Actively Secure Two-Party Evaluation of any Quantum Operation

Frédéric Dupuis[1⋆] Jesper Buus Nielsen[2⋆⋆] Louis Salvail[3⋆⋆⋆]

[1] Institute for Theoretical Physics, ETH Zurich, Switzerland
dupuis@phys.ethz.ch
[2] Department of Computer Science, Aarhus University, Denmark
jbn@cs.au.dk
[3] Université de Montréal (DIRO), QC, Canada
salvail@iro.umontreal.ca

Abstract. We provide the first two-party protocol allowing Alice and Bob to evaluate privately even against active adversaries any completely positive, trace-preserving map $\mathscr{F} \in \mathrm{L}(\mathcal{A}_{\mathrm{in}} \otimes \mathcal{B}_{\mathrm{in}}) \to \mathrm{L}(\mathcal{A}_{\mathrm{out}} \otimes \mathcal{B}_{\mathrm{out}})$, given as a quantum circuit, upon their joint quantum input state $\rho_{\mathrm{in}} \in \mathrm{D}(\mathcal{A}_{\mathrm{in}} \otimes \mathcal{B}_{\mathrm{in}})$. Our protocol leaks no more to any active adversary than an ideal functionality for $\mathscr{F}$ provided Alice and Bob have the cryptographic resources for active secure two-party classical computation. Our protocol is constructed from the protocol for the same task secure against specious adversaries presented in [4]. We show how to transform it so that it preserves privacy against active adversaries as well.

1 Introduction

We provide the first active-secure two-party protocol for computing on quantum data. We look at a model where Alice and Bob hold an input ρ_{in} on registers $\mathcal{A}_{\mathrm{in}}$ and $\mathcal{B}_{\mathrm{in}}$, where Alice holds register $\mathcal{A}_{\mathrm{in}}$ and Bob holds $\mathcal{B}_{\mathrm{in}}$. They agree on a completely positive, trace-preserving (CPTP) map $\mathscr{F}$ from registers $\mathcal{A}_{\mathrm{in}} \otimes \mathcal{B}_{\mathrm{in}}$ to registers $\mathcal{A}_{\mathrm{out}} \otimes \mathcal{B}_{\mathrm{out}}$, and they want to compute $\rho_{\mathrm{out}} = \mathscr{F}(\rho_{\mathrm{in}})$ such that, at the end of the protocol, Alice is in possession of $\mathcal{A}_{\mathrm{out}}$ and Bob is in possession of $\mathcal{B}_{\mathrm{out}}$. They want to do this in an actively secure manner. Our notion of active security is phrased via simulation, but intuitively it simply guarantees that any cheating Alice, even an infinitely powerful Alice, which might deviate from the protocol, can only affect the output of the protocol by replacing her own input and that she will at any point during the execution of the protocol only hold information which can be computed (efficiently) from either $\rho_{\mathrm{in}}^{\mathcal{A}}$ or $\rho_{\mathrm{out}}^{\mathcal{A}}$. The equivalent condition should hold for Bob.

⋆ Supported by Canada's NSERC Postdoctoral Fellowship Program and by the Swiss National Science Foundation via the National Center of Competence QSIT.

⋆⋆ Partially supported by an European Research Council, Starting Grant, grant agreement number 279447 and the Danish National Research Foundation and the National Science Foundation of China (under the grant 61061130540) for the Sino-Danish Center for the Theory of Interactive Computation.

⋆⋆⋆ Supported by Canada's NSERC discovery grant, FREQUENCY(NSERC), the QuantumWorks networks(NSERC), and INTRIQ.

A simple example of such an $\mathscr{F}$ is the quantum swap, where $\mathcal{A}_{\text{in}} = \mathcal{A}_{\text{out}} = \mathcal{A}$, $\mathcal{B}_{\text{in}} = \mathcal{B}_{\text{out}} = \mathcal{B}$, and $\rho^{\mathcal{A}}_{\text{out}} = \rho^{\mathcal{B}}_{\text{in}}$ and $\rho^{\mathcal{B}}_{\text{out}} = \rho^{\mathcal{A}}_{\text{in}}$. Securely implementing this unitary basically means to do an atomic swap of quantum states, which was shown impossible in [4] even against a restricted class of adversaries called *specious*. Extra assumptions are therefore needed to get unconditional security. Our way out is to look at a model where the two parties have access to an ideal functionality which allows them to securely do any *classical* computation on any *classical* data held jointly by the two parties. In this model we give an unconditionally secure protocol with active security. This is the first such protocol.

Formally, we use the notationally simpler model of [4], but it is easy to see that security in this model implies security in the stand-alone model of [11], as long as the simulator is poly-time. The stand-alone model of [11] allows, inside a secure quantum protocol, to replace a classical ideal functionality by a classical protocol which securely implements that ideal functionality against poly-time quantum adversaries. The result is a protocol secure against poly-time quantum adversaries. This, in particular, implies sequential security, i.e., if a protocol is secure, it remains secure when run in sequence with other secure protocols. Since the secure evaluation of any classical function with security against poly-time quantum adversaries can be done under the assumption that learning with errors is hard [13,11] and under the more general assumption that mixed commitment schemes exists [12], our poly-time simulator provides an active-secure two-party plain-model protocol for computing on quantum data with security against any poly-time quantum adversaries. This is the first such protocol.

1.1 Overview of our construction

We reuse many ideas from the protocol provided in [4], which gives a two-party protocol for computing on quantum data securely against so-called specious adversaries. The protocol therein is unconditionally secure given an ideal functionality for classical computation. Specious adversaries are a quantum version of the classical notion of passive adversaries. Technically, a specious adversary is an adversary which is allowed to deviate from the protocol, except that at any step of the protocol it should be able to reconstruct the honest state of the protocol from its current state. This basically allows it to purify itself and not much else.

In the protocol in [4], all wires are encrypted using a Pauli encryption: a qubit $|v\rangle$ is represented as $|V\rangle = \mathsf{X}^x\mathsf{Z}^z|v\rangle$, where the two uniform key bits x and z are secret-shared between Alice and Bob. For example, to secret-share x, Alice will hold a uniformly random bit x_A and Bob will hold a uniformly random bit x_B such that $x = x_A \oplus x_B$. All wires are independently encrypted like this, which ensures that intermediate states are perfectly hidden from both parties. Computation is then done "through the encryption". The CPTP map $\mathscr{F}$ is described by a quantum circuit made out of gates in the universal set X, Y, Z, CNOT, $\mathsf{H} = \frac{1}{\sqrt{2}}\left(\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix}\right)$, $\mathsf{P} = \left(\begin{smallmatrix} 1 & 0 \\ 0 & i \end{smallmatrix}\right)$ and $\mathsf{R} = \left(\begin{smallmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{smallmatrix}\right)$, together with a set of ancilla wires initialized in state $|0\rangle$, and a set of output wires that are discarded (or, in our case, that remain encrypted forever). The protocol evaluates each gate of $\mathscr{F}$ while preserving privacy. Handling the Pauli gates X, Y and Z is easy, as they commute or anti-commute with the encryption operators.

As an example, assume that the parties want to apply an X gate to a qubit $|v\rangle$, i.e., compute an encryption of $|v'\rangle = \mathsf{X}|v\rangle$. Since up to an overall phase factor $\mathsf{XZ} = \mathsf{ZX}$, it follows that $\mathsf{X}|V\rangle = \mathsf{X}\mathsf{X}^x\mathsf{Z}^z|v\rangle = \mathsf{X}^x\mathsf{Z}^z\mathsf{X}|v\rangle = \mathsf{X}^x\mathsf{Z}^z|v'\rangle$. So, the evaluation is simply performed on the encrypted qubit $|V\rangle$ and the key bits x and z are maintained. For the remaining Clifford gates CNOT, H and P other "commutation" rules with X and Z are used. For H, it is used that $\mathsf{HX} = \mathsf{ZH}$ and $\mathsf{HZ} = \mathsf{XH}$, so H is simply applied to the encrypted qubit, and then the keys x and z are swapped: Alice sets $x'_A = z_A$ and $z'_A = x_A$ and similarly for Bob. This leaves the non-Clifford gate R, where the relation $\mathsf{RX}^x\mathsf{Z}^z = \mathsf{P}^x\mathsf{X}^x\mathsf{Z}^z\mathsf{R}$ almost does the job, except that it leaves an extra P^x. Getting rid of this requires quantum computation and a classical secure two-party computation which computes how the parties should update their key bits. After such a gate-by-gate computation of U on the encrypted qubits, the shares of the keys needed for learning the states on ones own output wires are swapped with the other party in one atomic step, using the ideal functionality for classical secure computation.

The protocol in [4] is actually secure against an active adversary up to the final swap of key bits, as the encryptions of the wires are guaranteed to be perfect, independent of the behavior of the other party, as the parties pick their own shares of the sharings of x and z. This means that no party can get any information on any intermediary states, no matter how it deviates. It can, however, easily force the computation to be incorrect, by applying gates to the encrypted states, so the full protocol is not active secure. We note, however, that [4] is secure against what we could call *ultimately specious adversaries*: Adversaries promising to attack in such a way that both parties always be able to reconstruct the correct state at the *end* of the protocol, but that can otherwise behave as they want. This follows from the active security in the middle and a theorem in [4] which says that any attack which always allows both parties to obtain the correct output can be simulated given just the output of the corrupted party—basically, there is no way to learn extra information without sometimes irrevocably destroying the state of the other party.

In this paper, we use this observation in a protocol proceeding along the same lines as [4] but where we force the adversary to be ultimately specious. This is done by not only encrypting the wires, but by unconditionally authenticating them. In addition, we commit the parties to their key bits, to allow the recipient to verify the key bits swapped at the end. Since an unconditional quantum authentication code is also an unconditionally secure encryption, we get a protocol with at least the security of [4], but with the added property that an adversary who deviates from the protocol will be caught. More technically, if all checks of the authentication code succeed, then the authenticated values collapse to the correct values. This forces the adversary to either be detected or be ultimately specious. Since we do all the checks of the authentication codes *before* any key bits are revealed, the case where the adversary is detected can be simulated by simply asking the ideal world to abort too. The case where the adversary is ultimately specious is simulated similarly to [4].

The main technical challenge is then to devise an authentication code with these two properties:

1. It allows to perform computation "through the authentication".
2. It allows to check the authentication code without revealing what is authenticated.

The first property is important for hiding intermediate values during the computation. The second property is important when we force the adversary to either be detected or ultimately specious: when he is detected, he should learn nothing on the incorrect outputs.

We devise an authentication code with these properties based on the Clifford-based quantum authentication code proposed in [1]. It authenticates a quantum message using a random unitary implementable using gates X, Y, Z, CNOT, H and P. The authentication works as follows. In order to satisfy the second property, take a qubit $|v\rangle$ on some wire. Alice prepends k new wires, in the all-zero state $|0^k\rangle$. Then she applies a uniformly random $(k+1)$-bit Clifford operator A to the $k+1$ wires. She then sends the state to Bob, who appends k more wires in the all-zero state and applies a uniformly random $(2k+1)$-bit Clifford operator B to the $2k+1$ wires. We can write this as $|V\rangle = B(|0^k\rangle \otimes A(|v\rangle|0^k\rangle))$. The key is (A, B). This authentication can checked in two different ways. Either, apply $B^\dagger$ to the authenticated state and check that the first k wires are all zero. Or, apply $B^\dagger$ and then apply $A^\dagger$ to the last $k+1$ wires and check that the last k wires are all zero. One way is used for Alice to check that Bob did not change the authenticated value. The other is used by Bob to check Alice. In order to perform these without leakage, we use a 2-party classical ideal functionality as described below.

In our scheme, Alice will hold A as her share of the key and Bob will hold B as his key share. They are committed to their share by being committed to a poly-size classical description of the operator applied. We sketch why the scheme has the two necessary properties.

1. Since the authentication is performed using only Clifford gates, an approach as in [4] will allow to fairly easily apply Clifford gates "through the authentication". The R gate requires new techniques reminiscent of the fault tolerant implementation of it [14,6,7]. The details appear within, but we basically reduce it to securely producing a state $|0\rangle$, a magic state, a measurement in the computational basis, and applying secure Clifford gates together with a carefully chosen secure classical computation which tells the parties how to update their keys.
2. If Bob is to check an authenticated qubit, we simply give him $|V\rangle$ and he applies $B^\dagger$ and measures the first k wires, rejecting if they are not all zero. After that he re-applies B to recover the authentication $|V\rangle$. If Alice is to perform the check, we perform a secure classical computation where the inputs are the committed descriptions of A and B. The output to Alice is a canonical description of the $(2k+1)$-bit Clifford unitary $D = (C \otimes \mathbb{1}_k)(\mathbb{1}_k \otimes A^\dagger)B^\dagger$, where $\mathbb{1}_k$ is the identity unitary on k wires and C is a uniformly random Clifford unitary on $k+1$ wires. On a correct authenticated state $|V\rangle = B(|0^k\rangle A(|v\rangle|0^k\rangle))$, we have that $D|V\rangle = C(|0^k\rangle|v\rangle)|0^k\rangle$, and in general it can be shown that D exactly gives Alice access to her k check wires and no extra information. This allows Alice to perform her check without learning anything on $|v\rangle$ or B. After checking her wires, Alice applies $D^\dagger$ to recover the authenticated state. We use these checks before key bits are swapped, but we also use them in certain intermediary steps, to ensure that the parties either abort or hold a correct, authenticated state.

The final state of the computation is obtained after each party reveals the authentication keys needed by the other party to open its output wires.

2 Preliminaries

The set of linear operators from and to Hilbert space $\mathcal{A}$ is denoted by $\mathrm{L}(\mathcal{A})$. The set of trace 1 positive semi-definite operators is denoted by $\mathrm{D}(\mathcal{A})$; it is the set of quantum states for register $\mathcal{A}$. For $\rho, \rho' \in \mathrm{D}(\mathcal{A})$, we denote by $\Delta(\rho, \rho') := \frac{1}{2}\|\rho - \rho'\|_1$ the trace norm distance between ρ and ρ'. More information can be found in [5].

2.1 Secure Two-Party Classical Computation Against Quantum Adversaries

Our protocol will use various *classical* two-party computations throughout its execution, each modeled as an ideal functionality. Recent work [12,11] show that composable classical two-party computation protocols can be devised with security against quantum adversaries provided some classical computational assumptions hold against this class of adversaries. One example of such an assumption is *learning with errors is hard* [13,11]. The framework in [11] allows us to replace the ideal functionalities with such secure protocols. Here we therefore focus on proving security given the ideal functionalities.

In the following, the ideal functionality for string commitment will be denoted by id_{SC}. It is defined as follows:

$$\begin{aligned}
\mathsf{id}_{\text{SC}}((id, s), \bot) &= (\bot, (id, \texttt{committed})) \text{ if } id \text{ is new,} \\
\mathsf{id}_{\text{SC}}(\bot, (id, s)) &= ((id, \texttt{committed}), \bot) \text{ if } id \text{ is new,} \\
\mathsf{id}_{\text{SC}}(s, s') &= (\bot, \bot) \text{ otherwise .}
\end{aligned}$$

In order for Alice to commit on $s \in \{0,1\}^*$, Alice and Bob call $\mathsf{id}_{\text{SC}}((id, s), \bot)$, where id is an unused identifier chosen by Alice. In order for Bob to commit on s, Alice and Bob call $\mathsf{id}_{\text{SC}}(\bot, (id, s))$, where id is chosen by Bob. The opening of a commitment is performed by calling the ideal functionality $\mathsf{id}_{\text{OPEN}}$ defined as:

$$\begin{aligned}
\mathsf{id}_{\text{OPEN}}(id, id) &= (\bot, s) \text{ if } \mathsf{id}_{\text{SC}}((id, s), \bot) \text{ was performed ,} \\
\mathsf{id}_{\text{OPEN}}(id, id) &= (s, \bot) \text{ if } \mathsf{id}_{\text{SC}}(\bot, (id, s)) \text{ was performed ,} \\
\mathsf{id}_{\text{SC}}(\cdot, \cdot) &= (\bot, \bot) \text{ otherwise .}
\end{aligned}$$

Note that state has to be passed from $\mathsf{id}_{\text{SC}}(\ ,\ )$ to $\mathsf{id}_{\text{OPEN}}(\ ,\ )$ for the above descriptions to make sense. Our framework exactly allows that an ideal functionality from an earlier round passes its state to an ideal functionality in a later round. In the framework of [11] they would be considered one ideal functionality, with a state. We prefer the above notation for brevity of later protocol descriptions.

Ideal functionalities computing functions applied to committed values can now be easily defined. Suppose that Alice and Bob want to let Alice learn a function $f : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ upon two committed values, s and s', where s is committed upon by Alice under id and s' is committed upon by Bob under id'. It suffices to define the ideal functionality id^*_f as

$$\begin{aligned}
\mathsf{id}^*_f((id, id'), (id, id')) &= (f(s, s'), \bot) \text{ if } \mathsf{id}_{\text{OPEN}}(id, id) = (\bot, s) \wedge \mathsf{id}_{\text{OPEN}}(id', id') = (s', \bot) \text{ ,} \\
\mathsf{id}^*_f(\cdot, \cdot) &= (\bot, \bot) \text{ otherwise .}
\end{aligned}$$

The same if it is Bob who is to learn the output, but with $\mathsf{id}_f^*((id, id'), (id, id')) = (\perp, f(s, s'))$. The same construction can be used to implement any efficiently computable function f evaluated upon *any* number of committed values. The ideal functionality can also easily be extended to produce commitments to the outputs of f. It is this extended ideal functionality we use most often. Again, $\mathsf{id}_f^*(\cdot,\cdot)$ will have to share state with $\mathsf{id}_{\text{SC}}(\cdot,\cdot)$ and $\mathsf{id}_{\text{OPEN}}(\cdot,\cdot)$, which is allowed in our framework.

Note that all the above ideal functionalities are defined such that at most one party has a non-trivial output (i.e., an output which is not known before the inputs are provided). We avoid using functions where both parties have a non-trivial output as an easy way to deal with the problem that fairness in classical secure two-party computation is provably impossible for most functionalities. It is not hard to see that it follows from known completeness results of quantum-secure classical two-party computation[12,11] that the classical ideal functionalities specified above can be implemented with security against poly-time quantum adversaries. We skip the details of this, as our focus here is on using the classical ideal functionalities for constructing secure two-party protocols for computing on quantum data.

2.2 Clifford-Based Quantum Authentication

In order to detect misbehaviors of an active adversary, we will be evaluating a circuit upon authenticated quantum bits. We will be using a quantum authentication scheme (QAS) [2] based on Clifford operators introduced in [1] as our main building block.

Definition 2.1 (Quantum authentication scheme). *A* quantum authentication scheme *is a set of encryption and decryption superoperators* $\{(\mathscr{E}_k^{S\to C}, \mathscr{D}_k^{C\to SF}) : k \in \mathcal{K}\}$, *where* $\mathcal{K}$ *is the set of possible keys,* S *is the input system,* C *is the ciphertext system, and* F *is a "flag" system that contains either* $|\mathrm{acc}\rangle$ *or* $|\mathrm{rej}\rangle$. *A QAS is such that for all* $k \in \mathcal{K}$, $(\mathscr{D}_k \circ \mathscr{E}_k)(\rho_S) = \rho_S \otimes |\mathrm{acc}\rangle\langle\mathrm{acc}|_F$.

A QAS is secure if it satisfies the following:

Definition 2.2 (Security of a QAS). *Let* $\mathscr{E}_k^{S\to C}$ *and* $\mathscr{D}_k^{C\to SF}$ *be the encoder and decoder corresponding to key* k. *Then, we say that the QAS* $(\mathscr{E}, \mathscr{D})$ *is* ε*-secure if, for all attacks* U_{CR}, *there exists two CP maps* $\mathscr{U}_{R\to R}^{\mathrm{acc}}$ *and* $\mathscr{U}_{R\to R}^{\mathrm{rej}}$ *with* $\mathscr{U}^{\mathrm{acc}} + \mathscr{U}^{\mathrm{rej}} = \mathbb{1}$ *such that for all inputs* ψ_{SR}, *we have that for some fixed state* Ω_S*:*

$$\begin{aligned}\Big\| \mathbb{E}_k \mathscr{D}_k \left(U_{CR} \mathscr{E}_k \left(\psi_{SR} \right) U_{CR}^\dagger \right) \\ - \left(\mathscr{U}^{\mathrm{acc}}(\psi_{SR}) \otimes |\mathrm{acc}\rangle\langle\mathrm{acc}|_F + \mathscr{U}^{\mathrm{rej}}(\psi_R) \otimes \Omega_S \otimes |\mathrm{rej}\rangle\langle\mathrm{rej}|_F \right) \Big\|_1 \leqslant \varepsilon,\end{aligned} \tag{1}$$

where the expectation is taken over the uniform distribution on $\mathcal{K}$.

This definition can be shown to be equivalent to the existence of a simulator that interacts only with an ideal functionality in which Eve's only choice is whether or not to destroy the state and cause a rejection.

Definition 2.3 (Clifford-based QAS[1]). *Let S be an s–qubit system, A be an n-qubit system, and let $C = S \otimes A$. Let $\mathcal{K}$ index all Clifford unitaries C_k on $s+n$ qubits. Then, the Clifford-based QAS is defined by the following encryption and decryption maps where $P^{SA}_{\rm acc} = \mathbb{1}_S \otimes |0^n\rangle\langle 0^n|_A$ and $P^{SA}_{\rm rej} = \mathbb{1}_{SA} - P^{SA}_{\rm acc}$:*

$$\begin{aligned}\mathscr{E}_k(\rho_S) &= C_k\left(\rho_S \otimes |0^n\rangle\langle 0^n|_A\right) C_k^\dagger \ , \\ \mathscr{D}_k(\sigma_{SA}) &= \mathrm{tr}_A\Big(P_{\rm acc} C_k^\dagger \sigma_{SA} C_k P_{\rm acc} \otimes |{\rm acc}\rangle\langle{\rm acc}|_F \\ &\qquad + \mathrm{tr}\left(P_{\rm rej} C_k^\dagger \sigma_{SA} C_k\right)\pi_{SA} \otimes |{\rm rej}\rangle\langle{\rm rej}|_F\Big) \ .\end{aligned}$$

The following establishes the security of the QAS based on random Clifford operators. The proof of security is more or less the same as in [1] :

Theorem 2.4 (Security of Clifford-based QAS). *The QAS defined above is $\varepsilon(n)$-secure for $\varepsilon(n) = 6 \times 2^{-n}$.*

It should be mentioned that picking a random Clifford operation acting upon ℓ qubits requires to pick a uniformly random $\mathrm{poly}(\ell)$-bit classical key k and the mapping between k and the corresponding Clifford operation can be performed efficiently [6,1]. In other words, the key size of the Clifford-based QAS is polynomial in the number of qubits $\ell = s + n$ used to authenticate an s–qubit quantum state.

2.3 Two-Party Quantum Protocols

We define two-party strategies in a similar way as in [4,10], with some adaptations made for the fact that we are computing a CPTP map and not just a unitary and that we allow ideal functionalities of different rounds to share states (equivalent to considering one, stateful functionality). Two-party protocols for the evaluation of some CPTP map are particular cases of two-party strategies. Two-party strategies have access to some oracle in each round. An oracle is just a CPTP map acting on registers at both Alice and Bob. Oracles implement some functionalities like a communication channel or some more complex two-party functionalities.

An m–turn two-party strategy with oracle calls is defined by tuples of quantum operations $\mathscr{A} := (\mathscr{A}_1, \ldots, \mathscr{A}_{m+1})$, $\mathscr{B} := (\mathscr{B}_1, \ldots, \mathscr{B}_{m+1})$, and $\mathscr{O} := (\mathscr{O}_1, \ldots, \mathscr{O}_m)$. For $i \in [1..m+1]$, operations $\mathscr{A}_i \in \mathrm{L}(\mathcal{A}_{i-1} \otimes \mathcal{A}_{i-1}^{\mathscr{O},\mathrm{out}}) \mapsto \mathrm{L}(\mathcal{A}_i \otimes \mathcal{A}_i^{\mathscr{O},\mathrm{in}})$ and $\mathscr{B}_i \in \mathrm{L}(\mathcal{B}_{i-1} \otimes \mathcal{B}_{i-1}^{\mathscr{O},\mathrm{out}}) \mapsto \mathrm{L}(\mathcal{B}_i \otimes \mathcal{B}_i^{\mathscr{O},\mathrm{in}})$ are the actions performed at turn i by Alice and Bob respectively. The operation $\mathscr{O}_i : \mathrm{L}(\mathcal{A}_i^{\mathscr{O},\mathrm{in}} \otimes \mathcal{O}_{i-1} \otimes \mathcal{B}_i^{\mathscr{O},\mathrm{in}}) \mapsto \mathrm{L}(\mathcal{A}_i^{\mathscr{O},\mathrm{out}} \otimes \mathcal{O}_i \otimes \mathcal{B}_i^{\mathscr{O},\mathrm{out}})$ models the oracle provided to Alice and Bob at turn i, and $\mathcal{O}_i$ a register used for passing state from the oracle of one round to the oracle at the next round. The oracle at turn $i \in [1..m]$ takes place right after $\mathscr{A}_i$ and $\mathscr{B}_i$ have been applied. In particular, these operations set the input registers $\mathcal{A}_{i-1}^{\mathscr{O},\mathrm{in}}$ and $\mathcal{B}_{i-1}^{\mathscr{O},\mathrm{in}}$ for the call to $\mathscr{O}_i$. The outputs are available to Alice and Bob in the next turn, in the output registers $\mathcal{A}_i^{\mathscr{O},\mathrm{out}}$ and $\mathcal{B}_i^{\mathscr{O},\mathrm{out}}$. We make one exception to this general form, the last turn (that is turn $m+1$) of a strategy does not invoke any oracle, and is there simply to allow Alice and Bob to post-process the output of the last oracle.

Let $\Pi = (\mathscr{A}, \mathscr{B}, \mathscr{O}, m)$ be an m-turn two-party strategy with oracle calls. The final state of the interaction between $\mathscr{A}$ and $\mathscr{B}$ upon joint input state $\rho_{\mathrm{in}} \in \mathrm{D}(\mathcal{A}_0 \otimes \mathcal{B}_0 \otimes \mathcal{R})$, where $\mathcal{R}$ is a reference system with $\dim \mathcal{R} = \dim \mathcal{A}_0 \dim \mathcal{B}_0$, is denoted by

$$[\mathscr{A} \circledast \mathscr{B}]^{\mathscr{O}}(\rho_{\mathrm{in}}) := (\mathscr{A}_{m+1} \otimes \mathscr{B}_{m+1} \otimes \mathbb{1}_{\mathrm{L}(\mathcal{R} \otimes \mathcal{O}_{m+1})})(\mathbb{1}_{\mathrm{L}(\mathcal{A}_m \otimes \mathcal{B}_m \otimes \mathcal{R})} \otimes \mathscr{O}_m)(\mathscr{A}_m \otimes \mathscr{B}_m \otimes \mathbb{1}_{\mathrm{L}(\mathcal{R} \otimes \mathcal{O}_m)})$$
$$\ldots (\mathbb{1}_{\mathrm{L}(\mathcal{A}_1 \otimes \mathcal{B}_1 \otimes \mathcal{R})} \otimes \mathscr{O}_1)(\mathscr{A}_1 \otimes \mathscr{B}_1 \otimes \mathbb{1}_{\mathrm{L}(\mathcal{R})})(\rho_{\mathrm{in}}) \ .$$

A *communication oracle* from Alice to Bob is modeled by having $\mathcal{A}_i^{\mathscr{O}} \approx \mathcal{B}_i^{\mathscr{O}}$ and letting $\mathscr{O}_i$ move the state in $\mathcal{A}_i^{\mathscr{O}}$ to $\mathcal{B}_i^{\mathscr{O}}$. A classical ideal functionality, as those described in Sect. 2.1, can easily be made available as oracle calls. We assume that in each round i at most one classical ideal functionality is applied. The parties place their inputs in the appropriate registers $\mathcal{A}_{i-1}^{\mathscr{O}}$ and $\mathcal{B}_{i-1}^{\mathscr{O}}$. The operation of the oracle $\mathscr{O}_i$ is as follows: it measures the input registers $\mathcal{A}_{i-1}^{\mathscr{O}}$ and $\mathcal{B}_{i-1}^{\mathscr{O}}$ to force classical inputs. Then, it applies the appropriate classical ideal functionality on those classical inputs plus its classical internal state found in $\mathcal{O}_{i-1}$. This produces outputs for the parties and a new internal state. The outputs are placed in $\mathcal{A}_i^{\mathscr{O}}$ and $\mathcal{B}_i^{\mathscr{O}}$. The new state is placed in $\mathcal{O}_i$.

A two-party hybrid protocol for $\mathscr{F} : \mathrm{L}(\mathcal{A}_{\mathrm{in}} \otimes \mathcal{B}_{\mathrm{in}}) \to \mathrm{L}(\mathcal{A}_{\mathrm{out}} \otimes \mathcal{B}_{\mathrm{out}})$ between parties $\mathscr{A}$ and $\mathscr{B}$ upon joint input state $\rho_{\mathrm{in}} \in \mathrm{D}(\mathcal{A}_{\mathrm{in}} \otimes \mathcal{B}_{\mathrm{in}} \otimes \mathcal{R})$ is then defined as follows:

Definition 2.5. *An* m-turn two-party hybrid protocol $\Pi^{\mathscr{O}}_{\mathscr{F}} = (\mathscr{A}, \mathscr{B}, \mathscr{O}, m)$ for $\mathscr{F} : \mathrm{L}(\mathcal{A}_{\mathrm{in}} \otimes \mathcal{B}_{\mathrm{in}}) \to \mathrm{L}(\mathcal{A}_{\mathrm{out}} \otimes \mathcal{B}_{\mathrm{out}})$ *is a* m*–turn two-party strategy with oracle calls, where* $\mathcal{A}_0 := \mathcal{A}_{\mathrm{in}}$*,* $\mathcal{B}_0 := \mathcal{B}_{\mathrm{in}}$*,* $\mathcal{A}_{m+1} := \mathcal{A}_{\mathrm{out}}$*,* $\mathcal{B}_{m+1} := \mathcal{B}_{\mathrm{out}}$*, and where for all* $\rho_{\mathrm{in}} \in \mathrm{D}(\mathcal{A}_0 \otimes \mathcal{B}_0 \otimes \mathcal{R})$*,* $\Delta\left([\mathscr{A} \circledast \mathscr{B}]^{\mathscr{O}}(\rho_{\mathrm{in}}), (\mathscr{F} \otimes \mathbb{1}_{\mathcal{R}})(\rho_{\mathrm{in}})\right) = 0$*. In the following, we often write simply* two-party protocol *to refer to a* two-party hybrid protocol*.*

For $i \in [0..m]$, the joint state after turn $i+1$ in $\Pi^{\mathscr{O}}_{\mathscr{F}}$ is denoted by $[\mathscr{A} \circledast \mathscr{B}]^{\mathscr{O}}_{i+1}(\rho_{\mathrm{in}}) := (\mathbb{1}_{\mathrm{L}(\mathcal{B}_{i+1} \otimes \mathcal{A}_{i+1} \otimes \mathcal{R})} \otimes \mathscr{O}_{i+1})(\mathscr{A}_{i+1} \otimes \mathscr{B}_{i+1} \otimes \mathbb{1}_{\mathrm{L}(\mathcal{R} \otimes \mathcal{O}_{i+1})})[\mathscr{A} \circledast \mathscr{B}]^{\mathscr{O}}_i(\rho_{\mathrm{in}})$, where $[\mathscr{A} \circledast \mathscr{B}]^{\mathscr{O}}_0(\rho_{\mathrm{in}}) := \rho_{\mathrm{in}}$, and $[\mathscr{A} \circledast \mathscr{B}]^{\mathscr{O}}_{m+1}(\rho_{\mathrm{in}}) := [\mathscr{A} \circledast \mathscr{B}]^{\mathscr{O}}(\rho_{\mathrm{in}})$.

3 Modeling Active Security

We start by extending the framework in [4] to handle active security. Our model is very standard, defining security via simulation, but for completeness we describe and motivate the changes made in [5].

Let $\Pi^{\mathscr{O}}_{\mathscr{F}} = (\mathscr{A}, \mathscr{B}, \mathscr{O}, m)$ be a m-turn two-party hybrid protocol. Let $\tilde{\mathscr{A}}$ and $\tilde{\mathscr{B}}$ be adversaries in $\Pi^{\mathscr{O}}_{\mathscr{F}}$. We denote by $[\tilde{\mathscr{A}} \circledast \mathscr{B}]^{\mathscr{O}}$ and $[\mathscr{A} \circledast \tilde{\mathscr{B}}]^{\mathscr{O}}$ the resulting m–turn two-party strategies. We will as usual define the security of such actively attacked protocols by comparing to a simulation. The simulation is basically an ideally secure evaluation of $\mathscr{F}$.

The ideally secure protocol for evaluating $\mathscr{F}$ would be in a world where $\mathscr{F}$ actually existed as an oracle—the parties would simply call this oracle. We can, however, not expect any protocol to be as secure as this, as for most protocols we cannot ensure that either both parties get the output or no party gets the output, known as fairness, which is ensured in the above ideal setting. We will therefore consider a setting where one of the parties learns its output first, and where the party learning it first, if corrupted, can

prevent the other party from learning its output. We will only give the definition for the case where Alice learns first. Deriving the definition for the symmetric case where Bob learns first is trivial.

To avoid confusion between the parties in the real protocol and in the ideal protocol, we formulate the ideal protocol with parties Charleen, $\mathcal{C}$, and Dan, $\mathcal{D}$, taking the seats of Alice respectively Bob. So, we look at an ideal functionality $\mathscr{F} : \mathrm{L}(\mathcal{C}_{\mathrm{in}} \otimes \mathcal{D}_{\mathrm{in}}) \to \mathrm{L}(\mathcal{C}_{\mathrm{out}} \otimes \mathcal{D}_{\mathrm{out}})$, but where we keep a mental note reminding that $\mathcal{C}_{\mathrm{in}} := \mathcal{A}_{\mathrm{in}}$, $\mathcal{C}_{\mathrm{out}} := \mathcal{A}_{\mathrm{out}}$, $\mathcal{D}_{\mathrm{in}} := \mathcal{B}_{\mathrm{in}}$ and $\mathcal{D}_{\mathrm{out}} := \mathcal{B}_{\mathrm{out}}$.

The ideal protocol for $\mathscr{F}$ is then a 2-turn two-party hybrid protocol $\Gamma_{\mathscr{F}} = (\mathscr{C}, \mathscr{D}, \mathscr{F}, 2)$, which lets the parties query $\mathscr{F}$ in turn 1, but only letting $\mathcal{C}$ see her output in turn 1. In turn 2, Charleen then inputs a bit f, with $f = 1$ indicating that Dan should receive his output and $f = 0$ indicating that Dan should not receive his output. Dan will receive f, and if $f = 1$ he will additionally be given his output. This is handled by the second oracle. Then the parties output whatever they received from the oracles. In the honest protocol, Charleen always inputs $f = 1$. We call this *the ideal protocol for evaluating $\mathscr{F}$ without fairness for Dan*.

The ideal protocol $\Gamma_{\mathscr{F}} = (\mathscr{C}, \mathscr{D}, \mathscr{F}, 2)$ for $\mathscr{F}$ without fairness for Dan:

1. By convention we have $\mathcal{C}_0 = \mathcal{C}_{\mathrm{in}}$ and $\mathcal{D}_0 = \mathcal{D}_{\mathrm{in}}$ and that $\mathcal{O}_0$ is empty. In the ideal protocol we let $\mathcal{C}_1^{\mathscr{O},\mathrm{in}} = \mathcal{C}_0$ and $\mathcal{D}_1^{\mathscr{F},\mathrm{in}} = \mathcal{D}_0$, we let $\mathcal{C}_1$ and $\mathcal{D}_1$ be empty, and $\mathscr{C}_1 = \mathbb{1}_{\mathrm{L}(\mathcal{C}_0)}$ and $\mathscr{D}_1 = \mathbb{1}_{\mathrm{L}(\mathcal{D}_0)}$. I.e., in turn 1 the parties simply send their inputs to the first oracle $\mathscr{F}_1$. For the first oracle $\mathscr{F}_1 : \mathrm{L}(\mathcal{C}_1^{\mathscr{F},\mathrm{in}} \otimes \mathcal{O}_0 \otimes \mathcal{D}_1^{\mathscr{F},\mathrm{in}}) \mapsto \mathrm{L}(\mathcal{C}_1^{\mathscr{F},\mathrm{out}} \otimes \mathcal{O}_1 \otimes \mathcal{D}_1^{\mathscr{F},\mathrm{out}})$, we set $\mathcal{C}_1^{\mathscr{F},\mathrm{out}} \approx \mathcal{C}_{\mathrm{out}}$, $\mathcal{O}_1 \approx \mathcal{D}_{\mathrm{out}}$ and we let $\mathcal{D}_1^{\mathscr{F},\mathrm{out}}$ be empty. We let $\mathscr{F}_1 = \mathscr{F}$. I.e., $\mathscr{F}_1$ simply applies $\mathscr{F}$ to the inputs supplied by the parties, sends Charleen's output to Charleen in $\mathcal{C}_1^{\mathscr{F},\mathrm{out}}$ and saves Dan's output in the internal state $\mathcal{F}_1$ of the oracle, giving Dan no output so far.
2. We let $\mathcal{C}_2 \approx \mathcal{C}_1^{\mathscr{F},\mathrm{out}} (\approx \mathcal{C}_{\mathrm{out}})$ and we let $\mathcal{C}_2^{\mathscr{F},\mathrm{in}}$ be a one qubit register, holding a qubit we name $|f\rangle$. We let $\mathscr{C}_2 = \mathbb{1}_{\mathrm{L}(\mathcal{C}_1^{\mathscr{F},\mathrm{out}},\mathcal{C}_2)} \otimes |1\rangle$, i.e., it moves the output from the oracle from $\mathcal{C}_1^{\mathscr{F},\mathrm{out}}$ to $\mathcal{C}_2$ and it sets $|f\rangle = |1\rangle$. We let $\mathcal{D}_2^{\mathscr{F},\mathrm{in}}$ and $\mathcal{D}_2$ be empty, so there is no need to specify $\mathscr{D}_2$.
   For the second oracle $\mathscr{O}_2 : \mathrm{L}(\mathcal{C}_2^{\mathscr{F},\mathrm{in}} \otimes \mathcal{O}_1 \otimes \mathcal{D}_2^{\mathscr{F},\mathrm{in}}) \mapsto \mathrm{L}(\mathcal{C}_2^{\mathscr{F},\mathrm{out}} \otimes \mathcal{O}_2 \otimes \mathcal{D}_2^{\mathscr{F},\mathrm{out}})$, we let $\mathcal{C}_2^{\mathscr{F},\mathrm{out}}$ and $\mathcal{O}_2$ be empty, and we let $\mathcal{D}_2^{\mathscr{F},\mathrm{out}}$ be of the same dimension as $\mathcal{D}_{\mathrm{out}}$, plus room for one qubit $|a\rangle$. It starts by measuring $|f\rangle$ in the computational basis. If $|f\rangle = |1\rangle$, it then sets $\mathcal{D}_2^{\mathscr{F},\mathrm{out}}$ to hold $|a\rangle = |1\rangle$ along with the state in $\mathcal{O}_1$. If $|f\rangle = |0\rangle$, it sets $\mathcal{D}_2^{\mathscr{F},\mathrm{out}}$ to hold $|a\rangle = |0\rangle$ along with some fixed dummy state $|\perp\rangle$ of the right dimension to fill $\mathcal{C}_2^{\mathscr{F},\mathrm{out}}$. I.e., if Charleen inputs $f = 1$, then Dan will get his output. If $f = 0$, Dan gets no output, except a bit $a = 0$ telling him that Charleen cheated him of his output (we say that Charleen *aborted* the computation).
3. We let $\mathcal{C}_3 \approx \mathcal{C}_2$ and $\mathscr{C}_3 = \mathbb{1}_{\mathrm{L}(\mathcal{C}_2,\mathcal{C}_3)}$. We let $\mathcal{D}_3 \approx \mathcal{D}_2^{\mathscr{F},\mathrm{out}}$ and $\mathscr{D}_3 = \mathbb{1}_{\mathrm{L}(\mathcal{D}_2^{\mathscr{F},\mathrm{out}},\mathcal{D}_3)}$. I.e., in the last round the parties just output whatever they received from $\mathscr{F}$, with Dan possibly outputting a dummy state, in case of abort.

Consider the powers of a corrupted Charleen, $\tilde{\mathscr{C}}$. She might in the first round provide an alternative input for $\mathscr{F}$, possibly saving her original input, or a part thereof, in some ancilla. This is *input substitution*. Her choice of alternative input can only depend on her own original input, not that of Dan. This is *input independence*. Then she learns only the output of the oracle. This is *privacy*. After learning her own output she might

then specify that Dan is not to learn his output. This is the necessary *lack of fairness*. A corrupted Dan has similar powers, except that he cannot abort after seeing his output. We then say that a protocol is secure if it only allows attacks which could be mounted in the ideal protocol, i.e., it only allows the inevitable attack.

Definition 3.1. *Let $\Pi^{\mathscr{O}}_{\mathscr{F}} = (\mathscr{A}, \mathscr{B}, \mathscr{O}, m)$ be a two-party hybrid protocol for $\mathscr{F} : \mathrm{L}(\mathcal{A}_{\mathrm{in}} \otimes \mathcal{B}_{\mathrm{in}}) \to \mathrm{L}(\mathcal{A}_{\mathrm{out}} \otimes \mathcal{B}_{\mathrm{out}})$. Let $\Gamma_{\mathscr{F}}$ be the ideal protocol for $\mathscr{F}$ without fairness for Dan. Let $0 \leq \delta \leq 1$. We say that $\Pi^{\mathscr{O}}_{\mathscr{F}}$ is* δ-active secure without fairness for Bob*, if for all adversaries $\tilde{\mathscr{A}}$ and $\tilde{\mathscr{B}}$ in $\Pi^{\mathscr{O}}_{\mathscr{F}}$, there exist adversaries $\tilde{\mathscr{C}}$ and $\tilde{\mathscr{D}}$ in $\Gamma_{\mathscr{F}}$ with sizes polynomial in the sizes of $\tilde{\mathscr{A}}$ and $\tilde{\mathscr{B}}$ such that for all input states $\rho_{\mathrm{in}} \in \mathrm{D}(\mathcal{A}_{\mathrm{in}} \otimes \mathcal{B}_{\mathrm{in}} \otimes \mathcal{R})$,*

$$\Delta\left([\tilde{\mathscr{A}} \circledast \mathscr{B}]^{\mathscr{O}}(\rho_{\mathrm{in}}), [\tilde{\mathscr{C}} \circledast \mathscr{D}]^{\mathscr{F}}(\rho_{\mathrm{in}})\right) \leq \delta \text{ and } \Delta\left([\mathscr{A} \circledast \tilde{\mathscr{B}}]^{\mathscr{O}}(\rho_{\mathrm{in}}), [\mathscr{C} \circledast \tilde{\mathscr{D}}]^{\mathscr{F}}(\rho_{\mathrm{in}})\right) \leq \delta \ .$$

If a protocol is δ-active secure for $\delta = 0$, then it is called perfectly active secure*. If it is δ-active secure for δ negligible in some security parameter n, then it is called* statistically active secure *(in n).*

4 Securely Swaddling Wires to Ensure Ultimate Speciousness

The Clifford based QAS, described in Sect. 2.2, can be used to share the authentication of a quantum message in a way that allows any of the players, when helped by a TPC, to verify that a state has not been tampered with, and this without being able to get information on the encoded state. This primitive will be used extensively in our protocol to ensure that all players are *ultimately specious*. It relies on a string commitment scheme and secure TPCs provided as oracles.

The basic idea consists in swaddling each input wire $\mathtt{w}$ into a set of $2n$ dummy wires where n *belongs* to Alice and n *belongs* to Bob. No party will be able to extract information about the state of the original wire from the swaddling. Moreover, any attempt to modify the state of the original wire will be detected except with negligible probability in n. The sub-protocol $\mathtt{Swaddle}(\mathtt{w})$, described below, uses two application of the Clifford-based QAS (one on top of the other) in order to achieve this. Alice authenticates the state of her wire $\mathtt{w}$ and commits to her authentication key $K_{\mathtt{w},a}$ using identifier $id_A(\mathtt{w})$. She then sends the resulting system to Bob who authenticates it using key $K_{\mathtt{w},b}$ that he also commits upon using identifier $id_B(\mathtt{w})$. Bob sends back the resulting system to Alice allowing her to test the validity of the swaddling. The swaddling therefore uses a total of $2n + 1$ wires, where Alice holds the *outermost* and the *innermost* key of the resulting swaddling $\mathfrak{s}(\mathtt{w})$ of wire $\mathtt{w}$.

Consider now the test performed by Alice at Step 7. Let A and B be the Clifford operators corresponding to keys $K_{\mathtt{w},a}$ and $K_{\mathtt{w},b}$ respectively. Notice that Alice can easily test the authenticity of a swaddling when she holds the outermost authentication. She simply applies $A^\dagger$ upon the $2n + 1$ wires, measures her n dummy wires in the computational basis to verify that they are in state $|0^n\rangle$. She then re-applies A to the $2n + 1$ wires of the swaddling. If Alice holds the innermost authentication of the swaddling,

the testing procedure relies on the ideal functionality $\mathsf{id}_{\text{TEST}}$ defined as:

$$\mathsf{id}_{\text{TEST}}((i_a, i_b), (i'_a, i'_b)) = \begin{cases} ((t, K'_{\mathtt{w},a}), K'_{\mathtt{w},b}) & \text{if } i_a = i'_a = id_A(\mathtt{w}) \wedge i_b = i'_b = id_B(\mathtt{w}), \\ (\perp, \perp) & \text{otherwise,} \end{cases}$$

where t, $K'_{\mathtt{w},a}$, and $K'_{\mathtt{w},b}$ correspond to random Clifford operations T, A', and B' respectively, subject to:

1. $\mathsf{id}_{\text{OPEN}}(id_A(\mathtt{w}), id_A(\mathtt{w})) = (\perp, K'_{\mathtt{w},a})$ and $\mathsf{id}_{\text{OPEN}}(id_B(\mathtt{w}), id_B(\mathtt{w})) = (K'_{\mathtt{w},b}, \perp)$,
2. If Alice holds the outermost authentication key of $\mathtt{w}$ then $T = (\mathbb{1}_n \otimes B'B^\dagger)A^\dagger$, and
3. If Alice holds the innermost authentication key of $\mathtt{w}$ then $T = (\mathbb{1}_n \otimes A'A^\dagger)B^\dagger$.

$\mathtt{Swaddle}(\mathcal{W})$, with $\mathcal{W} \subseteq \mathcal{A}$:

1. Alice initializes $n \cdot \#\mathcal{W}$ dummy wires in state $|0\rangle$.
2. For each $\mathtt{w} \in \mathcal{W}$, Alice randomly chooses Clifford $K_{\mathtt{w},a}$ on $n+1$ qubits, commits to it in $\mathsf{id}_{\text{SC}}((id_A(\mathtt{w}), K_{\mathtt{w},a}), \perp)$, and applies it to $\mathtt{w}$ and her dummies.
3. Alice sends these $\#\mathcal{W} \cdot (n+1)$ wires to Bob.
4. Bob initializes $\#\mathcal{W} \cdot n$ dummy wires in the state $|0\rangle$.
5. For each $\mathtt{w} \in \mathcal{W}$, Bob randomly chooses a Clifford $K_{\mathtt{w},b}$ on $2n+1$ qubits, commits to it in $\mathsf{id}_{\text{SC}}(\perp, (id_B(\mathtt{w}), K_{\mathtt{w},b}))$, and applies it to the $n+1$ wires received from Alice as well as his own dummies.
6. Bob sends all $\#\mathcal{W} \cdot (2n+1)$ wires back to Alice. Let $\mathfrak{s}(\mathtt{w})$ denote the resulting swaddling of $\mathtt{w} \in \mathcal{W}$.
7. For each $\mathtt{w} \in \mathcal{W}$, Alice runs $\mathtt{TestSwaddling}(\mathfrak{s}(\mathtt{w}))$.

Condition 1 ensures that the authentication keys with identifiers $id_A(\mathtt{w})$ and $id_B(\mathtt{w})$ have been updated to hold values $K'_{\mathtt{w},a}$ and $K'_{\mathtt{w},b}$ respectively. Conditions 2 and 3 make sure that the state $|\sigma\rangle$ of a valid swaddling $\mathfrak{s}(\mathtt{w})$ satisfies $T|\sigma\rangle = |0^n\rangle \otimes B'(|0^n\rangle \otimes |\varphi\rangle)$, where $|\varphi\rangle$ is the logical state of $\mathfrak{s}(\mathtt{w})$. Notice that Alice gets no information about Bob's Clifford B' if she had no information about B to start with.

$\mathtt{TestSwaddling}(\mathfrak{s}(\mathtt{w}))$, with Alice doing the testing:

1. Alice and Bob call the ideal functionality $((t, K'_{\mathtt{w},a}), K'_{\mathtt{w},b}) = \mathsf{id}_{\text{TEST}}((id_A(\mathtt{w}), id_B(\mathtt{w})), (id_A(\mathtt{w}), id_B(\mathtt{w})))$,
2. If one party gets $\perp$ in $\mathsf{id}_{\text{TEST}}$ then **abort**.
3. Alice applies T on $\mathfrak{s}(\mathtt{w})$ where T is the Clifford operator corresponding to string t.
4. Alice tests that her dummies are together in state $|0^n\rangle$. If not then she **aborts**.
5. Alice applies A' to the the $2n+1$ qubits of the swaddling (note that A' and B' are committed upon with identifiers $id_A(\mathtt{w})$ and $id_B(\mathtt{w})$ respectively).

Notice also that Bob can do the testing by the exact same procedure provided the roles of Alice and Bob are reversed in $\mathtt{TestSwaddling}(\mathtt{w})$. The security of these procedures will be discussed in Sect. 5.5. Intuitively, we expect that $\mathtt{TestSwaddling}(\mathtt{w})$ allows parties to test that a swaddling $\mathfrak{s}(\mathtt{w})$ has not been tampered with. It will also be used to test that each party transforms a swaddling $\mathfrak{s}(\mathtt{w})$ the way they should during the execution of the protocol. Notice that no information about the logical state of wire $\mathtt{w}$ leaks to any party in $\mathtt{Swaddle}(\mathtt{w})$ and $\mathtt{TestSwaddling}(\mathfrak{s}(\mathtt{w}))$ since statistically secure authentication must also encrypt $\mathtt{w}$ [2].

At the end of the our protocol, each party will be asked to verify that the other party's registers upon which the circuit is evaluated are all in the states they should be.

It simply consists in the execution of $\texttt{TestSwaddling}(\mathfrak{s}(\mathtt{w}))$ for all wires held by each party. The full description can be found in [5].

The following sub-protocol implements the obvious way the openings of all committed Clifford operators, thereby allowing Alice and Bob to get the final state of the computation.

OpenAllSwaddlings:
1. For each wire $\mathtt{w} \in \mathcal{A}_{\mathrm{out}}$, Bob reveals to Alice his committed secret key encrypting $\mathtt{w}$, with identifier $id_B(\mathtt{w})$, by calling $\mathsf{id}_{\text{OPEN}}(id_B(\mathtt{w}), id_B(\mathtt{w}))$.
2. For each wire $\mathtt{w} \in \mathcal{B}_{\mathrm{out}}$, Alice reveals to Bob her committed secret key encrypting $\mathtt{w}$, with identifier $id_A(\mathtt{w})$, by calling $\mathsf{id}_{\text{OPEN}}(id_A(\mathtt{w}), id_A(\mathtt{w}))$.

Notice that since Alice learns Bob's secret keys before she unveils them, our protocol will lack fairness for Bob.

5 Description of the protocol

We first start by defining the various spaces on which (the honest) Alice and Bob will be working. The circuit that they want to execute acts on some wires in Alice's possession and some in Bob's possession, and these will be swaddled as described above. We will denote by $\mathcal{A}_u$, $\mathcal{B}_u$ the spaces corresponding to Alice and Bob's unswaddled wires, reserving $\mathcal{A}$ and $\mathcal{B}$ for the actual swaddled wires.

We first initialize all the qubits by swaddling them, we then perform each gate from the circuit one after the other on the authenticated data. Hence, we need to give subprotocols for the initialization as well as for each of the gates in our universal set.

For all of the Clifford gates (i.e., all gates in $\mathcal{UG}$ except for R), the subprotocols are fairly simple: we use classical two-party computation to reveal a Clifford operation that executes the gate while updating the encryption key; the revealed Clifford then looks uniformly distributed and independent of everything else.

Implementing the R-gate is more involved. We use ideas from fault-tolerant computation, where this gate is implemented by doing gate teleportation via a so-called *magic state*: one prepares a special state (namely $|M\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{i\pi/4}|1\rangle)$) and then use a teleportation-like circuit, which itself requires only Clifford gates and measurements, to execute the gate. The problem is then reduced to that of producing this magic state, which can be done by a distillation process. The distillation process that we use is exactly the one considered in [3] (where $|M\rangle$ is an "H-type magic state" in their language); a description of it can also be found in [5].

5.1 Main Protocol

We now give the full description of our protocol, denoted $\widehat{\Pi}^{\mathscr{O}'}_{\mathscr{F}} = (\mathscr{A}, \mathscr{B}, \mathscr{O}', m)$, allowing to evaluate the CPTP map $\mathscr{F} : \mathrm{L}(\mathcal{A}_{\mathrm{in}} \otimes \mathcal{B}_{\mathrm{in}}) \mapsto \mathrm{L}(\mathcal{A}_{\mathrm{out}} \otimes \mathcal{B}_{\mathrm{out}})$ upon joint input state $\rho_{\mathrm{in}} \in \mathrm{D}(\mathcal{R} \otimes \mathcal{A}_{\mathrm{in}} \otimes \mathcal{B}_{\mathrm{in}})$. The oracle list needed to run the protocol in the hybrid model is provided implicitly in Sect. 5. The operations performed at each step by $\mathscr{A}$ and $\mathscr{B}$ are described informally as the instructions of Alice and Bob respectively. We will view $\mathscr{F}$ as being implemented by a quantum circuit acting on $\mathcal{A}_{\mathrm{in}} \otimes \mathcal{B}_{\mathrm{in}}$ together

with ancillas $\mathcal{A}_a$ and $\mathcal{B}_a$ initialized in state $|0\rangle$ (we shall explain below how to test that ancillas are really in state $|0\rangle$). At the end of the circuit, some of these wires become part of the outputs $\mathcal{A}_{\rm out}$ and $\mathcal{B}_{\rm out}$, and the rest are part of the environment that will remain encrypted ($\mathcal{A}_e$ and $\mathcal{B}_e$). Let $G_1, G_2, \ldots, G_{\ell(n)}$ be an enumeration of all gates of the circuit for $\mathscr{F}$ where $\ell(n)$ is polynomial in n, and for $i < j$, G_i is executed before G_j.

Protocol $\widehat{\Pi}^{\mathscr{O}'}_{\mathscr{F}}$ for the evaluation of $\mathscr{F}$ upon joint input $\rho_{\rm in} \in {\rm D}(\mathcal{R} \otimes \mathcal{A}_{\rm in} \otimes \mathcal{B}_{\rm in})$:

1. Alice and Bob **run** `Initialization`,
2. **For** $i = 1 \ldots \ell(n)$:
   - **If** G_i is a one-bit Clifford gate applied to wire `w` **then** Alice and Bob **run** `OneQubitClifford`$(G_i, \mathfrak{s}(\mathtt{w}))$,
   - **If** G_i is a CNOT-gate applied to control wire $\mathtt{w}_c$ and $\mathtt{w}_t$ **then** Alice and Bob **run** `CNOT`$(\mathfrak{s}(\mathtt{w}_c), \mathfrak{s}(\mathtt{w}_t))$,
   - **If** G_i is an R-gate applied to wire `w` **then** Alice and Bob **run** `RGate`$(\mathfrak{s}(\mathtt{w}))$,
3. Alice and Bob **run** `TestAllSwaddlings`,
4. Alice and Bob **run** `OpenAllSwaddlings` ,
5. Alice and Bob decrypt the swaddlings for all wires in $\mathcal{A}_{\rm out}$ and $\mathcal{B}_{\rm out}$ using the keys received by the other party together with their own.

This protocol calls a number of sub-protocols, which are partially described next and in full details in [5].

5.2 Subprotocols

The `Initialization` subprotocol prepares all swaddlings required during the evaluation of the circuit. In addition to swaddling all wires holding the qubits upon which the circuit acts, some ancillary states also have to be swaddled: one wire in the magic state and one wire in state $|0\rangle$ per R-gate in the circuit.

`Initialization`**:**

1. **For** each R-gate being applied to a wire in $\mathcal{A}$, Alice adds one additional wire $\mathtt{m}_{a,i}$ initialized in state $|M\rangle$ to $\mathcal{A}$.
2. **For** each R-gate being applied to a wire in $\mathcal{B}$, Bob adds an additional wire $\mathtt{m}_{b,i}$ initialized in state $|M\rangle$ to $\mathcal{B}$.
3. **For** each R-gate being applied to a wire in $\mathcal{B}$, Alice adds an additional wire $\mathtt{c}_{a,i}$ initialized in state $|0\rangle$ to $\mathcal{A}$.
4. **For** each R-gate being applied to a wire in $\mathcal{A}$, Bob adds an additional wire $\mathtt{c}_{b,i}$ initialized in state $|0\rangle$ to $\mathcal{B}$.
5. Alice **calls** `Swaddle`$(\mathcal{A})$.
6. Bob **calls** `Swaddle`$(\mathcal{B})$.
7. **For** every wire $\mathtt{a} \in \mathcal{A}_a$ or added in step 3 above, Alice **runs** `VerifyAncilla`$(\mathfrak{s}(\mathtt{a}))$.
8. **For** every wire $\mathtt{a} \in \mathcal{B}_a$ or added in step 4 above, Bob **runs** `VerifyAncilla`$(\mathfrak{s}(\mathtt{a}))$.
9. **For** each magic wire $\mathtt{m} \in \mathcal{A}$, Alice **calls** `DistillMagic`$(\mathfrak{s}(\mathtt{m}))$.
10. **For** each magic wire $\mathtt{m} \in \mathcal{B}$, Bob **calls** `DistillMagic`$(\mathfrak{s}(\mathtt{m}))$.

Sub-protocols `VerifyAncilla`$(\mathfrak{s}(\mathtt{a}))$ and `DistillMagic`$(\mathfrak{s}(\mathtt{m}))$ are described in [5]. The following sub-protocol evaluates any one-qubit Clifford:

$\texttt{OneQubitClifford}(\texttt{C}, \mathfrak{s}(\texttt{w}))$ **with** $\texttt{w} \in \mathcal{A}$**:**
1. Alice and Bob perform a TPC whose outcome tells Alice a randomly chosen Clifford gate which performs the gate, and changes the QAS key.
2. Alice performs the gate on $\mathfrak{s}(\texttt{w})$.

The idea behind the sub-protocol is to use TPC to compute the gate *through* the authentication. Of course, the TPC needed depends upon the gate. The CNOT-gate, the only two-qubit gate of our universal set, is done as follows:

$\texttt{CNOT}(\mathfrak{s}(\texttt{w}_\texttt{c}), \mathfrak{s}(\texttt{w}_\texttt{t}))$ **with wire** $\texttt{w}_c \in \mathcal{A}$ **as control and** $\texttt{w}_t$ **as target:**
1. Bob sends $\mathfrak{s}(\texttt{w}_t)$ to Alice if Bob holds it, in which case Alice runs $\texttt{TestDummies}(\mathfrak{s}(\texttt{w}_\texttt{t}))$.
2. Alice performs a randomly selected Clifford C on $4n + 2$ qubits jointly on $\mathfrak{s}(\texttt{w}_c)$ and $\mathfrak{s}(\texttt{w}_t)$.
3. Alice sends $\mathfrak{s}(\texttt{w}_c)$ and $\mathfrak{s}(\texttt{w}_t)$ to Bob; Bob runs TestDummies jointly on them.
4. Alice and Bob perform a TPC whose outcome tells Bob to perform a Clifford unitary $C' = (K'_c \otimes K'_t)(\mathsf{CNOT})K^\dagger$ where K is the key of the swaddling at this point, and K'_c and K'_t are randomly-chosen Cliffords that become the new key.
5. Bob sends $\mathfrak{s}(\texttt{w}_c)$ (and $\mathfrak{s}(\texttt{w}_t)$ if she held this one too) to Alice. Alice runs TestDummies on them.

The idea in this subprotocol is to take the two swaddled qubits involved and turn them into one big swaddling. Then, performing a CNOT on them is no different from performing any other Clifford gate on the whole block. We then separate the big swaddling back into its components. The only gate that remains is the R-gate.

$\texttt{RGate}(\mathfrak{s}(\texttt{w}))$ **with** $\texttt{w} \in \mathcal{A}$**:**
1. Alice performs a swaddled CNOT with $\texttt{m}_w$ as a control, and w as target.
2. Alice sends $\mathfrak{s}(\texttt{w})$ to Bob. Bob runs $\texttt{TestSwaddling}(\mathfrak{s}(\texttt{w}))$ on it.
3. Alice runs $\texttt{Measure}(\mathfrak{s}(\texttt{w}))$.
4. Alice and Bob perform a TPC whose result is a Clifford which, if both measurement results were zero, updates the key, and if both measurement results were one, performs a swaddled $e^{i\pi/4}XP^\dagger$ and then updates the key. If the measurement results differ, then they abort.
5. Alice relabels $\texttt{m}_{a,i}$ to w.

This subprotocol performs the R-gate using gate teleportation[9,8] via the magic state is very similar to the fault-tolerant version of the R-gate introduced in [7,14]. To perform an R-gate on a wire w via gate teleportation, we would first perform a CNOT from the magic state to w, and then measure w in the computational basis. If the answer is 0, we do nothing, and if it is 1, we perform $e^{i\pi/4}XP^\dagger$ on the former magic state, that we then rename w. $\texttt{Measure}(\mathfrak{s}(\texttt{w}))$ is described in [5].

5.3 Security of the subprotocols

We will now show that the protocol described in the previous section has the following property: any adversary which deviates significantly from the protocol will be caught cheating with high probability. The general strategy will be as follows. For the initialization, any adversary will be forced to input something into the protocol, and will end up

with some state that is properly swaddled. Then, for every other protocol step, we will assume that at the beginning, the inputs are properly swaddled, and will aim to show that after the protocol step is done, we are once again left with a correct swaddling of the data, to which the correct operation has been applied. Furthermore, at every step, any deviation from the protocol will be essentially equivalent to an attack on the authentication scheme, which means that an adversary's chances of succeeding in changing the state without getting caught will be exponentially small in the number of dummy wires per data qubit.

5.4 Some additional definitions

Before we start, we will find it convenient to introduce some additional notation. From now on, ρ_0 will consist of ρ_{in} augmented with all additional qubits introduced in the `Initialization` phase: the additional ancillas, and the dummy wires Alice and Bob use for swaddling. Furthermore, we will denote by $\mathcal{K}_a$ and $\mathcal{K}_b$ systems which represent Alice's and Bob's current key, respectively. These should be thought of as being part of the inner state of the TPC ideal functionality, and therefore cannot be changed at will. $\mathcal{A}$ represents all other systems at Alice, $\mathcal{B}$ represents all systems in Bob's possession, and $\mathcal{R}$ is a system that includes everything else and ensures that the total state is pure.

Furthermore, in the sequel, we will call a *step* an execution of any of the subprotocols listed above; each of these steps consists of multiple *turns* in the sense of Section 2.3: the state at turn i consists of the state before the ith use of an oracle in the protocol (either a communication oracle or a classical computation oracle).

We will denote by $\mathsf{C}_{a,s}$ the operation that encodes Alice's qubits according to the keys stored in the TPC ideal functionalities at turn s in the protocol, and likewise for Bob's encoding operation $\mathsf{C}_{b,s}$. See [5] for more precise definitions. Furthermore, we will denote by $[\tilde{\mathscr{A}} \circledast \mathscr{B} \wedge \overline{\text{E-ABORT}}]_s^{\mathscr{O}'} \left(\rho_{\text{in}}^{\mathcal{A}_{\text{in}}\mathcal{B}_{\text{in}}\mathcal{R}}\right)$ the global state of the protocol at turn s conditioned on the fact that the protocol doesn't abort before `TestAllSwaddlings` is completed.

Louis: At the very end or at the end of the evaluation phase, that is just after testallswaddlings?

Fred: Just after testallswaddlings. The thing that would intuitively make more sense would be to say "doesn't abort up to the current point", but in some proofs we need to know that it won't abort in the future either...

Note that this state is not normalized; its trace corresponds to the probability of not aborting before the end.

Definition 5.1 (Forcing). *We will say that the protocol is $\mathscr{G}^{\mathcal{A}\mathcal{B}}$-*forcing *for $\tilde{\mathscr{A}}$ at turn s with initial operation $\tilde{\mathscr{E}}_0^{\mathcal{A}}$ if there exists a final operation $\tilde{\mathscr{E}}^{\mathcal{A}}$ such that:*

$$\Delta\left([\tilde{\mathscr{A}} \circledast \mathscr{B} \wedge \overline{\text{E-ABORT}}]_s^{\mathscr{O}'}\left(\rho_{\text{in}}^{\mathcal{A}_{\text{in}}\mathcal{B}_{\text{in}}\mathcal{R}}\right), \tilde{\mathscr{E}} \circ \mathsf{C}_{b,s} \circ \mathscr{G} \circ \tilde{\mathscr{E}}_0\left(\rho_{\text{in}}^{\mathcal{A}_{\text{in}}\mathcal{B}_{\text{in}}\mathcal{R}}\right)\right) \leqslant \mathsf{negl}(n) \ ,$$

where $\tilde{\mathscr{E}}_0$ is a completely positive, trace non-increasing map that acts only on qubits in Alice's possession at the beginning of the protocol: her own input qubits, the dummies she inputs in the swaddling, and the various ancillas that she adds.

In other words, if the protocol is $\mathscr{G}$-forcing for Alice at some turn s, then, if the protocol doesn't abort early, regardless of what she tries to do, it will be essentially equivalent to changing her input (using the initial operation $\tilde{\mathscr{E}}_0$), executing the operation $\mathscr{G}$ (which will turn out to be the circuit that is supposed to be executed up to turn s), swaddling the result, and then doing an arbitrary operation on her share alone, represented by $\tilde{\mathscr{E}}$.

Definition 5.2. *We say that a subprotocol is $\mathscr{G}$-forcing if the protocol is $\mathscr{G} \circ \mathscr{G}_0$-forcing at the end of the subprotocol given that it was $\mathscr{G}_0$-forcing at the beginning.*

Definition 5.3 (Hiding). *We will say that the protocol is* hiding *for $\tilde{\mathscr{A}}$ at turn s, if, for all input states $\rho_{\text{in}}^{\mathcal{A}_{\text{in}}\mathcal{B}_{\text{in}}\mathcal{R}_{\text{in}}}$, we have that*

$$\Delta\left(\text{tr}_{\mathcal{B}}\left[[\tilde{\mathscr{A}} \circledast \mathscr{B}]_s^{\mathscr{O}'}(\rho_{\text{in}}^{\mathcal{A}_{\text{in}}\mathcal{B}_{\text{in}}\mathcal{R}_{\text{in}}})\right], \text{tr}_{\mathcal{B}}\left[[\tilde{\mathscr{A}} \circledast \mathscr{B}]_s^{\mathscr{O}'}(\rho_{\text{in}}^{\mathcal{A}_{\text{in}}\mathcal{R}_{\text{in}}} \otimes |0\rangle\langle 0|^{\mathcal{B}_{\text{in}}})\right]\right) \leqslant \mathsf{negl}(n) \ .$$

Hence, the protocol is hiding if the state seen by a dishonest Alice is independent of Bob's input. Note that this in particular implies that $\tilde{\mathscr{A}}$'s action at turn s is necessarily independent of Bob's input.

5.5 Proving hidingness and forcingness

To construct the simulator needed to prove security, we will need to show two things. First, we will have to show that before the keys are revealed, the cheater's internal state can be produced by running the protocol internally with a dummy input from the honest party; this will follow from the fact that the Clifford QAS is a secure encryption scheme. Second, we will have to show that after the keys are revealed, the correct circuit has been applied. These two properties correspond to the "hiding" and "forcing" properties defined in the previous section, and proving these for our protocol will be the focus of this section. The fact that the protocol is hiding simply follows from the security of the Clifford QAS as an encryption scheme; we state this as a lemma below. To prove forcingness, the usual trick will be to assume that we pass every call to `TestDummies` in the protocol (since we only need to look at the no-early-abort case) and use the security definition of the Clifford QAS (Definition 2.2) to show that the dishonest party's attack can be represented by a completely positive, trace non-increasing map $\mathscr{U}^{\text{acc}}$ that acts only on his/her other systems (i.e. the ones that were not involved in the `TestDummies`). If the adversary decides to try to break the QAS at this step, this $\mathscr{U}^{\text{acc}}$ will simply decrease the trace to reflect the probability of abortion. We prove the forcingness of the various subprotocols in [5], and simply summarize the end result as Lemma 5.5 below.

Lemma 5.4. *For every turn before* `OpenAllSwaddlings`*, the protocol is hiding for every adversary $\tilde{\mathscr{A}}$ and $\tilde{\mathscr{B}}$.*

Proof. During this phase of the protocol, all $\tilde{\mathscr{A}}$ ever gets from Bob (and $\tilde{\mathscr{B}}$ from Alice) is encrypted in a QAS, whose key is managed by the classical two-party computations. Hence, this follows directly from the security of the Clifford QAS (see Section 2.2, or the full version [5] for more details).

Lemma 5.5. *All subprotocols are $\mathscr{G}$-forcing for any $\tilde{\mathscr{A}}$ and $\tilde{\mathscr{B}}$, where $\mathscr{G}$ is the operation performed by the subprotocol.*

6 Proving Active Security

Here is the rough description of $\tilde{\mathscr{C}}$'s operations for the start of the simulation. These steps allow to simulate any execution aborting in the real world before the adversary $\tilde{\mathscr{A}}$ receives back all her swaddlings from $\mathscr{B}$ in `TestAllSwaddlings`. Let s^* be the turn in $\widehat{\Pi}^{\mathscr{O}'}_{\mathscr{F}}$ at which this transmission is received by $\tilde{\mathscr{A}}$. The simulator $\tilde{\mathscr{C}}$ runs $\tilde{\mathscr{A}}$ as a subroutine using an internal copy of $\mathscr{B}$'s instructions run upon a (or any) dummy input state $|0\rangle^{\mathcal{B}_{\rm in}}$. In the following, we denote by $\mathscr{B}^*$ the simulated $\mathscr{B}$ on a dummy input run internally in $\tilde{\mathscr{C}}$. In order to distinguish the registers of the real $\mathscr{B}$ from the ones of $\mathscr{B}^*$, we denote by $\mathcal{B}^*$ a register $\mathcal{B}$ of $\mathscr{B}$ used by $\mathscr{B}^*$.

We define E-ABORT_s as the event consisting in an execution between two parties aborting at turn $s < s^*$. Let $[\tilde{\mathscr{C}} \circledast \mathscr{D} \wedge \text{E-ABORT}_s]^{\mathscr{F}}(\rho_{\rm in})$ and $[\tilde{\mathscr{A}} \circledast \mathscr{B} \wedge \text{E-ABORT}_s]^{\mathscr{O}'}(\rho_{\rm in})$ be denoting the the joint state of an execution that aborts at turn s upon joint input state $\rho_{\rm in}$ between $\tilde{\mathscr{C}}$ and $\mathscr{D}$ (in the ideal world) and between $\tilde{\mathscr{A}}$ and $\mathscr{B}$ (in the real world) respectively. States $[\tilde{\mathscr{C}} \circledast \mathscr{D} \wedge \text{E-ABORT}_s]^{\mathscr{F}}(\rho_{\rm in})$ and $[\tilde{\mathscr{A}} \circledast \mathscr{B} \wedge \text{E-ABORT}_s]^{\mathscr{O}'}(\rho_{\rm in})$ are not normalized, $\mathrm{tr}([\tilde{\mathscr{C}} \circledast \mathscr{D} \wedge \text{E-ABORT}_s]^{\mathscr{F}}(\rho_{\rm in}))$ and $\mathrm{tr}([\tilde{\mathscr{A}} \circledast \mathscr{B} \wedge \text{E-ABORT}_s]^{\mathscr{O}'}(\rho_{\rm in}))$ are the probabilities that $[\tilde{\mathscr{C}} \circledast \mathscr{D}]^{\mathscr{F}}(\rho_{\rm in})$ and $[\tilde{\mathscr{A}} \circledast \mathscr{B}]^{\mathscr{O}'}(\rho_{\rm in})$ aborts at step s respectively. The proof of next lemma can be found in [5] and is easy since all wires remain encrypted throughout.

Lemma 6.1 (Early abort simulation). *Let $\tilde{\mathscr{A}}$ be an adversary in hybrid protocol $\widehat{\Pi}^{\mathscr{O}'}_{\mathscr{F}} = (\mathscr{A}, \mathscr{B}, \mathscr{O}', m)$ for $\mathscr{F} : \mathrm{L}(\mathcal{A}_{\rm in} \otimes \mathcal{B}_{\rm in}) \to \mathrm{L}(\mathcal{A}_{\rm out} \otimes \mathcal{B}_{\rm out})$. Let s^* be the turn in $\widehat{\Pi}^{\mathscr{O}'}_{\mathscr{F}}$ at which $\mathscr{B}$ returns all of $\tilde{\mathscr{A}}$'s swaddlings in* `TestAllSwaddlings` *and let $0 \leq s < s^*$. Then, there exists an adversary $\tilde{\mathscr{C}}$ in $\Gamma_{\mathscr{F}}$(polysize in the size of $\tilde{\mathscr{A}}$) such that for any $\rho_{\rm in} \in \mathrm{D}(\mathcal{A}_{\rm in} \otimes \mathcal{B}_{\rm in} \otimes \mathcal{R})$,*

$$\Delta\left([\tilde{\mathscr{C}} \circledast \mathscr{D} \wedge \text{E-ABORT}_s]^{\mathscr{F}}(\rho_{\rm in}), [\tilde{\mathscr{A}} \circledast \mathscr{B} \wedge \text{E-ABORT}_s]^{\mathscr{O}'}(\rho_{\rm in})\right) \leq \mathsf{negl}(n) \ .$$

By symmetry, the same is also true with respect to adversaries $\tilde{\mathscr{B}}$ in $\widehat{\Pi}^{\mathscr{O}'}_{\mathscr{F}}$ and $\tilde{\mathscr{D}}$ in $\Gamma_{\mathscr{F}}$.

It remains to simulate the execution from turn s^* until the end. In the simulated world, if $\tilde{\mathscr{C}}$ reaches turn s^* when $\tilde{\mathscr{A}}$ and $\mathscr{B}^*$ are interacting then the output state $\mathscr{F}(\rho_{\rm in}^{\mathcal{A}_{\rm in}\mathcal{R}} \otimes |0\rangle\langle 0|)$ can be recovered. The reason being that at turn s^*, all swaddlings have been tested by $\mathscr{B}^*$ in `TestAllSwaddlings`. $\tilde{\mathscr{C}}$ can get the output state since it knows all keys allowing to decrypt all logical wires, and all these wires have not been tampered with. The simulation then works along the same lines than in [4]. At turn s^*, $\tilde{\mathscr{C}}$ is simulating $\mathscr{B}^*$'s quantum transmission of all swaddlings belonging to $\tilde{\mathscr{A}}$ back to $\tilde{\mathscr{A}}$. These swaddlings have been successfully tested by $\mathscr{B}^*$ in `TestAllSwaddlings`. $\tilde{\mathscr{C}}$ intercepts all these swaddlings and decrypts them together with all swaddlings held by $\mathscr{B}^*$. $\mathscr{C}$ then recovers the output state. $\mathscr{C}$ undoes the quantum operation $\mathscr{F}$ in order to recover $\tilde{\mathscr{A}}$'s effective input state before querying $\mathscr{F}_1$ with the effective input state. $\tilde{\mathscr{C}}$ swaddles back the answer to the query using the same keys. The swaddlings are finally sent to $\tilde{\mathscr{A}}$ before resuming the interaction between $\tilde{\mathscr{A}}$ and $\mathscr{B}^*$. Two things can happen in `OpenAllSwaddlings`: 1) $\tilde{\mathscr{A}}$'s behavior makes the execution **abort**, and 2) $\tilde{\mathscr{A}}$ and $\mathscr{B}$ reach the end of the execution in normal conditions. In the first case, $\tilde{\mathscr{A}}$ may even get the output state of the computation while preventing $\mathscr{B}$ from recovering his own.

When the output state is available to $\tilde{\mathscr{A}}$, $\tilde{\mathscr{C}}$ will need to call $\mathscr{F}_1$ (with $\tilde{\mathscr{A}}$'s effective input) and $\mathscr{F}_2$ where the later call is without fairness (i.e., $f = 0$) when $\tilde{\mathscr{A}}$ prevents $\mathscr{B}$ from decrypting his output logical wires.

Let $\overline{\text{E-ABORT}}$ be the event of not having E-ABORT_s at any turn $s < s^*$. Next lemma establishes the active security when no early aborting occurs. The proof can be found in [5].

Lemma 6.2 (No early abort simulation). *For any quantum adversary $\tilde{\mathscr{A}}$ in hybrid protocol $\widehat{\Pi}^{\mathscr{O}'}_{\mathscr{F}} = (\mathscr{A}, \mathscr{B}, \mathscr{O}', m)$ for $\mathscr{F} : \mathrm{L}(\mathcal{A}_{\mathrm{in}} \otimes \mathcal{B}_{\mathrm{in}}) \to \mathrm{L}(\mathcal{A}_{\mathrm{out}} \otimes \mathcal{B}_{\mathrm{out}})$, there exists an adversary $\tilde{\mathscr{C}}$ in $\Gamma_{\mathscr{F}}$ (polysize in the size of $\tilde{\mathscr{A}}$ and $\mathscr{B}$) such that for any $\rho_{\mathrm{in}} \in \mathrm{D}(\mathcal{A}_{\mathrm{in}} \otimes \mathcal{B}_{\mathrm{in}} \otimes \mathcal{R})$,*

$$\Delta\left([\tilde{\mathscr{C}} \circledast \mathscr{D} \wedge \overline{\text{E-ABORT}}]^{\mathscr{F}}(\rho_{\mathrm{in}}), [\tilde{\mathscr{A}} \circledast \mathscr{B} \wedge \overline{\text{E-ABORT}}]^{\mathscr{O}'}(\rho_{\mathrm{in}})\right) \leq \mathsf{negl}(n) \ .$$

By symmetry, the same is also true with respect to adversaries $\tilde{\mathscr{B}}$ in $\widehat{\Pi}^{\mathscr{O}'}_{\mathscr{F}}$ and $\tilde{\mathscr{D}}$ in $\Gamma_{\mathscr{F}}$.

Lemmas 6.1 and 6.2 allow to conclude the active security of the protocol:

Theorem 6.3 (Active security). *The two-party hybrid protocol $\widehat{\Pi}^{\mathscr{O}'}_{\mathscr{F}}$ for any polynomial-time quantum operation $\mathscr{F} : \mathrm{L}(\mathcal{A}_{\mathrm{in}} \otimes \mathcal{B}_{\mathrm{in}}) \to \mathrm{L}(\mathcal{A}_{\mathrm{out}} \otimes \mathcal{B}_{\mathrm{out}})$ is statistically active secure without fairness for Bob.*

References

1. D. Aharonov, M. Ben-Or, and E. Eban. Interactive proofs for quantum computations. In *Proceedings of Innovations in Computer Science*, 2008. http://arxiv.org/abs/0810.5375.
2. H. Barnum, C. Crépeau, D. Gottesman, A. Smith, and A. Tapp. Authentication of quantum messages. In *43rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 449–458, 2002.
3. S. Bravyi and A. Kitaev. Universal quantum computation with ideal clifford gates and noisy ancillas. *Physical Review A*, 71(022316), 2005. quant-ph/0403025.
4. F. Dupuis, J. B. Nielsen, and L. Salvail. Secure two-party quantum evaluation of unitaries against specious adversaries. In T. Rabin, editor, *CRYPTO*, volume 6223 of *Lecture Notes in Computer Science*, pages 685–706. Springer, 2010.
5. F. Dupuis, J. B. Nielsen, and L. Salvail. Actively secure two-party evaluation of any quantum operation. Cryptology ePrint Archive, record 2012/XXX, http://eprint.iacr.org/, 2012.
6. D. Gottesman. *Stabilizer codes and quantum error correction*. PhD thesis, California Institute of Technology, 1997.
7. D. Gottesman. An introduction to quantum error correction and fault-tolerant quantum computation. In J. Samuel J. Lomonaco, editor, *Quantum Information Science and Its Contributions to Mathematics*, volume 68, pages 13–60. Proceedings of Symposia in Applied Mathematics, April 2010. http://arxiv.org/abs/0904.2557.
8. D. Gottesman and I. L. Chuang. Demonstrating the viability of universal quantum computation using teleportation and single-qubit operations. *Nature*, 402:390–393, November 1999.

9. D. Gottesman and I. L. Chuang. Quantum teleportation is a universal computational primitive. http://arxiv.org/abs/quant-ph/9908010, August 1999.
10. G. Gutoski and J. Watrous. Toward a general theory of quantum games. In *39th Annual ACM Symposium on Theory of Computing (STOC)*, pages 565–574, 2007.
11. S. Hallgren, A. Smith, and F. Song. Classical cryptographic protocols in a quantum world. In P. Rogaway, editor, *CRYPTO*, volume 6841 of *Lecture Notes in Computer Science*, pages 411–428. Springer, 2011.
12. C. Lunemann and J. B. Nielsen. Fully simulatable quantum-secure coin-flipping and applications. In A. Nitaj and D. Pointcheval, editors, *AFRICACRYPT*, volume 6737 of *Lecture Notes in Computer Science*, pages 21–40. Springer, 2011.
13. O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In *37th Annual ACM Symposium on Theory of Computing (STOC)*, pages 84–93, 2005.
14. P. W. Shor. Fault-tolerant quantum computation. In *37th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 56–65, 1996.