{mid}}$ and $\gamma_\ell^{\mathsf{out}}$ after the adversary has interacted with the EPR pair halves. Effectively, this postpones the generation of the gadget structure to decryption time. Of course, the measurement outcomes are taken into account by VerDec when calculating updates to the quantum one-time pad. As can be seen from the description of $\mathsf{TrapTP}_4^{(\ell)}$, all corrections that follow the ℓth one are unaffected by the fact that the server cannot hold the correct information about these postponed measurements, not even in encrypted form.

3. $\mathsf{TrapTP}_3^{(\ell)}$ is the same as $\mathsf{TrapTP}_2^{(\ell)}$, except that gadget generation for the ℓth gadget is handled by TrapTP.FakeGadgetGen instead of TrapTP.PostGadgetGen.

Algorithm 12. TrapTP.FakeGadgetGen(sk_i)

1: $g_i \leftarrow 0^{|g(sk_i)|}$
2: $(\gamma^{\mathsf{in}}, \gamma^{\mathsf{mid}}, \gamma^{\mathsf{out}}) \leftarrow$ halves of EPR pairs (send other halves to VerDec)
3: Send γ^{mid} to VerDec as well
4: **return** $(g_i, \gamma_i^{\mathsf{in}}, |00\cdots 0\rangle, \gamma_i^{\mathsf{out}})$

This algorithm prepares, instead of halves of EPR pairs, $|0\rangle$-states of the appropriate dimension for $\gamma_\ell^{\mathsf{mid}}$. (Note that this dimension does not depend on $sk_{\ell-1}$). For $\gamma_\ell^{\mathsf{in}}$ and $\gamma_\ell^{\mathsf{out}}$, halves of EPR pairs are still generated, as in $\mathsf{TrapTP}_2^{(\ell)}$. Via the side channel, the full EPR pairs for $\gamma_\ell^{\mathsf{mid}}$ are sent to VerDec. As in the previous hybrids, the returned gadget is encrypted in TrapTP.KeyGen.

$\mathsf{TrapTP}_3^{(\ell)}$.VerDec verifies that the adversary performed the correct Bell measurements on the fake ℓth gadget by calling TC.VerDec. If this procedure accepts, $\mathsf{TrapTP}_3^{(\ell)}$.VerDec performs the verified Bell measurements on the halves of the EPR pairs received from $\mathsf{TrapTP}_3^{(\ell)}$.KeyGen (and subsequently performs the Bell measurements that depend on g_ℓ on the other halves, as in $\mathsf{TrapTP}_2^{(\ell)}$). Effectively, $\mathsf{TrapTP}_3^{(\ell)}$.VerDec thereby performs a protocol for HE.Dec, removing the phase error in the process.

4. $\mathsf{TrapTP}_4^{(\ell)}$ is the same as $\mathsf{TrapTP}_3^{(\ell)}$, except that VerDec (instead of performing the Bell measurements of the gadget protocol) uses its knowledge of the initial QOTP keys and all intermediate measurement outcomes to compute whether or not a phase correction is necessary after the ℓth T gate. $\mathsf{TrapTP}_4^{(\ell)}$.VerDec then performs this phase correction on the EPR half entangled with $\gamma_\ell^{\mathsf{in}}$, followed by a Bell measurement with the EPR half entangled with $\gamma_\ell^{\mathsf{out}}$.

The first $\ell - 1$ gadgets in $\mathsf{TrapTP}_1^{(\ell)}$ through $\mathsf{TrapTP}_4^{(\ell)}$ are always functional gadgets, as in TrapTP. The last $t - \ell$ gadgets are all completely replaced by dummy states, and their functionality is completely outsourced to VerDec. In four steps described above, the functionality of the ℓth gadget is also transferred to VerDec. It is important to replace only one gadget at a time, because replacing a real gadget with a fake one breaks the functionality of the gadgets that

occur later in the evaluation: the encrypted classical information held by the server does not correspond to the question of whether or not a phase correction is needed. By completely outsourcing the phase correction to VerDec, as is done for all gadgets after the ℓth one in all $\mathsf{TrapTP}_i^{(\ell)}$ schemes, we ensure that this incorrect classical information does not influence the outcome of the computation. Hence, correctness is maintained throughout the hybrid transformations. We now show that these transformations of the scheme do not significantly affect the adversary's winning probability in the hybrid indistinguishability game.

Lemma 3. *For any QPT $\mathcal{A}$, there exists a negligible function* negl *such that for all $1 \leq \ell \leq t$, $\mathsf{AdvHyb}^{\mathsf{TrapTP}_4^{(\ell+1)}}_{\mathsf{TrapTP}_1^{(\ell)}}(\mathcal{A}, \kappa) \leq \mathrm{negl}(\kappa)$.*

Proof (sketch). In $\mathsf{TrapTP}_4^{(\ell+1)}$, no information about $sk_{(\ell)}$ is sent to the adversary. In the original TrapTP scheme, the structure of the quantum state $\Gamma_{\ell+1}$ depended on it, but this structure has been replaced with dummy states in several steps in $\mathsf{TrapTP}_2^{\ell+1}$ through $\mathsf{TrapTP}_4^{\ell+1}$.

This is fortunate, since if absolutely no secret-key information is present, we are able to bound the difference in winning probability between $\mathsf{Hyb}_{\mathcal{A},\mathsf{TrapTP}_4^{(\ell+1)}}(\kappa)$ and $\mathsf{Hyb}_{\mathcal{A},\mathsf{TrapTP}_1^{\ell}}(\kappa)$ by reducing it to the IND-CPA security against quantum adversaries [6] of the classical homomorphic encryption scheme HE.

The proof is closely analogous to the proof of Lemma 1 in [12], and on a high level it works as follows. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ be a QPT adversary for the game $\mathsf{Hyb}_{\mathcal{A},\mathsf{TrapTP}_1^{(\ell)}}(\kappa)$ or $\mathsf{Hyb}_{\mathcal{A},\mathsf{TrapTP}_4^{(\ell+1)}}(\kappa)$ (we do not need to specify for which one, since they both require the same input/output interface). A new quantum adversary $\mathcal{A}'$ for the classical IND-CPA indistinguishability game is defined by having the adversary taking the role of challenger in either the game $\mathsf{Hyb}_{\mathcal{A},\mathsf{TrapTP}_1^{(\ell)}}(\kappa)$ or the game $\mathsf{Hyb}_{\mathcal{A},\mathsf{TrapTP}_4^{(\ell+1)}}(\kappa)$. Which game is simulated depends on the coin flip of the challenger for the IND-CPA indistinguishability game, and is unknown to $\mathcal{A}'$. This situation is achieved by having $\mathcal{A}'$ send any classical plaintext that should be encrypted under pk_ℓ to the challenger, so that either that plaintext is encrypted or a string of zeros is.

Based on the guess of the simulated $\mathcal{A}$, which $\mathcal{A}'$ can verify to be correct or incorrect in his role of challenger, $\mathcal{A}'$ will guess which of the two games was just simulated. By IND-CPA security of the classical scheme against quantum adversaries, $\mathcal{A}'$ cannot succeed in this guessing game with nonnegligible advantage over random guessing. This means that the winning probability of $\mathcal{A}$ in both games cannot differ by a lot. For details, we refer the reader the proof of Lemma 5, in which a very similar approach is taken.

Technically, the success probability of $\mathcal{A}'$, and thus the function negl, may depend on ℓ. A standard randomizing argument, as found in e.g. the discussion of hybrid arguments in [16], allows us to get rid of this dependence by defining another adversary $\mathcal{A}''$ that selects a random value of j, and then bounding the advantage of $\mathcal{A}''$ by a negligible function that is independent of j. □

Lemma 4. *For $1 \leq \ell \leq t$ and any QPT $\mathcal{A}$,* $\mathsf{AdvHyb}^{\mathsf{TrapTP}_2^{(\ell)}}_{\mathsf{TrapTP}_1^{(\ell)}}(\mathcal{A}, \kappa) = 0$.

Proof. In $\mathsf{TrapTP}_1^{(\ell)}$, the ℓth error-correction gadget consists of a number of EPR pairs arranged in a certain order, as described by the garden-hose protocol for $\mathsf{HE.Dec}$. For example, this protocol may dictate that the ith and jth qubit of the gadget must form an EPR pair $|\Phi^+\rangle$ together. This can alternatively be achieved by creating two EPR pairs, placing half of each pair in the ith and jth position of the gadget state, and performing a Bell measurement on the other two halves. This creates a Bell pair $\mathsf{X}^a\mathsf{Z}^b|\Phi^+\rangle$ in positions i and j, where $a, b \in \{0, 1\}$ describe the outcome of the Bell measurement.

From the point of view of the adversary, it does not matter whether these Bell measurements are performed during KeyGen, or whether the halves of EPR pairs are sent to VerDec for measurement – because the key to the quantum one-time pad of the ℓth gadget is not sent to the adversary at all, the same state is created with a completely random Pauli in either case. Of course, the teleportation correction Paulis of the form $\mathsf{X}^a\mathsf{Z}^b$ need to be taken into account when updating the keys to the quantum one-time pad on the data qubits after the gadget is used. VerDec has all the necessary information to do this, because it observes the measurement outcomes, and computes the key updates itself (instead of decrypting the final keys from the computation log).

Thus, with the extra key update steps in $\mathsf{TrapTP}_2^{(\ell)}.\mathsf{VerDec}$, the inputs to the adversary are exactly the same in the games of $\mathsf{TrapTP}_1^{(\ell)}$ and $\mathsf{TrapTP}_2^{(\ell)}$. □

Lemma 5. *For any QPT $\mathcal{A}$, there exists a negligible function* negl *such that for all $1 \leq \ell \leq t$,* $\mathsf{AdvHyb}^{\mathsf{TrapTP}_3^{(\ell)}}_{\mathsf{TrapTP}_2^{(\ell)}}(\mathcal{A}, \kappa) \leq \mathrm{negl}(\kappa)$.

Proof. We show this by reducing the difference in winning probabilities in the statement of the lemma to the IND-VER security of the somewhat homomorphic scheme TC. Intuitively, because TC is IND-VER, if $\mathsf{TrapTP}_2^{(\ell)}$ accepts the adversary's claimed circuit of Bell measurements on the EPR pair halves, the effective map on those EPR pairs is the claimed circuit. Therefore, we might just as well ask VerDec to apply this map, as we do in $\mathsf{TrapTP}_3^{(\ell)}$, to get the same output state. If $\mathsf{TrapTP}_2^{(\ell)}$ rejects the adversary's claimed circuit on those EPR pair halves, then $\mathsf{TrapTP}_3^{(\ell)}$ should reject too. This is why we let the adversary act on an encrypted dummy state of $|0\rangle$s.

Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ be a set of QPT algorithms on the appropriate registers, so that we can consider it as an adversary for the hybrid indistinguishability game for either $\mathsf{TrapTP}_2^{(\ell)}$ or $\mathsf{TrapTP}_3^{(\ell)}$ (see Definition 7). Note the input/output wires to the adversary in both these games are identical, so we can evaluate $\Pr[\mathsf{Hyb}_{\mathcal{A},\mathsf{TrapTP}_2^{(\ell)}}(\kappa) = 1]$ and $\Pr[\mathsf{Hyb}_{\mathcal{A},\mathsf{TrapTP}_3^{(\ell)}}(\kappa) = 1]$ for the same $\mathcal{A}$.

Now define an adversary $\mathcal{A}' = (\mathcal{A}'_1, \mathcal{A}'_2, \mathcal{A}'_3)$ for the IND-VER game against TC, $\mathsf{VerGame}_{\mathcal{A}',\mathsf{TC}}(\kappa)$, as follows:

1. $\mathcal{A}'_1$: Run $\mathsf{TrapTP}_2^{(\ell)}.\mathsf{KeyGen}$ until the start of line 15 in the ℓth iteration of that loop. Up to this point, $\mathsf{TrapTP}_2^{(\ell)}.\mathsf{KeyGen}$ is identical to $\mathsf{TrapTP}_3^{(\ell)}.\mathsf{KeyGen}$.

It has generated real gadgets Γ_1 through $\Gamma_{\ell-1}$, and halves of EPR pairs for $\gamma_\ell^{\mathsf{in}}$, $\gamma_\ell^{\mathsf{mid}}$ and $\gamma_\ell^{\mathsf{out}}$. Note furthermore that the permutation π_ℓ is used nowhere. Now send $\gamma_\ell^{\mathsf{mid}}$ to the challenger via the register X, and everything else (including sk) to $\mathcal{A}_2'$ via the side register R.

2. $\mathcal{A}_2'$: Continue $\mathsf{TrapTP}_2^{(\ell)}.\mathsf{KeyGen}$ using the response from the challenger instead of $\mathsf{TrapTP}.\mathsf{Enc}'(sk', \gamma_\ell^{\mathsf{mid}})$ on line 15 in the ℓth iteration. Call the result ρ_{evk}. Again, this part of the key generation procedure is identical for $\mathsf{TrapTP}_2^{(\ell)}$ and $\mathsf{TrapTP}_3^{(\ell)}$. Start playing the hybrid indistinguishability game with $\mathcal{A}$:

- Flip a bit $r \in \{0,1\}$.
- Send ρ_{evk} to $\mathcal{A}_1$. If $r = 0$, encrypt the response of $\mathcal{A}_1$ using the secret key sk generated by $\mathcal{A}_1'$. Note that for this, the permutation π_ℓ is also not needed. If $r = 1$, encrypt a $|0\rangle$ state of appropriate dimension instead.
- Send the resulting encryption, along with the side info from $\mathcal{A}_1$, to $\mathcal{A}_2$.
- On the output of $\mathcal{A}_2$, start running $\mathsf{TrapTP}_2^{(\ell)}.\mathsf{VerDec}$ until the actions on the ℓth gadget need to be verified. Since the permutation on the state $\gamma_\ell^{\mathsf{mid}}$ is unknown to $\mathcal{A}_2'$ (it was sent to the challenger for encryption), it cannot verify this part of the computation.
- Instead, send the relevant part of the computation log to the challenger for verification, along with the relevant part of the claimed circuit (the Bell measurements on the gadget state), and the relevant qubits, all received from $\mathcal{A}_2$, to the challenger for verification and decryption.
- In the meantime, send the rest of the working memory to $\mathcal{A}_3'$ via register R'.

3. $\mathcal{A}_3'$: Continue the simulation of the hybrid game with $\mathcal{A}$:

- If the challenger rejects, reject and replace the entire quantum state by the fixed dummy state Ω.
- If the challenger accepts, then we know that the challenger applies the claimed subcircuit to the quantum state it did not encrypt (either $|0\rangle$ or $\gamma_\ell^{\mathsf{mid}}$), depending on the bit the challenger flipped), and possibly swaps this state back in (again depending on which bit it flipped). Continue the $\mathsf{TrapTP}_2^{(\ell)}.\mathsf{VerDec}$ computation for the rest of the computation log.
- Send the result (the output quantum state, the claimed circuit, and the accept/reject flag) to $\mathcal{A}_3$, and call its output bit r'.

Output 0 if $r = r'$, and 1 otherwise. (i.e., output $NEQ(r, r')$)

Recall from Definition 7 that the challenger flips a coin (let us call the outcome $s \in \{0,1\}$) to decide whether to encrypt the quantum state provided by $\mathcal{A}'$, or to swap in an all-zero dummy state before encrypting. Keeping this in mind while inspecting the definition of $\mathcal{A}'$, one can see that whenever $s = 0$, $\mathcal{A}'$ takes the role of challenger in the game $\mathsf{Hyb}_{\mathcal{A},\mathsf{TrapTP}_2^{(\ell)}}(\kappa)$ with $\mathcal{A}$, and whenever $s = 1$, they play $\mathsf{Hyb}_{\mathcal{A},\mathsf{TrapTP}_3^{(\ell)}}(\kappa)$. Now let us consider when the newly defined adversary $\mathcal{A}'$ wins the VER indistinguishability game for TC. If $s = 0$, $\mathcal{A}'$ needs to output a bit $s' = 0$ to win. This happens, by definition of $\mathcal{A}'$, if and only if $\mathcal{A}$ wins the game $\mathsf{Hyb}_{\mathcal{A},\mathsf{TrapTP}_2^{(\ell)}}(\kappa)$ (i.e. $r = r'$). On the other hand, if $s = 1$, $\mathcal{A}'$

needs to output a bit $s' = 1$ to win. This happens, by definition of $\mathcal{A}'$, if and only if $\mathcal{A}$ loses the game $\mathsf{Hyb}_{\mathcal{A},\mathsf{TrapTP}_3^{(\ell)}}(\kappa)$ (i.e. $r \neq r'$). Thus the winning probability of $\mathcal{A}'$ is:

$$\begin{aligned}
&\Pr[\mathsf{VerGame}_{\mathcal{A}',\mathsf{TC}}(\kappa) = 1] \\
&= \Pr[s = 0] \cdot \Pr[\mathsf{Hyb}_{\mathcal{A},\mathsf{TrapTP}_2^{(\ell)}}(\kappa) = 1] + \Pr[s = 1] \cdot \Pr[\mathsf{Hyb}_{\mathcal{A},\mathsf{TrapTP}_3^{(\ell)}}(\kappa) = 0] \\
&= \frac{1}{2}\Pr[\mathsf{Hyb}_{\mathcal{A},\mathsf{TrapTP}_2^{(\ell)}}(\kappa) = 1] + \frac{1}{2}\Big(1 - \Pr[\mathsf{Hyb}_{\mathcal{A},\mathsf{TrapTP}_3^{(\ell)}}(\kappa) = 1]\Big) \\
&= \frac{1}{2} + \frac{1}{2}\Big(\Pr[\mathsf{Hyb}_{\mathcal{A},\mathsf{TrapTP}_2^{(\ell)}}(\kappa) = 1] - \Pr[\mathsf{Hyb}_{\mathcal{A},\mathsf{TrapTP}_3^{(\ell)}}(\kappa) = 1]\Big)
\end{aligned}$$

From the IND-VER property of TC (see Theorem 4) we know that the above is at most $\frac{1}{2} + \mathrm{negl}(\kappa)$. From this (and a randomizing argument similar to Lemma 3), the statement of the lemma follows directly. □

Lemma 6. *For any QPT $\mathcal{A}$, there exists a negligible function* negl *such that for all $1 \leq \ell \leq t$, $\mathsf{AdvHyb}^{\mathsf{TrapTP}_4^{(\ell)}}_{\mathsf{TrapTP}_3^{(\ell)}}(\mathcal{A}, \kappa) \leq \mathrm{negl}(\kappa)$.*

Proof. Let $f(s)$ be the bit that, after the ℓth T gate, determines whether or not a phase correction is necessary. Here, s is all the relevant starting information (such as quantum one-time pad keys, gadget structure, permutations, and applied circuit), and f is some function that determines the X key on the relevant qubit right before application of the T gate.

In $\mathsf{TrapTP}_3^{(\ell)}$, a phase correction after the ℓth T gate is applied conditioned on the outcome of

$$\mathsf{HE.Dec}_{sk_{\ell-1}}(\mathsf{HE.Eval}^{f}_{evk_0,\ldots,evk_{\ell-1}}(\mathsf{HE.Enc}_{pk_0}(s))),$$

because the garden-hose computation in the gadget computes the classical decryption. In the above expression, we again slightly abuse notation, as in the proof of Lemma 2, and include recryption steps in $\mathsf{HE.Eval}_{evk_0,\ldots,\mathsf{evk}_{\ell-1}}$. As long as t is polynomial in κ, we have, by correctness of HE,

$$\Pr[\mathsf{HE.Dec}_{sk_{\ell-1}}(\mathsf{HE.Eval}^{f}_{evk_0,\ldots,evk_{\ell-1}}(\mathsf{HE.Enc}_{pk_0}(s))) \neq f(s)] \leq \mathrm{negl}(\kappa).$$

In $\mathsf{TrapTP}_4^{(\ell)}$, the only difference from $\mathsf{TrapTP}_3^{(\ell)}$ is that, instead of performing the garden-hose computation on the result of the classical homomorphic evaluation procedure, the phase correction is applied directly by VerDec, conditioned on $f(s)$. The probability that in $\mathsf{TrapTP}_4^{(\ell)}$, a phase is applied (or not) when in $\mathsf{TrapTP}_3^{(\ell)}$ it is not (or is), is negligible. The claim follows directly. □

Final Hybrid: Removing All Classical FHE. In $\mathsf{TrapTP}_4^{(1)}$, all of the error-correction gadgets have been removed from the evaluation key, and the error-correction functionality has been redirected to VerDec completely. Effectively, $\mathsf{TrapTP}_4^{(1)}.\mathsf{KeyGen}$ samples a permutation π, generates a lot of magic states (for

P, H and T) and encrypts them using $\mathsf{TC.Enc}_\pi$, after which the keys to the quantum one-time pad used in that encryption are homomorphically encrypted under pk_0. The adversary is allowed to act on those encryptions, but while its homomorphic computations are syntactically checked in the log, VerDec does not decrypt and use the resulting values. This allows us to link $\mathsf{TrapTP}_4^{(1)}$ to a final hybrid, TrapTP^f, where all classical information is replaced with zeros before encrypting.

The proof of the following lemma is analogous to that of Lemma 3, and reduces to the IND-CPA security of the classical scheme HE:

Lemma 7. *For any QPT $\mathcal{A}$,* $\mathsf{AdvHyb}^{\mathsf{TrapTP}^f}_{\mathsf{TrapTP}_4^{(1)}}(\mathcal{A},\kappa) \leq \mathrm{negl}(\kappa)$.

Proof of Main Theorem. Considering TrapTP^f in more detail, we can see that it is actually very similar to TC. This allows us to prove the following lemma, which is the last ingredient for the proof of verifiability of TrapTP.

Lemma 8. *For any QPT $\mathcal{A}$,* $\Pr[\mathsf{Hyb}_{\mathcal{A},\mathsf{TrapTP}^f}(\kappa)=1] \leq \frac{1}{2}+\mathrm{negl}(\kappa)$.

Proof. To see the similarity with TC, consider the four algorithms of TrapTP^f.

In $\mathsf{TrapTP}^f.\mathsf{KeyGen}$, a permutation π is sampled, and magic states for P, H and T are generated, along with some EPR pair halves (to replace in_i and out_i). For all generated quantum states, random keys for QOTPs are sampled, and the states are encrypted using $\mathsf{TC.Enc}$ with these keys as secret keys. No classical FHE is present anymore. Thus, $\mathsf{TrapTP}^f.\mathsf{KeyGen}$ can be viewed as $\mathsf{TC.KeyGen}$, followed by $\mathsf{TC.Enc}$ on the magic states and EPR pair halves.

$\mathsf{TrapTP}^f.\mathsf{Enc}$ is identical to $\mathsf{TC.Enc}$, only the keys to the quantum one-time pad are sampled on the fly and sent to $\mathsf{TrapTP}^f.\mathsf{VerDec}$ via a classical side-channel, whereas $\mathsf{TC.VerDec}$ receives them as part of the secret key. Since the keys are used exactly once and not used anywhere else besides in Enc and VerDec, this difference does not affect the outcome of the game.

$\mathsf{TrapTP}^f.\mathsf{Eval}$ only requires CNOT, classically controlled Paulis, and computational/Hadamar basis measurements. For the execution of any other gate, it suffices to apply a circuit of those gates to the encrypted data, encrypted magic states and/or encrypted EPR halves.

$\mathsf{TrapTP}^f.\mathsf{VerDec}$ does two things: (i) it syntactically checks the provided computation log, and (ii) it runs $\mathsf{TC.VerDec}$ to verify that the evaluation procedure correctly applied the circuit of CNOTs and measurements.

An execution of $\mathsf{Hyb}_{\mathcal{A},\mathsf{TrapTP}^f}(\kappa)$ for any $\mathcal{A}$ corresponds to the two-round VER indistinguishability game for TC as follows. Let $\mathcal{A}=(\mathcal{A}_1,\mathcal{A}_2,\mathcal{A}_3)$ be a polynomial-time adversary for the game $\mathsf{Hyb}_{\mathcal{A},\mathsf{TrapTP}^f}(\kappa)$. Define an additional QPT $\mathcal{A}_0$ that produces magic states and EPR pair halves to the register X_1. The other halves of the EPR pairs are sent through R, and untouches by $\mathcal{A}_1$ and $\mathcal{A}_2$. The above analysis shows that the adversary $\mathcal{A}'=(\mathcal{A}_0,\mathcal{A}_1,\mathcal{A}_2,\mathcal{A}_3)$ can be viewed as an adversary for the VER-2 indistinguishability game $\mathsf{VerGame}^2_{\mathcal{A}',\mathsf{TC}}(\kappa)$ and wins whenever $\mathsf{Hyb}_{\mathcal{A},\mathsf{TrapTP}^f}(\kappa)=1$. The other direction does not hold: $\mathcal{A}$

loses the hybrid indistinguishability game if $\mathsf{TrapTP}^f.\mathsf{VerDec}$ rejects check (i), but accepts check (ii) (see above). In this case, $\mathcal{A}'$ would still win the VER-2 indistinguishability game. Hence,

$$\Pr[\mathsf{Hyb}_{\mathcal{A},\mathsf{TrapTP}^f}(\kappa)=1] \leq \Pr[\mathsf{VerGame}^2_{\mathcal{A}',\mathsf{TC}}(\kappa)=1].$$

Theorem 4 yields $\Pr[\mathsf{VerGame}^2_{\mathcal{A}',\mathsf{TC}}(\kappa) = 1] \leq \frac{1}{2} + \mathrm{negl}(\kappa)$, and the result follows. □

Theorem 5. *The vQFHE scheme* TrapTP *satisfies* κ*-SEM-VER.*

Proof. From Lemmas 1, 2, 3, 4, 5, 6, and 7, we may conclude that if t (the number of T gates in the circuit) is polynomial in κ (the security parameter), then for any polynomial-time adversary $\mathcal{A}$,

$$\Pr[\mathsf{VerGame}_{\mathcal{A},\mathsf{TrapTP}}(\kappa)=1] \quad - \quad \Pr[\mathsf{Hyb}_{\mathcal{A},\mathsf{TrapTP}^f}(\kappa)=1] \quad \leq \quad \mathrm{negl}(\kappa),$$

since the sum poly-many negligible terms is negligible (it is important to note that there is only a constant number of *different* negligible terms involved). By Lemma 8, which reduces verifiability of TrapTP^f to verifiability of TC, $\Pr[\mathsf{Hyb}_{\mathcal{A},\mathsf{TrapTP}^f}(\kappa)=1] \leq 1/2+\mathrm{negl}(\kappa)$. It follows that $\Pr[\mathsf{VerGame}_{\mathcal{A},\mathsf{TrapTP}}(\kappa) = 1] \leq 1/2 + \mathrm{negl}(\kappa)$, i.e., that TrapTP is κ-IND-VER. By Theorem 3, TrapTP is also κ-SEM-VER. □

6 Application to Quantum One-Time Programs

One-Time Programs. We now briefly sketch an application of the vQFHE scheme to one-time programs. A classical one-time program (or cOTP) is an idealized object which can be used to execute a function once, but then self-destructs. In the case of a quantum OTP (or qOTP), the program executes a quantum channel Φ. In the usual formalization, Φ has two inputs and is public. One party (the sender) creates the qOTP by fixing one input, and the qOTP is executed by a receiver who selects the other input. To recover the intuitive notion of OTP, choose Φ to be a universal circuit. We will work in the universally-composable (UC) framework, following the approach of [9]. We thus first define the ideal functionality of a qOTP.

Definition 8 (Functionality 3 in [9]). *The ideal functionality* $\mathcal{F}_\Phi^{\mathsf{OTP}}$ *for a channel* $\Phi_{XY\to Z}$ *is the following:*

1. **Create:** *given register* X *from sender, store* X *and send* create *to receiver.*
2. **Execute:** *given register* Y *from receiver, send* Φ *applied to* XY *to receiver. Delete any trace of this instance.*

A qOTP is then a real functionality which "UC-emulates" the ideal functionality [22]. As in [9], we only allow corrupting receivers; unlike [9], we consider computational (rather than statistical) UC security. The achieved result is therefore slightly weaker. The construction within our vQFHE framework is however much simpler, and shows the relative ease with which applications of vQFHE can be constructed.

The Construction. Choose a vQFHE scheme $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{VerDec})$ satisfying SEM-VER. For simplicity, we first describe the classical input/output case, i.e., the circuit begins and ends with full measurement of all qubits. Let C be such a circuit, for the map $\Phi_{XY\to Z}$. On **Create**, the sender generates keys $(k, \rho_{\mathsf{evk}}) \leftarrow \mathsf{KeyGen}$ and encrypts their input register X using k. The sender also generates a classical OTP for the public, classical function VerDec, choosing the circuit and key inputs to be C and k; the computation log is left open for the receiver to select. The qOTP is then the triple

$$\Xi_C^X := (\rho_{\mathsf{evk}}, \mathsf{Enc}_k(\rho_X), \mathsf{OTP}_{\mathsf{VerDec}}(C, k)) .$$

On **Execute**, the receiver computes as follows. The receiver's (classical) input Y together with the (public) circuit C defines a homomorphic computation on the ciphertext $\mathsf{Enc}_k(\rho_X)$, which the receiver can perform using Eval and ρ_{evk}. Since C has only classical outputs, the receiver measures the final state completely. At the end of that computation, the receiver holds the (completely classical) output of the computation log from Eval. The receiver plugs the log into $\mathsf{OTP}_{\mathsf{VerDec}}(C, k)$, which produces the decrypted output.

We handle the case of arbitrary circuits C (with quantum input and output) as follows. Following the ideas of [9], we augment the above quantum OTP with two auxiliary quantum states: an "encrypt-through-teleport" gadget σ_{in} and a "decrypt-through-teleport" gadget σ_{out}. These are maximally entangled states with the appropriate map (encrypt or decrypt) applied to one half. The receiver uses teleportation on $\sigma^{\mathsf{in}}_{Y_1W_1}$ to encrypt their input register Y before evaluating, and places the teleportation measurements into the computation log. After evalution, the receiver uses $\sigma^{\mathsf{out}}_{W_2Y_2}$ to teleport the plaintext out, combining the teleportation measurements with the output of $\mathsf{OTP}_{\mathsf{VerDec}}(C, k)$ to compute the final QOTP decryption keys.

Security Proof Sketch. Starting with a QPT adversary $\mathcal{A}$ which attacks the real functionality, we construct a QPT simulator $\mathcal{S}$ which attacks the ideal functionality (with similar success probability). We split $\mathcal{A}$ into $\mathcal{A}_1$ (receive input, output the OTP query and side information) and $\mathcal{A}_2$ (receive result of OTP query and side information, produce final output). The simulator $\mathcal{S}$ will generate its own keys, provide fake gadgets that will trick $\mathcal{A}$ into teleporting its input to $\mathcal{S}$, who will then use that input on the ideal functionality. Details follow.

The simulator first generates $(k, \rho_{\mathsf{evk}}) \leftarrow \mathsf{KeyGen}$ and encrypts the input X via Enc_k. Instead of the encrypt gadget $\sigma^{\mathsf{in}}_{Y_1W_1}$, $\mathcal{S}$ provides half of a maximally entangled state in register Y and likewise in register W. The other halves Y_1' and W_1' of these entangled states are kept by $\mathcal{S}$. The same is done in place of the decrypt gadget $\sigma^{\mathsf{out}}_{W_2Y_2}$, with $\mathcal{S}$ keeping Y_2' and W_2'. Then $\mathcal{S}$ runs $\mathcal{A}_1$ with input $\rho_{\mathsf{evk}}, \mathsf{Enc}_k(\rho_X)$ and registers Y and W. It then executes VerDec_k on the output (i.e., the query) of $\mathcal{A}_1$ to see if $\mathcal{A}_1$ correctly followed the Eval protocol. If it did not, then $\mathcal{S}$ aborts; otherwise, $\mathcal{S}$ plugs register Y_1' into the ideal functionality, and then teleports the output into register W_2'. Before responding to $\mathcal{A}_2$, it corrects the one-time pad keys appropriately using its teleportation measurements.

7 Conclusion

In this work, we devised a new quantum-cryptographic primitive: quantum fully-homomorphic encryption with verification (vQFHE). Using the trap code for quantum authentication [9] and the garden-hose gadgets of [12], we constructed a vQFHE scheme TrapTP which satisfies (i) correctness, (ii) compactness, (iii) security of verification, (iv) IND-CPA secrecy, and (v) authentication. We also outlined a first application of vQFHE, to quantum one-time programs.

We leave open several interesting directions for future research. Foremost is finding more applications of vQFHE. Another interesting question is whether vQFHE schemes exist where verification can be done publicly (i.e., without the decryption key), as is possible classically. Finally, it is unknown whether vQFHE (or even QFHE) schemes exist with evaluation key that does not scale with the size of the circuit at all.

Acknowledgements. This work was completed while GA was a member of the QMATH center at the Department of Mathematical Sciences at the University of Copenhagen. GA and FS acknowledge financial support from the European Research Council (ERC Grant Agreement no 337603), the Danish Council for Independent Research (Sapere Aude), Qubiz - Quantum Innovation Center, and VILLUM FONDEN via the QMATH Centre of Excellence (Grant No. 10059). CS is supported by an NWO VIDI grant.

References

1. Aharonov, D., Ben-Or, M., Eban, E.: Interactive proofs for quantum computations. arXiv preprint arXiv:0810.5375 (2008)
2. Alagic, G., Broadbent, A., Fefferman, B., Gagliardoni, T., Schaffner, C., St. Jules, M.: Computational security of quantum encryption. In: Nascimento, A.C.A., Barreto, P. (eds.) ICITS 2016. LNCS, vol. 10015, pp. 47–71. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-49175-2_3
3. Alagic, G., Dulek, Y., Schaffner, C., Speelman, F.: Quantum fully homomorphic encryption with verification. arXiv preprint arXiv:1708.09156 (2017)
4. Barak, B., Brakerski, Z.: Windows on theory: the swiss army knife of cryptography (2012). URL https://windowsontheory.org/2012/05/01/the-swiss-army-knife-of-cryptography/
5. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. In: 52nd Annual Symposium on Foundations of Computer Science (FOCS), pp. 97–106 (2011). https://doi.org/10.1109/FOCS.2011.12
6. Broadbent, A., Jeffery, S.: Quantum homomorphic encryption for circuits of low T-gate complexity. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9216, pp. 609–629. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48000-7_30
7. Broadbent, A., Wainewright, E.: Efficient simulation for quantum message authentication. arXiv preprint arXiv:1607.03075 (2016)
8. Broadbent, A., Fitzsimons, J., Kashefi, E.: Universal blind quantum computation. In: 50th Annual Symposium on Foundations of Computer Science (FOCS), pp. 517–526. IEEE (2009)

9. Broadbent, A., Gutoski, G., Stebila, D.: Quantum one-time programs. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8043, pp. 344–360. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40084-1_20
10. Broadbent, A., Ji, Z., Song, F., Watrous, J.: Zero-knowledge proof systems for QMA. In: 57th Annual Symposium on Foundations of Computer Science (FOCS), pp. 31–40, October 2016. https://doi.org/10.1109/FOCS.2016.13
11. Coladangelo, A., Grilo, A., Jeffery, S., Vidick, T.: Verifier-on-a-leash: new schemes for verifiable delegated quantum computation, with quasilinear resources. arXiv preprint arXiv:1708.02130 (2017)
12. Dulek, Y., Schaffner, C., Speelman, F.: Quantum homomorphic encryption for polynomial-sized circuits. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9816, pp. 3–32. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53015-3_1
13. Dupuis, F., Nielsen, J.B., Salvail, L.: Actively secure two-party evaluation of any quantum operation. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 794–811. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_46
14. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: 54th Annual Symposium on Foundations of Computer Science (FOCS), pp. 40–49, October (2013). https://doi.org/10.1109/FOCS.2013.13
15. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: 41st Annual ACM Symposium on Theory of Computing (STOC), pp. 169–178 (2009). https://doi.org/10.1145/1536414.1536440
16. Katz, J., Lindell, Y.: Introduction to Modern Cryptography. CRC Press, Boca Raton (2014)
17. Mahadev, U.: Classical homomorphic encryption for quantum circuits. arXiv preprint arXiv:1708.02130 (2017)
18. Newman, M., Shi, Y.: Limitations on transversal computation through quantum homomorphic encryption. arXiv e-prints, April 2017
19. Ouyang, Y., Tan, S.-H., Fitzsimons, J.: Quantum homomorphic encryption from quantum codes. arXiv preprint arXiv:1508.00938 (2015)
20. Shor, P.W., Preskill, J.: Simple proof of security of the BB84 quantum key distribution protocol. Phys. Rev. Lett. **85**, 441–444 (2000). https://doi.org/10.1103/PhysRevLett.85.441
21. Tan, S.-H., Kettlewell, J.A., Ouyang, Y., Chen, L., Fitzsimons, J.: A quantum approach to homomorphic encryption. Sci. Rep. **6**, 33467 (2016). https://doi.org/10.1038/srep33467
22. Unruh, D.: Universally composable quantum multi-party computation. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 486–505. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_25
23. Li, Y., Pérez-Delgado, C.A., Fitzsimons, J.F.: Limitations on information-theoretically-secure quantum homomorphic encryption. Phys. Rev. A **90**, 050303 (2014). https://doi.org/10.1103/PhysRevA.90.050303