# 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 [DNS10].

## 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 $\rho_{\mathrm{in}}$ 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.

A simple example of such an $\mathscr{F}$ is the quantum swap, where $\mathcal{A}_{\mathrm{in}} = \mathcal{A}_{\mathrm{out}} = \mathcal{A}$, $\mathcal{B}_{\mathrm{in}} = \mathcal{B}_{\mathrm{out}} = \mathcal{B}$, and $\rho_{\mathrm{out}}^{\mathcal{A}} = \rho_{\mathrm{in}}^{\mathcal{B}}$ and $\rho_{\mathrm{out}}^{\mathcal{B}} = \rho_{\mathrm{in}}^{\mathcal{A}}$. Securely implementing this unitary basically means to do an atomic swap of quantum states, which was shown impossible in [DNS10] 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*

⋆ 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, 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(FQRNT).

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 [DNS10], but it is easy to see that security in this model implies security in the stand-alone model of [HSS11], as long as the simulator is poly-time. The stand-alone model of [HSS11] 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 [Reg05,HSS11] and under the more general assumption that mixed commitment schemes exist [LN11], 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 [DNS10], 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 [DNS10], 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 the universal set of gates $\mathcal{UG}$ consisting of gates $\mathsf{X}$, $\mathsf{Y}$, $\mathsf{Z}$, $\mathsf{CNOT}$, $\mathsf{H} = \frac{1}{\sqrt{2}}\left(\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix}\right)$, $\mathsf{P} = \left(\begin{smallmatrix} 1 & 0 \\ 0 & i \end{smallmatrix}\right)$, 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 $\mathsf{X}$, $\mathsf{Y}$ and $\mathsf{Z}$ is easy, as they commute or anti-commute with the encryption operators. As an example, assume that the parties want to apply an $\mathsf{X}$ 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 $\mathsf{CNOT}$, $\mathsf{H}$ and $\mathsf{P}$ other "commutation" rules with $\mathsf{X}$ and $\mathsf{Z}$ are used. For $\mathsf{H}$, it is used that $\mathsf{HX} = \mathsf{ZH}$ and $\mathsf{HZ} = \mathsf{XH}$, so $\mathsf{H}$ 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 $\mathsf{R}$, 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 $\mathsf{P}^x$. Getting rid of this requires quantum communication 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 [DNS10] 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 [DNS10] 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 [DNS10] 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 [DNS10] 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 [DNS10], 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 [DNS10].

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 [ABE08]. It authenticates a quantum message using a random unitary implementable using gates $\mathsf{X}$, $\mathsf{Y}$, $\mathsf{Z}$, $\mathsf{CNOT}$, $\mathsf{H}$ and $\mathsf{P}$. The authentication works as follows. In order to satisfy the second property, take a qubit $|v\rangle$ on some wire. Alice prepends $n$ new wires, in the all-zero state $|0^n\rangle$. Then she applies a uniformly random $(n+1)$-bit Clifford operator $A$ to the $n+1$ wires. She then sends the state to Bob, who appends $n$ more wires in the all-zero state and applies a uniformly random $(2n+1)$-bit Clifford operator $B$ to the $2n+1$ wires. We can write this as $|V\rangle = B(|0^n\rangle \otimes A(|v\rangle|0^n\rangle))$. The key is $(A, B)$. This authentication can be checked in two different ways. Either, apply $B^\dagger$ to the authenticated state and check that the first $n$ wires are all zero. Or, apply $B^\dagger$ and then apply $A^\dagger$ to the last $n+1$ wires and check that the last $n$ 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 [DNS10] will allow to fairly easily apply Clifford gates "through the authentication". Since the Clifford unitaries form a group, the operation consisting of decrypting a qubit, applying a Clifford gate to it and reauthenticating it using a different key is also Clifford unitary. Hence, we can apply a Clifford gate to a swaddled qubit simply by changing the authentication keys in the appropriate manner. More precisely, to execute the Clifford gate $G \in \mathfrak{C}_1$, Alice updates her key to $A(G^\dagger \otimes \mathbb{1}_n)$, and Alice and Bob use a TPC to update Alice's commitment to her key. Executing $\mathsf{CNOT}$ gates is similar, but involves two swaddlings. The $\mathsf{R}$-gate requires new techniques reminiscent of the fault tolerant implementation of it [Sho96,Got97,Got10]. 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 $n$ 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 $B$ which effectively swaps the inner and outer authentications (see below for details). Alice then holds the outermost authentication and can easily test its integrity.

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

We refer to [DNS10] for a more complete introduction to the notation used throughout. In the following, various Hilbert spaces will be seen as collections of *wires*, represented in lower-case typewriter font (such as $\mathtt{w}, \mathtt{x}, \mathtt{y}$); each wire represents a qubit involved in a computation. We will denote by $\mathtt{w} \in \mathcal{A}$ the fact that $\mathtt{w}$ is part of the system $\mathcal{A}$. In the following, registers like $\mathcal{A}$ are considered as being composed by a set of wires or qubit registers. Henceforth, we shall denote a negligible function in $n$ by $\mathsf{negl}(n)$.

The set of positive semi-definite operators on $\mathcal{A}$ is denoted by $\mathrm{Pos}(\mathcal{A})$ and the one with trace 1 are denoted by $\mathrm{D}(\mathcal{A})$; $\mathrm{D}(\mathcal{A})$ is the set of all possible quantum states for register $\mathcal{A}$. An operator $A \in \mathrm{L}(\mathcal{A}, \mathcal{B})$ is called a *linear isometry* if $A^\dagger A = \mathbb{1}_\mathcal{A}$. The set of unitary operators (i.e., linear isometries with $\mathcal{B} = \mathcal{A}$) acting in $\mathcal{A}$ is denoted by $\mathrm{U}(\mathcal{A})$. The identity operator in $\mathcal{A}$ is denoted $\mathbb{1}_\mathcal{A}$ while the completely mixed state in $\mathrm{D}(\mathcal{A})$ is denoted by $\mathbb{I}_\mathcal{A}$. For any positive integer $N > 0$, $\mathbb{1}_N$ and $\mathbb{I}_N$ denote the identity operator respectively the completely mixed state in the $N$-dimensional Hilbert space $\mathcal{H}_N$. When the context requires, a pure state $|\psi\rangle \in \mathcal{A} \otimes \mathcal{B}$ will be written $|\psi\rangle^{\mathcal{AB}}$ to make explicit the registers in which it is stored. For Hilbert spaces $\mathcal{A}$ and $\mathcal{B}$, we write $\mathcal{A} \approx \mathcal{B}$ whenever $\dim \mathcal{A} = \dim \mathcal{B}$.

A linear mapping $\Phi : \mathrm{L}(\mathcal{A}) \mapsto \mathrm{L}(\mathcal{B})$ is called a *super-operator*. $\Phi$ is said to be *positive* if $\Phi(A) \in \mathrm{Pos}(\mathcal{B})$ for all $A \in \mathrm{Pos}(\mathcal{A})$. The super-operator $\Phi$ is said to be *completely positive* if $\Phi \otimes \mathbb{1}_{\mathrm{L}(\mathcal{Z})}$ is positive for every choice of the Hilbert space $\mathcal{Z}$. A super-operator $\Phi$ can be physically realized or is *admissible* if it is completely positive and preserves the trace: $\mathrm{tr}(\Phi(A)) = \mathrm{tr}(A)$ for all $A \in \mathrm{L}(\mathcal{A})$. We call such a super-operator a *quantum operation* or a *CPTP map*. Any quantum operation $\Phi : \mathrm{L}(\mathcal{A}) \mapsto \mathrm{L}(\mathcal{B})$ can be written in its Kraus form $\{E_j\}_{j=1}^{\dim(\mathcal{A})\cdot\dim(\mathcal{B})}$ where $E_j \in \mathrm{L}(\mathcal{A}, \mathcal{B})$ satisfies $\sum_j E_j^\dagger E_j = \mathbb{1}_{\mathcal{A}}$ and $\Phi(\rho) = \sum_j E_j \rho E_j^\dagger$, for any $\rho \in \mathrm{Pos}(\mathcal{A})$. Another, yet equivalent, way to represent any quantum operation is through a linear isometry $W \in \mathrm{L}(\mathcal{A}, \mathcal{B} \otimes \mathcal{Z})$ such that $\Phi(\rho) = \mathrm{tr}_{\mathcal{Z}}(W \rho W^\dagger)$, for some extra space $\mathcal{Z}$. Any such isometry $W$ can be implemented by a physical process as long as resources to implement the space $\mathcal{Z}$ are available. This is just a unitary transform in $\mathrm{U}(\mathcal{A} \otimes \mathcal{Z})$ where the system in $\mathcal{Z}$ is initially in known state $|0\rangle^{\mathcal{Z}}$. If $\mathscr{E} : \mathrm{L}(\mathcal{A}) \mapsto \mathrm{L}(\mathcal{B})$ and $\mathscr{F} : \mathrm{L}(\mathcal{B}) \mapsto \mathrm{L}(\mathcal{C})$ are two quantum operations (i.e. CPTP maps) then $\mathscr{F} \circ \mathscr{E} : \mathrm{L}(\mathcal{A}) \mapsto \mathrm{L}(\mathcal{C})$ denotes the quantum operation that first applies $\mathscr{E}$ then $\mathscr{F}$.

For $\rho_0, \rho_1 \in \mathrm{D}(\mathcal{A})$, we denote by $\Delta(\rho_0, \rho_1)$ the trace norm distance between $\rho_0$ and $\rho_1$: $\Delta(\rho_0, \rho_1) := \frac{1}{2} \| \rho_0 - \rho_1 \|$.

The Pauli group on one qubit is defined as $\mathrm{P}_1 := \{\pm\mathbb{1}, \pm i\mathbb{1}, \pm\mathsf{X}, \pm i\mathsf{X}, \pm\mathsf{Y}, \pm i\mathsf{Y}, \pm\mathsf{Z}, \pm i\mathsf{Z}\}$ while the $n$-qubit Pauli group is simply $\mathrm{P}_n := \mathrm{P}_1^{\otimes n}$. We denote by $\mathfrak{C}_n := \{U \in \mathrm{U}(\mathcal{H}_{2^n}) : (\forall \sigma \in \mathrm{P}_n)[U\sigma U^\dagger \in \mathrm{P}_n]\}$, the set of all Clifford operators acting on $n$ qubits. The Clifford group is just the set of $n$-qubit unitary operators $U$ with the property that for all Pauli operators $\sigma \in \mathrm{P}_n$, we have $\sigma' \in \mathrm{P}_n$ such that $U\sigma = \sigma' U$.

### 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 [LN11,HSS11] 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* [Reg05,HSS11]. The framework in [HSS11] 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 $\mathsf{id}_{\text{SC}}$. It is defined as follows:

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

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

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

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 [HSS11] 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 $\mathsf{id}^*_f$ as

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

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[LN11,HSS11] 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) [BCG$^+$02] based on Clifford operators introduced in [ABE08] 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 $|\text{acc}\rangle$ or $|\text{rej}\rangle$. A QAS is such that for all $k \in \mathcal{K}$, $(\mathscr{D}_k \circ \mathscr{E}_k)(\rho_S) = \rho_S \otimes |\text{acc}\rangle\langle\text{acc}|_F$.*

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 $\varepsilon$-secure if, for all attacks $U_{CR}$, there exists two CP maps $\mathscr{U}^{\text{acc}}_{R\to R}$ and $\mathscr{U}^{\text{rej}}_{R\to R}$ with $\mathscr{U}^{\text{acc}} + \mathscr{U}^{\text{rej}} = \mathbb{1}$ such that for all inputs $\psi_{SR}$, we*

*have that for some fixed state $\Omega_S$:*

$$\Big\| \frac{1}{\#\mathcal{K}} \sum_{k\in\mathcal{K}} \mathscr{D}_k \left( U_{CR} \mathscr{E}_k \left( \psi_{SR} \right) U_{CR}^\dagger \right) \\ - \left( \mathscr{U}^{\mathrm{acc}}(\psi_{SR}) \otimes |\mathrm{acc}\rangle\langle\mathrm{acc}|_F + \mathscr{U}^{\mathrm{rej}}(\psi_R) \otimes \Omega_S \otimes |\mathrm{rej}\rangle\langle\mathrm{rej}|_F \right) \Big\|_1 \leqslant \varepsilon \ . \tag{1}$$

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[ABE08]).** *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 $P_{\mathrm{acc}}^{SA} = \mathbb{1}_S \otimes |0^n\rangle\langle 0^n|_A$ and $P_{\mathrm{rej}}^{SA} = \mathbb{1}_{SA} - P_{\mathrm{acc}}^{SA}$:*

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

*where $\pi_{SA}$ is an arbitrary fixed state that the decoder outputs when it rejects the authentication.*

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 [ABE08] and is provided in Appendix C for completeness:

**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 $\ell$ 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 [Got97,ABE08]. 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 [DNS10,GW07], 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 call 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 $\mathcal{A}_i^{\mathscr{O},\mathrm{in}}$ and $\mathcal{B}_i^{\mathscr{O},\mathrm{in}}$ 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

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

A *communication oracle* from Alice to Bob is modeled by having $\mathcal{A}_i^{\mathscr{O},\mathrm{in}} \approx \mathcal{B}_i^{\mathscr{O},\mathrm{out}}$ and letting $\mathscr{O}_i$ move the state in $\mathcal{A}_i^{\mathscr{O},\mathrm{in}}$ to $\mathcal{B}_i^{\mathscr{O},\mathrm{out}}$. A classical ideal functionality, as those described in Sect. 2.1, can easily be made available as oracle calls. The parties place their inputs in the appropriate registers $\mathcal{A}_i^{\mathscr{O},\mathrm{in}}$ and $\mathcal{B}_i^{\mathscr{O},\mathrm{in}}$. The operation of the oracle $\mathscr{O}_i$ is as follows: it measures the input registers $\mathcal{A}_i^{\mathscr{O},\mathrm{in}}$ and $\mathcal{B}_i^{\mathscr{O},\mathrm{in}}$ 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 $\mathcal{A}_i^{\mathscr{O},\mathrm{out}}$ and $\mathcal{B}_i^{\mathscr{O},\mathrm{out}}$. The new internal state of the oracle 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 defined as:

**Definition 2.5.** *An* $m$-turn two-party hybrid protocol $\Pi_{\mathscr{F}}^{\mathscr{O}} = (\mathscr{A}, \mathscr{B}, \mathscr{O}, m)$ 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 $\Pi_{\mathscr{F}}^{\mathscr{O}}$ is denoted by $[\mathscr{A} \circledast \mathscr{B}]_{i+1}^{\mathscr{O}}(\rho_{\mathrm{in}}) := (\mathbb{1}_{\mathrm{L}(\mathcal{B}_{i+1} \otimes \mathcal{A}_{i+1} \otimes \mathcal{R})} \otimes \mathscr{O}_{i+1})(\mathscr{A}_{i+1} \otimes \mathscr{B}_{i+1} \otimes \mathbb{1}_{\mathrm{L}(\mathcal{R} \otimes \mathcal{O}_{i+1})})[\mathscr{A} \circledast \mathscr{B}]_i^{\mathscr{O}}(\rho_{\mathrm{in}})$, where $[\mathscr{A} \circledast \mathscr{B}]_0^{\mathscr{O}}(\rho_{\mathrm{in}}) := \rho_{\mathrm{in}}$, and $[\mathscr{A} \circledast \mathscr{B}]_{m+1}^{\mathscr{O}}(\rho_{\mathrm{in}}) := [\mathscr{A} \circledast \mathscr{B}]^{\mathscr{O}}(\rho_{\mathrm{in}})$.

## 3 Modeling Active Security

We start by extending the framework in [DNS10] to handle active security. Our model is very standard, defining security via simulation, but for completeness we describe and motivate the changes made in Appendix B.

Let $\Pi_{\mathscr{F}}^{\mathscr{O}} = (\mathscr{A}, \mathscr{B}, \mathscr{O}, m)$ be a $m$-turn two-party hybrid protocol. Let $\tilde{\mathscr{A}}$ and $\tilde{\mathscr{B}}$ be adversaries in $\Pi_{\mathscr{F}}^{\mathscr{O}}$. 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 $\mathcal{D}_0 = \mathcal{D}_{\mathrm{in}}$ while $\mathcal{O}_0$ is empty. In the ideal protocol, we let $\mathcal{C}_1^{\mathscr{F},\mathrm{in}} \approx \mathcal{C}_0$ and $\mathcal{D}_1^{\mathscr{F},\mathrm{in}} \approx \mathcal{D}_0$. $\mathscr{C}_1$ and $\mathscr{D}_1$ simply send their respective input stored in $\mathcal{C}_0$ and $\mathcal{D}_0$ to $\mathcal{C}_1^{\mathscr{F},\mathrm{in}}$ and $\mathcal{D}_1^{\mathscr{F},\mathrm{in}}$ respectively. The 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. $\mathscr{F}_1$ simply applies the quantum operation $\mathscr{F}$ to the inputs supplied by the parties and sends Charleen's output to Charleen in $\mathcal{C}_1^{\mathscr{F},\mathrm{out}}$. Dan's output is saved in the internal state $\mathcal{O}_1$ 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 denoted $|f\rangle$. We let $\mathscr{C}_2$ moves the output from the oracle stored in $\mathcal{C}_1^{\mathscr{F},\mathrm{out}}$ to $\mathcal{C}_2$ and sets $|f\rangle = |1\rangle$ in $\mathcal{C}_2^{\mathscr{F},\mathrm{in}}$. $\mathscr{D}_2$ does nothing at this point.
   For the second oracle $\mathscr{F}_2 : \mathrm{L}(\mathcal{C}_2^{\mathscr{F},\mathrm{in}} \otimes \mathcal{O}_1 \otimes \mathcal{D}_2^{\mathscr{F},\mathrm{in}}) \mapsto \mathrm{L}(\mathcal{C}_2^{\mathscr{F},\mathrm{out}} \otimes \mathcal{O}_2 \otimes \mathcal{D}_2^{\mathscr{F},\mathrm{out}})$, 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 $\mathcal{C}_2^{\mathscr{F},\mathrm{out}}$. When Charleen inputs $f = 1$, 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 $\mathcal{C}_3 := \mathcal{C}_{\mathrm{out}} \approx \mathcal{C}_2$ and $\mathcal{D}_3 := \mathcal{D}_{\mathrm{out}} \approx \mathcal{D}_2^{\mathscr{F},\mathrm{out}}$. $\mathscr{C}_3$ and $\mathscr{D}_3$ simply output whatever they received from $\mathscr{F}_1$ and $\mathscr{F}_2$ in $\mathcal{C}_2$ and $\mathcal{D}_2^{\mathscr{F},\mathrm{out}}$ respectively, except qubit $|a\rangle$ in the latter register. When the execution has aborted, Dan outputs a dummy state.

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 $\Gamma_{\mathscr{F}} = (\mathscr{C}, \mathscr{D}, \mathscr{F}, 2)$ be the ideal protocol for $\mathscr{F}$ without fairness for Dan. Let $\delta \in [0..1]$. We say that $\Pi^{\mathscr{O}}_{\mathscr{F}}$ is* $\delta$-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 $\delta$-active secure for $\delta = 0$, then it is called* perfectly active secure*. If it is $\delta$-active secure for $\delta$ 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 both players are *ultimately specious*. It relies on a string commitment scheme and secure TPCs provided as oracles, as described in Sect. 2.1.

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 subprotocol $\mathtt{Swaddle}(\mathtt{w})$, described below, uses two applications 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 *innermost* and Bob the *outermost* key of the resulting swaddling $\mathfrak{s}(\mathtt{w})$ for 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:

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

where $K'_{\mathtt{w},a}$, and $K'_{\mathtt{w},b}$ correspond to random Clifford operations $A' \in \mathfrak{C}_{2n+1}$ and $B' \in \mathfrak{C}_{n+1}$ respectively, and $t$ corresponds to Clifford $T \in \mathfrak{C}_{2n+1}$ subject to:

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

$\mathtt{Swaddle}(\mathcal{W})$**, 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 **calls** $\mathtt{TestSwaddling}(\mathfrak{s}(\mathtt{w}))$.

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 makes sure that the state $|\sigma\rangle$ of a valid swaddling $\mathfrak{s}(\mathtt{w})$ satisfies $T|\sigma\rangle = |0^n\rangle \otimes B'(|\varphi\rangle \otimes |0^n\rangle)$, 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. The security of $\mathtt{Swaddle}(\mathtt{w})$ will be established in Appendix G.

$\mathtt{TestSwaddling}(\mathfrak{s}(\mathtt{w}))$**, with Alice doing the testing:**
1. **If** Alice holds the outermost authentication key $K_{\mathtt{w},a}$ **then**:
   - Alice applies $A^\dagger$ on $\mathfrak{s}(\mathtt{w})$, where $A$ is the Clifford corresponding to string $K_{\mathtt{w},a}$,
   - Alice tests that her dummies are together in state $|0^n\rangle$. **Otherwise**, she **aborts**.
   - Alice re-applies $A$ on the $2n+1$ qubits of the swaddling.
2. **Else** they **call** $((t, K'_{\mathtt{w},a}), K'_{\mathtt{w},b}) = \mathsf{id}_{\text{TEST}}((id_A(\mathtt{w}), id_B(\mathtt{w})), (id_A(\mathtt{w}), id_B(\mathtt{w})))$,
   - **If** one party gets $\perp$ in $\mathsf{id}_{\text{TEST}}$ **then abort**.
   - Alice applies $T$ on $\mathfrak{s}(\mathtt{w})$ where $T$ is the Clifford operator corresponding to string $t$.
   - Alice tests that her dummies are together in state $|0^n\rangle$. **Otherwise**, she **aborts**.
   - 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).

At the end of $\mathtt{TestSwaddling}(\mathfrak{s}(\mathtt{w}))$, Alice holds the outermost authentication key $K'_{\mathtt{w},a}$ of the resulting swaddling. Bob can do the testing by the exact same procedure provided the roles of Alice and Bob are reversed in $\mathtt{TestSwaddling}(\mathfrak{s}(\mathtt{w}))$. The security of these procedures will be discussed in Sect. 5.5. Intuitively, we expect that $\mathtt{TestSwaddling}(\mathfrak{s}(\mathtt{w}))$ 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 $\mathtt{w}$ [BCG+02].

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. This subprotocol is called $\mathtt{TestAllSwaddlings}$. It simply consists in the execution of $\mathtt{TestSwaddling}(\mathfrak{s}(\mathtt{w}))$ for all wires $\mathtt{w}$ held by each party.

`TestAllSwaddlings`:
1. For all wires $\mathtt{w}$ held by Bob, Bob sends $\mathfrak{s}(\mathtt{w})$ to Alice before Alice and Bob run $\texttt{TestSwaddling}(\mathfrak{s}(\mathtt{w}))$ with Alice doing the testing.
2. All swaddles $\mathfrak{s}(\mathtt{w})$ received by Alice are sent back to Bob before Alice and Bob run $\texttt{TestSwaddling}(\mathfrak{s}(\mathtt{w}))$ with Bob doing the testing.
3. For all wires $\mathtt{w}$ held by Alice, Alice sends $\mathfrak{s}(\mathtt{w})$ to Bob before Alice and Bob run $\texttt{TestSwaddling}(\mathfrak{s}(\mathtt{w}))$ with Bob doing the testing.
4. All swaddles $\mathfrak{s}(\mathtt{w})$ received by Bob are sent back to Alice before Alice and Bob run $\texttt{TestSwaddling}(\mathfrak{s}(\mathtt{w}))$ with Alice doing the testing.

The following subprotocol implements the obvious way the openings of all committed Clifford operators, thereby allowing Alice and Bob to get the final state of the computation. Notice that since Alice learns Bob's secret keys before she unveils her own, our protocol will lack fairness for Bob.

`OpenAllSwaddlings`:
1. **For** each wire $\mathtt{w} \in \mathcal{A}_{\text{out}}$, Bob reveals to Alice his committed secret key encrypting $\mathtt{w}$, with identifier $id_B(\mathtt{w})$, by **calling** $\mathsf{id}_{\text{OPEN}}(id_B(\mathtt{w}), id_B(\mathtt{w}))$. **If** Alice or Bob get $\perp$ from the ideal functionality **then abort**.
2. **For** each wire $\mathtt{w} \in \mathcal{B}_{\text{out}}$, Alice reveals to Bob her committed secret key encrypting $\mathtt{w}$, with identifier $id_A(\mathtt{w})$, by **calling** $\mathsf{id}_{\text{OPEN}}(id_A(\mathtt{w}), id_A(\mathtt{w}))$. **If** Alice or Bob get $\perp$ from the ideal functionality **then abort**.

## 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.

The core idea of our protocol is the same as in [DNS10]: 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 Clifford gates (i.e., all gates in $\mathcal{UG}$ except for $\mathsf{R}$), 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 $\mathsf{R}$-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 [BK05] (where $|M\rangle$ is an "$H$-type magic state" in their language); a description of it can also be found in Appendix D.

### 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 $\mathscr{F} : \mathrm{L}(\mathcal{A}_{\text{in}} \otimes \mathcal{B}_{\text{in}}) \mapsto \mathrm{L}(\mathcal{A}_{\text{out}} \otimes \mathcal{B}_{\text{out}})$ upon joint input state $\rho_{\text{in}} \in$

$\mathrm{D}(\mathcal{R} \otimes \mathcal{A}_{\rm in} \otimes \mathcal{B}_{\rm in})$. 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 $\mathcal{A}_{\rm in} \otimes \mathcal{B}_{\rm in}$ 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 $\mathcal{B}_e$). In the following, we denote by $U_{\mathscr{F}} \in \mathrm{U}(\mathcal{A}_{\rm in} \otimes \mathcal{A}_a \otimes \mathcal{B}_{\rm in} \otimes \mathcal{B}_a)$ the unitary transform implemented by the circuit that satisfies, for all mixed quantum state $\rho_{\rm in}^{\mathcal{A}_{\rm in}\mathcal{B}_{\rm in}}$,

$$\mathscr{F}(\rho_{\rm in}^{\mathcal{A}_{\rm in}\mathcal{B}_{\rm in}}) = \mathrm{tr}_{\mathcal{A}_e\mathcal{B}_e}\left(U_{\mathscr{F}}\rho_{\rm in}^{\mathcal{A}_{\rm in}\mathcal{B}_{\rm in}} \otimes |0\rangle\langle 0|^{\mathcal{A}_a} \otimes |0\rangle\langle 0|^{\mathcal{B}_a} U_{\mathscr{F}}^\dagger\right) \quad ,$$

where $\mathcal{A}_e \subseteq \mathcal{A}_{\rm in} \otimes \mathcal{A}_a$, $\mathcal{B}_e \subseteq \mathcal{B}_{\rm in} \otimes \mathcal{B}_a$.

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 $G_i$ is executed before $G_{i+1}$. This protocol calls a number of subprotocols described next and in Appendix E.

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

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 **call** `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 **call** `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 **call** `RGate`$(\mathfrak{s}(\mathtt{w}))$,
3. Alice and Bob **call** `TestAllSwaddlings`,
4. Alice and Bob **call** `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.

## 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 $|M\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{i\pi/4}|1\rangle)$ 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 **calls** `VerifyAncilla`$(\mathfrak{s}(\mathtt{a}))$.
8. **For** every wire $\mathtt{a} \in \mathcal{B}_a$ or added in step 4 above, Bob **calls** `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`$(\mathfrak{s}(\mathtt{m}))$.

Subprotocols `VerifyAncilla`$(\mathfrak{s}(\mathtt{a}))$ and `DistillMagic`$(\mathfrak{s}(\mathtt{m}))$ are described in Appendices E and D. The following subprotocols evaluates any one-qubit Clifford, the CNOT-gate, respectively the R. The idea behind the one-qubit protocol is to use TPC to compute the gate *through* the

authentication. The idea in the CNOT-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.

To express the TPCs needed for these subprotocols, we will need to define some more ideal functionalities. For the one-qubit Clifford gates, we will define $\mathsf{id}_{1\mathrm{QC}}$, which simply updates the commitment to the inner key of the swaddling: a call to $\mathsf{id}_{1\mathrm{QC}}((i,G),(i',G'))$, if $i = i' = id_A(\mathtt{w})$ for some wire $\mathtt{w}$ for which Alice has the inner swaddling and $G = G'$ is some one-qubit Clifford, results in the commitments to the key $K_{\mathtt{w},a}$ to be updated to commitments to $K_{\mathtt{w},a}G^\dagger$(and likewise when Bob holds the inner key).

The subprotocols are then as follows:

---

`OneQubitClifford`$(\mathtt{C}, \mathfrak{s}(\mathtt{w}))$**:**

1. If Alice has the inner swaddling of $\mathtt{w}$, Alice and Bob call $\mathsf{id}_{1\mathrm{QC}}((id_A(\mathtt{w}),C),(id_A(\mathtt{w}),C))$, otherwise they call $\mathsf{id}_{1\mathrm{QC}}((id_B(\mathtt{w}),C),(id_B(\mathtt{w}),C))$.

---

`CNOT`$(\mathfrak{s}(\mathtt{w_c}),\mathfrak{s}(\mathtt{w_t}))$ **with wire** $\mathtt{w}_c \in \mathcal{A}$ **as control and** $\mathtt{w}_t$ **as target:**

1. Bob **sends** $\mathfrak{s}(\mathtt{w}_t)$ to Alice if Bob holds it, in which case Alice **calls** `TestDummies`$(\mathfrak{s}(\mathtt{w_t}))$.
2. Alice performs a randomly selected Clifford $C$ on $4n+2$ qubits jointly on $\mathfrak{s}(\mathtt{w}_c)$ and $\mathfrak{s}(\mathtt{w}_t)$.
3. Alice **sends** $\mathfrak{s}(\mathtt{w}_c)$ and $\mathfrak{s}(\mathtt{w}_t)$ to Bob; Bob **calls** `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** $\mathfrak{s}(\mathtt{w}_c)$ (and $\mathfrak{s}(\mathtt{w}_t)$ if she held this one too) to Alice. Alice **calls** `TestDummies` on them.

---

`RGate`$(\mathfrak{s}(\mathtt{w}))$ **with** $\mathtt{w} \in \mathcal{A}$**:**

1. 
2. Alice performs a swaddled CNOT with $\mathtt{m}_w$ as a control, and $\mathtt{w}$ as target.
3. Alice **sends** $\mathfrak{s}(\mathtt{w})$ to Bob. Bob **calls** `TestSwaddling`$(\mathfrak{s}(\mathtt{w}))$ on it.
4. Alice **calls** `Measure`$(\mathfrak{s}(\mathtt{w}))$.
5. 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.
6. Alice relabels $\mathtt{m}_{a,i}$ to $\mathtt{w}$.

---

The `RGate` subprotocol performs the R-gate using gate teleportation[GC99b,GC99a] via the magic state similarly to the fault-tolerant version of the R-gate introduced in [Got10,Sho96]. To perform an R-gate on a wire $\mathtt{w}$ via gate teleportation, we would first perform a CNOT from the magic state to $\mathtt{w}$, and then measure $\mathtt{w}$ 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 $\mathtt{w}$. `Measure`$(\mathfrak{s}(\mathtt{w}))$ is described in Appendix E.

### 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 negligible in the number of dummy wires per qubit.

### 5.4 Some additional definitions

Before we start, we will find it convenient to introduce some additional notation. From now on, $\rho_0$ will consist of $\rho_{\text{in}}$ 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.

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 $i$th 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 Appendix F 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. 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 Appendix G, 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 and Appendix C).

**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

In this section, we prove active security of $\widehat{\Pi}_{\mathscr{F}}^{\mathscr{O}'}$ by providing a simulator $\tilde{\mathscr{C}}$. We prove that for any shared input state, when $\tilde{\mathscr{C}}$ interacts with $\mathscr{D}$, it reproduces the state generated by $\tilde{\mathscr{A}}$ interacting with $\mathscr{B}$ on that same input state.

At the beginning, $\tilde{\mathscr{C}}$ simulates the view of any execution between $\tilde{\mathscr{A}}$ and $\mathscr{B}$ that aborts before all swaddlings are opened. This part of the simulation is described and analyzed in the next subsection. In Sect. 6.2, we provide and analyze the simulation when no early aborting occurs between $\tilde{\mathscr{A}}$ and $\mathscr{B}$.

### 6.1 Simulation of Early Aborting Executions

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 $|0\rangle^{\mathcal{B}_{\text{in}}}$. 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 $\mathscr{B}^*$, register $\mathcal{B}$ of $\mathscr{B}$ will be denoted by $\mathcal{B}^*$ when used by $\mathscr{B}^*$. Consider the simulator $\tilde{\mathscr{C}}$ starting its simulation as follows:

1. $\tilde{\mathscr{C}}$ simulates the protocol execution between $\tilde{\mathscr{A}}$ and $\mathscr{B}^*$ holding the dummy input state $|0^n\rangle^{\mathcal{B}^*_{\text{in}}}$ until turn $s^*$ is reached. Remember that $\tilde{\mathscr{C}}$ simulates all oracle calls during the execution between $\tilde{\mathscr{A}}$ and $\mathscr{B}^*$. In particular $\tilde{\mathscr{C}}$ stores all queries to classical oracles made by $\tilde{\mathscr{A}}$ and $\mathscr{B}^*$.

2. If the execution **aborts** at any point before $\tilde{\mathscr{A}}$ and $\mathscr{B}^*$ reach turn $s^*$ then $\tilde{\mathscr{C}}$ **aborts** as well after having deleted all its internal registers except the ones held by $\tilde{\mathscr{A}}$.

At Step 2 , $\tilde{\mathscr{C}}$ and $\mathscr{B}^*$ **aborts** with the same probability as a real world execution aborts prior to turn $s^*$. The simulation reproduces $\tilde{\mathscr{A}}$'s view in the real world when such an aborting execution occurs. This follows from the fact that up to that point, $\mathscr{B}^*$ never provides any information about its input state. We now prove that the simulation of an early abort is correct. We only provide the proof that the simulator does the job when the adversary $\tilde{\mathscr{A}}$ is impersonating $\mathscr{A}$. The proof for an adversary $\tilde{\mathscr{B}}$ impersonating $\mathscr{B}$ follows the same lines.

We define $\text{E-ABORT}_s$ as the event consisting in an execution between two parties aborting at turn $s < s^*$. Let $[\tilde{\mathscr{A}} \circledast \mathscr{B} \wedge \text{E-ABORT}_s]^{\mathscr{O}'}(\rho_{\text{in}})$ be denoting the the joint state of an execution that aborts at turn $s$ whenever $\tilde{\mathscr{A}}$ and $\mathscr{B}$ are interacting upon joint input state $\rho_{\text{in}}$. Let $[\tilde{\mathscr{C}} \circledast \mathscr{D} \wedge \text{E-ABORT}_s]^{\mathscr{F}}(\rho_{\text{in}})$ be denoting an execution between $\tilde{\mathscr{C}}$ and $\mathscr{D}$ (in the ideal world) upon joint input state $\rho_{\text{in}}$ aborting while $\tilde{\mathscr{C}}$ is simulating $\tilde{\mathscr{A}}$ and $\mathscr{B}^*$ at turn $s$ upon input state $\text{tr}_{\mathcal{B}_{\text{in}}}(\rho_{\text{in}}^{\mathcal{A}_{\text{in}}\mathcal{B}_{\text{in}}\mathcal{R}}) \otimes |0\rangle\langle 0|^{\mathcal{B}^*_{\text{in}}}$ . States $[\tilde{\mathscr{C}} \circledast \mathscr{D} \wedge \text{E-ABORT}_s]^{\mathscr{F}}(\rho_{\text{in}})$ and $[\tilde{\mathscr{A}} \circledast \mathscr{B} \wedge \text{E-ABORT}_s]^{\mathscr{O}'}(\rho_{\text{in}})$ are not normalized, $\text{tr}([\tilde{\mathscr{C}} \circledast \mathscr{D} \wedge \text{E-ABORT}_s]^{\mathscr{F}}(\rho_{\text{in}}))$ and $\text{tr}([\tilde{\mathscr{A}} \circledast \mathscr{B} \wedge \text{E-ABORT}_s]^{\mathscr{O}'}(\rho_{\text{in}}))$ are the probabilities that $[\tilde{\mathscr{C}} \circledast \mathscr{D}]^{\mathscr{F}}(\rho_{\text{in}})$ and $[\tilde{\mathscr{A}} \circledast \mathscr{B}]^{\mathscr{O}'}(\rho_{\text{in}})$ *aborts* at step $s$ respectively. We have,

**Lemma 6.1 (Early abort).** *Let $\tilde{\mathscr{A}}$ be an adversary in hybrid protocol $\widehat{\Pi}^{\mathscr{O}'}_{\mathscr{F}} = (\mathscr{A}, \mathscr{B}, \mathscr{O}', m)$ for $\mathscr{F} : \text{L}(\mathcal{A}_{\text{in}} \otimes \mathcal{B}_{\text{in}}) \to \text{L}(\mathcal{A}_{\text{out}} \otimes \mathcal{B}_{\text{out}})$. 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 $\rho_{\text{in}} \in \text{D}(\mathcal{A}_{\text{in}} \otimes \mathcal{B}_{\text{in}} \otimes \mathcal{R})$,*

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

*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 $\Gamma_{\mathscr{F}}$.*

*Proof (sketch).* The proof proceeds along the same lines than in [DNS10]. Since all communications between parties are statistically encrypted, an execution between $\tilde{\mathscr{A}}$ and $\mathscr{B}$ upon joint input state $\rho_{\text{in}}^{\mathcal{A}_{\text{in}}\mathcal{R}} \otimes |0\rangle\langle 0|^{\mathcal{B}_{\text{in}}}$ is identical to an execution with joint input state $\rho_{\text{in}}$ from the adversary's

point of view. Remember that when $\tilde{\mathscr{C}} \circledast \mathscr{D}$ and $\tilde{\mathscr{A}} \circledast \mathscr{B}$ abort at turn $s < s^*$, $\mathscr{D}$ and $\mathscr{B}$ output a special state $|\bot\rangle\langle\bot|$. We have,

$$\begin{aligned}
&\Delta\left([\tilde{\mathscr{C}} \circledast \mathscr{D} \wedge \text{E-ABORT}_s]^{\mathscr{F}}(\rho_{\text{in}}), [\tilde{\mathscr{A}} \circledast \mathscr{B} \wedge \text{E-ABORT}_s]^{\mathscr{O}'}(\rho_{\text{in}})\right) \\
&= \Delta\left([\tilde{\mathscr{A}} \circledast \mathscr{B} \wedge \text{E-ABORT}_s]^{\mathscr{O}'}(\rho_{\text{in}}^{\mathcal{A}_{\text{in}}\mathcal{R}} \otimes |0\rangle\langle 0|^{\mathcal{B}_{\text{in}}}), [\tilde{\mathscr{A}} \circledast \mathscr{B} \wedge \text{E-ABORT}_s]^{\mathscr{O}'}(\rho_{\text{in}})\right) \qquad (2) \\
&= \Delta\Big(\text{tr}_{\mathscr{B}}\left([\tilde{\mathscr{A}} \circledast \mathscr{B} \wedge \text{E-ABORT}_s]^{\mathscr{O}'}(\rho_{\text{in}}^{\mathcal{A}_{\text{in}}\mathcal{R}} \otimes |0\rangle\langle 0|^{\mathcal{B}_{\text{in}}})\right), \\
&\qquad\qquad \text{tr}_{\mathscr{B}}\left([\tilde{\mathscr{A}} \circledast \mathscr{B} \wedge \text{E-ABORT}_s]^{\mathscr{O}'}(\rho_{\text{in}})\right)\Big) \qquad (3) \\
&\leq \mathsf{negl}(n) \ , \qquad (4)
\end{aligned}$$

where (2) is a consequence of how the simulator is constructed, and (3) follows from the fact that in such executions, the output of the honest player is $|\bot\rangle\langle\bot|$. Equation (4) follows from the hiding property at every turn stated in Lemma 5.4. Notice that some measurements can be applied to some swaddled ancillas (initially in state $|0\rangle$) or magic states while performing `VerifyMagic` or evaluating an $\mathsf{R}$-gate. None of these measurements can reveal anything non-negligible about $\mathscr{B}$'s input state. For the evaluation of an $\mathsf{R}$-gate, the outcome of measurements are independent of $\mathscr{B}$'s input state. For the magic state distillation process, measurements are only applied to candidate magic states also independent of $\mathscr{B}$'s input state.

To prove the statement for adversary $\tilde{\mathscr{B}}$ interacting with $\mathscr{A}$ in $\widehat{\Pi}_{\mathscr{F}}^{\mathscr{O}'}$, we define $s^*$ to be the turn at which $\tilde{\mathscr{B}}$ receives all his swaddlings from $\mathscr{A}$. The simulator $\tilde{\mathscr{D}}$ for executions aborting early proceeds the same way than $\tilde{\mathscr{C}}$ except that $\tilde{\mathscr{B}}$ replaces $\tilde{\mathscr{A}}$ while $\mathscr{A}^*$ replaces $\mathscr{B}^*$. □

### 6.2 Simulation of Executions that do not Abort Early

It remains to simulate the execution from turn $s^*$ until the end. In the simulated world, if $\tilde{\mathscr{C}}$ reaches the end of Step 2 then the output state $\mathscr{F}(\rho_{\text{in}}^{\mathcal{A}_{\text{in}}\mathcal{R}} \otimes |0\rangle\langle 0|)$ can be recovered. The reason being that at turn $s^*$ (taking place before `OpenAllSwaddlings`), 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 [DNS10]. 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 $\mathscr{B}^*$. $\tilde{\mathscr{C}}$ then recovers the output state. $\tilde{\mathscr{C}}$ 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. This operation is performed using $U_{\mathscr{F}}^{\dagger} \in \text{U}(\mathcal{A}_{\text{out}} \otimes \mathcal{A}_a \otimes \mathcal{B}_{\text{out}} \otimes \mathcal{B}_a)$, where $U_{\mathscr{F}}$ is the unitary implemented by the circuit as defined in Sect. 5.1. Obviously, $U_{\mathscr{F}}^{\dagger}$ has the same circuit complexity than $U_{\mathscr{F}}$.

Then, $\tilde{\mathscr{C}}$ 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 during the execution of `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.

More precisely, the simulator $\tilde{\mathscr{C}}$ will proceed as follows for the rest of the simulation:

3. When $\tilde{\mathscr{C}}$ receives from $\mathscr{B}^*$ all of $\tilde{\mathscr{A}}$'s swaddlings in `TestAllSwaddlings`, it decrypts them together with all of $\mathscr{B}^*$'s swaddlings. This can be done since $\tilde{\mathscr{C}}$ knows all (classical) encryption keys. The ancilla held in register $\mathcal{Z}$ is also decrypted.
4. The output of the computation (i.e., between $\tilde{\mathscr{A}}$ and $\mathscr{B}^*$) is now available to $\tilde{\mathscr{C}}$. $\tilde{\mathscr{C}}$ then runs $U_{\mathscr{F}}^{\dagger}$ upon the output registers in order to recover $\tilde{\mathscr{A}}$'s effective input state.
5. $\tilde{\mathscr{C}}$ queries $\mathscr{F}_1$ with $\tilde{\mathscr{A}}$'s effective input state. $\tilde{\mathscr{C}}$ then re-swaddles all of $\tilde{\mathscr{A}}$'s logical wires (now storing the real-world output of the computation) using all secret keys used by $\tilde{\mathscr{C}}$ at Step 3. $\tilde{\mathscr{C}}$ then sends all of $\tilde{\mathscr{A}}$'s swaddlings back to $\tilde{\mathscr{A}}$.
6. $\tilde{\mathscr{A}}$ and $\mathscr{B}^*$ resume their execution from turn $s^*$ onward. If $\tilde{\mathscr{A}}$ **aborts** the execution before $\mathscr{B}^*$ gets his output state then $\tilde{\mathscr{C}}$ queries $\mathscr{F}_2$ with $f = 0$ otherwise $\tilde{\mathscr{C}}$ queries $\mathscr{F}_2$ with $f = 1$. $\tilde{\mathscr{C}}$ deletes all its internal registers except the ones held by $\tilde{\mathscr{A}}$.

Let $\overline{\text{E-ABORT}}$ be the event of not having $\text{E-ABORT}_s$ at any turn $s < s^*$. Let $[\tilde{\mathscr{A}} \circledast \mathscr{B} \wedge \overline{\text{E-ABORT}}]^{\mathscr{O}'}(\rho_{\text{in}})$ be denoting an execution between $\tilde{\mathscr{A}}$ and $\mathscr{B}$ upon joint input state $\rho_{\text{in}}$ that does not abort before turn $s^*$. Let $[\tilde{\mathscr{C}} \circledast \mathscr{D} \wedge \overline{\text{E-ABORT}}]^{\mathscr{F}}(\rho_{\text{in}})$ be denoting an execution between $\tilde{\mathscr{C}}$ and $\mathscr{B}$, where the simulated $\tilde{\mathscr{A}}$ and $\mathscr{B}^*$ do not abort before turn $s^*$ upon joint input state $\text{tr}_{\mathcal{B}_{\text{in}}}(\rho_{\text{in}}^{\mathcal{A}_{\text{in}}\mathcal{B}_{\text{in}}\mathcal{R}}) \otimes |0\rangle\langle 0|^{\mathcal{B}^*_{\text{in}}}$.

Next lemma establishes the active security when no early aborting occurs. The following lemma follows from the forcingness of the protocol established in Lemma 5.5.

**Lemma 6.2 (No early abort).** *For any quantum adversary $\tilde{\mathscr{A}}$ in hybrid protocol $\widehat{\Pi}_{\mathscr{F}}^{\mathscr{O}'} = (\mathscr{A}, \mathscr{B}, \mathscr{O}', m)$ for $\mathscr{F} : \text{L}(\mathcal{A}_{\text{in}} \otimes \mathcal{B}_{\text{in}}) \to \text{L}(\mathcal{A}_{\text{out}} \otimes \mathcal{B}_{\text{out}})$, 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 $\rho_{\text{in}} \in \text{D}(\mathcal{A}_{\text{in}} \otimes \mathcal{B}_{\text{in}} \otimes \mathcal{R})$,*

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

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

*Proof (sketch).* Remember that turn $s^*$ is where $\mathscr{B}^*$ is sending $\tilde{\mathscr{A}}$'s swaddlings after having tested them with success in `TestAllSwaddlings`. At Step 3, $\tilde{\mathscr{C}}$ intercepts this quantum transmission from $\mathscr{B}^*$ to $\tilde{\mathscr{A}}$. Let $[\tilde{\mathscr{C}} \circledast \mathscr{D}]_{\mathsf{S3}}^{\mathscr{F}}(\rho_{\text{in}}^{\mathcal{A}_{\text{in}}\mathcal{B}_{\text{in}}\mathcal{R}})$ be denoting the joint state when $\tilde{\mathscr{C}}$ reaches Step 3 in the simulation above. As we have seen in the proof of Lemma 6.1, we have $[\tilde{\mathscr{C}} \circledast \mathscr{D}]_{\mathsf{S3}}^{\mathscr{F}}(\rho_{\text{in}}^{\mathcal{A}_{\text{in}}\mathcal{B}_{\text{in}}\mathcal{R}}) = [\tilde{\mathscr{A}} \circledast \mathscr{B}]_{s^*}^{\mathscr{O}'}(\rho_{\text{in}}^{\mathcal{A}_{\text{in}}\mathcal{R}} \otimes |0\rangle\langle 0|^{\mathcal{B}_{\text{in}}})$ since turn $s^*$ occurs before any player reveals its encryption keys to the other player. Moreover, Lemma 5.5 tells us that

$$\Delta\left([\tilde{\mathscr{A}} \circledast \mathscr{B}]_{s^*}^{\mathscr{O}'}(\rho_{\text{in}}^{\mathcal{A}_{\text{in}}\mathcal{R}} \otimes |0\rangle\langle 0|^{\mathcal{B}_{\text{in}}}), \tilde{\mathscr{E}} \circ \mathsf{C}_{s^*} \circ \mathscr{G}_{s^*} \circ \tilde{\mathscr{E}}_0(\rho_{\text{in}}^{\mathcal{A}_{\text{in}}\mathcal{R}} \otimes |0\rangle\langle 0|^{\mathcal{B}_{\text{in}}})\right) \leq \mathsf{negl}(n) \ . \quad (5)$$

where $\tilde{\mathscr{E}}_0$ is defined in Definition 5.1, $\mathscr{G}_{s^*}$ is the *honest* operation so far, and $\tilde{\mathscr{E}}$ does not act on any of the swaddlings held by $\tilde{\mathscr{C}}$ since $\mathscr{B}^*$ just checked them with success (since $\overline{\text{E-ABORT}}$) and are not held by $\tilde{\mathscr{A}}$ yet. At this point of the protocol, $\mathscr{G}_{s^*}$ corresponds to the application of $U_{\mathscr{F}} \in \text{U}(\mathcal{A}_{\text{in}} \otimes \mathcal{A}_a \otimes \mathcal{B}_{\text{in}} \otimes \mathcal{B}_a)$ where $\mathcal{A}_a$ and $\mathcal{B}_a$ are the spaces of the ancilla needed to implement $\mathscr{F}$ unitarily. Notice that $\mathsf{C}_{s^*}$ is the authentication encoding using all keys (and not only $\mathscr{B}^*$'s keys)

since all $\tilde{\mathscr{A}}$'s swaddlings have been tested with success by $\mathscr{B}^*$ before they were returned to $\tilde{\mathscr{A}}$. We have:

$$\tilde{\mathscr{E}} \circ \mathsf{C}_{s^*} \circ \mathscr{G}_{s^*} \circ \tilde{\mathscr{E}}_0(\rho_{\text{in}}^{\mathcal{A}_{\text{in}}\mathcal{R}} \otimes |0\rangle\langle 0|^{\mathcal{B}^*_{\text{in}}}) = \tilde{\mathscr{E}} \circ \mathsf{C}_{s^*} \circ U_{\mathscr{F}} \left( \tilde{\mathscr{E}}_0(\rho_{\text{in}}^{\mathcal{A}_{\text{in}}\mathcal{R}}) \otimes |0\rangle\langle 0|^{\mathcal{B}^*_{\text{in}}} \otimes |0\rangle\langle 0|^{\mathcal{Z}} \right) U_{\mathscr{F}}^\dagger \ .$$

Using this in (5) before applying $U_{\mathscr{F}}^\dagger \circ \mathsf{C}_{s^*}^\dagger$, as $\tilde{\mathscr{C}}$ does at Steps 3 and 4, results in:

$$\Delta \Big( U_{\mathscr{F}}^\dagger \circ \mathsf{C}_{s^*}^\dagger \circ [\tilde{\mathscr{A}} \circledast \mathscr{B}]_{s^*}^{\mathscr{O}'}(\rho_{\text{in}}^{\mathcal{A}_{\text{in}}\mathcal{R}} \otimes |0\rangle\langle 0|^{\mathcal{B}^*_{\text{in}}}), \\ \tilde{\mathscr{E}} \circ \left( \tilde{\mathscr{E}}_0(\rho_{\text{in}}^{\mathcal{A}_{\text{in}}\mathcal{R}}) \otimes |0\rangle\langle 0|^{\mathcal{B}^*_{\text{in}}} \otimes |0\rangle\langle 0|^{\mathcal{Z}} \right) \Big) \leq \mathsf{negl}(n) \ .$$

At Step 5, $\tilde{\mathscr{C}}$ calls the ideal functionality $\mathscr{F}_1$ evaluating $\mathscr{F}$ upon $\tilde{\mathscr{A}}$'s effective input together with $\mathscr{B}$'s input while keeping $\mathscr{B}$'s output state:

$$\Delta \Big( \mathscr{F}_1 \circ U_{\mathscr{F}}^\dagger \circ \mathsf{C}_{s^*}^\dagger \circ [\tilde{\mathscr{A}} \circledast \mathscr{B}]_{s^*}^{\mathscr{O}'}(\rho_{\text{in}}^{\mathcal{A}_{\text{in}}\mathcal{B}_{\text{in}}\mathcal{R}} \otimes |0\rangle\langle 0|^{\mathcal{B}^*_{\text{in}}}), \\ \tilde{\mathscr{E}} \circ \left( \mathscr{F}_1(\tilde{\mathscr{E}}_0(\rho_{\text{in}}^{\mathcal{A}_{\text{in}}\mathcal{B}_{\text{in}}\mathcal{R}})) \otimes |0\rangle\langle 0|^{\mathcal{B}^*_{\text{in}}} \otimes |0\rangle\langle 0|^{\mathcal{Z}} \right) \Big) \leq \mathsf{negl}(n) \ . \quad (6)$$

At this point, $\tilde{\mathscr{C}}$ encrypts/authenticates, using the same keys as before, all of $\tilde{\mathscr{A}}$'s output wires (including her part of $\mathcal{Z}$ and $\mathcal{B}^*_{\text{in}}$) held by $\tilde{\mathscr{C}}$:

$$\Delta \Big( \mathsf{C}_{s^*} \circ \mathscr{F}_1 \circ U_{\mathscr{F}}^\dagger \circ \mathsf{C}_{s^*}^\dagger \circ [\tilde{\mathscr{A}} \circledast \mathscr{B}]_{s^*}^{\mathscr{O}'}(\rho_{\text{in}}^{\mathcal{A}_{\text{in}}\mathcal{B}_{\text{in}}\mathcal{R}} \otimes |0\rangle\langle 0|^{\mathcal{B}^*_{\text{in}}}), \\ \tilde{\mathscr{E}} \circ \mathsf{C}_{s^*} \left( \mathscr{F}_1(\tilde{\mathscr{E}}_0(\rho_{\text{in}}^{\mathcal{A}_{\text{in}}\mathcal{B}_{\text{in}}\mathcal{R}})) \otimes |0\rangle\langle 0|^{\mathcal{B}^*_{\text{in}}} \otimes |0\rangle\langle 0|^{\mathcal{Z}} \right) \Big) \leq \mathsf{negl}(n) \ . \quad (7)$$

Applying Lemma 5.5 once again, results in:

$$\Delta \left( [\tilde{\mathscr{A}} \circledast \mathscr{B}]_{s^*}^{\mathscr{O}'}(\rho_{\text{in}}), \tilde{\mathscr{E}} \circ \mathsf{C}_{s^*} \circ \mathscr{F} \left( \tilde{\mathscr{E}}_0(\rho_{\text{in}}^{\mathcal{A}_{\text{in}}\mathcal{B}_{\text{in}}\mathcal{R}}) \right) \right) \leq \mathsf{negl}(n) \ . \quad (8)$$

Let $[\tilde{\mathscr{C}} \circledast \mathscr{D}]_{\mathsf{S6}}^{\mathscr{F}}(\rho_{\text{in}}^{\mathcal{A}_{\text{in}}\mathcal{B}_{\text{in}}\mathcal{R}}) := \mathsf{C}_{s^*} \circ \mathscr{F}_1 \circ U_{\mathscr{F}}^\dagger \circ \mathsf{C}_{s^*}^\dagger \circ [\tilde{\mathscr{A}} \circledast \mathscr{B}]_{s^*}^{\mathscr{O}'}(\rho_{\text{in}}^{\mathcal{A}_{\text{in}}\mathcal{B}_{\text{in}}\mathcal{R}} \otimes |0\rangle\langle 0|^{\mathcal{B}^*_{\text{in}}})$ be denoting the joint state when $\tilde{\mathscr{C}}$ reaches Step 6 in the simulation above. Using the triangle inequality for the trace-norm distance using (7) and (8) results in:

$$\Delta \left( \text{tr}_{\mathscr{B}^*} \left( [\tilde{\mathscr{C}} \circledast \mathscr{D}]_{\mathsf{S6}}^{\mathscr{F}}(\rho_{\text{in}}^{\mathcal{A}_{\text{in}}\mathcal{B}_{\text{in}}\mathcal{R}}) \right), [\tilde{\mathscr{A}} \circledast \mathscr{B}]_{s^*}^{\mathscr{O}'}(\rho_{\text{in}}) \right) \leq \mathsf{negl}(n) \ . \quad (9)$$

Equation (9) implies the statement to prove since the interaction between $\tilde{\mathscr{A}}$ and $\mathscr{B}^*$ is resumed once $\tilde{\mathscr{C}}$ transmitted $\tilde{\mathscr{A}}$'s swaddlings back to $\tilde{\mathscr{A}}$. It follows that $\tilde{\mathscr{A}}$'s further actions will produce an output at negligible distance from $[\tilde{\mathscr{A}} \circledast \mathscr{B} \wedge \overline{\text{E-ABORT}}]^{\mathscr{O}'}(\rho_{\text{in}})$ and since $\tilde{\mathscr{C}}$ deletes all of $\mathscr{B}^*$'s registers before getting its output.

To prove the statement for adversary $\tilde{\mathscr{B}}$ interacting with $\mathscr{A}$ in $\widehat{\Pi}_{\mathscr{F}}^{\mathscr{O}'}$ is easier than for adversary $\tilde{\mathscr{A}}$. Remember that for $\tilde{\mathscr{D}}$, $s^*$ is defined to be the turn at which $\tilde{\mathscr{B}}$ receives all his swaddlings from $\mathscr{A}$. In this case, $\tilde{\mathscr{D}}$ always allows $\mathscr{A}$ to get the final state of the computation since the protocol has no lack of fairness for $\mathscr{A}$ (i.e. $f = 1$ always). Apart from this difference, $\tilde{\mathscr{D}}$ proceeds in a very similar way than $\tilde{\mathscr{C}}$. □

## 7 Conclusion

Since Lemmas 6.1 and 6.2 together show that a simulator succeeds in reproducing any real execution, we conclude the active security of the protocol:

**Theorem 7.1 (Active security).** *For any polynomial-time quantum operation* $\mathscr{F} : \mathrm{L}(\mathcal{A}_{\mathrm{in}} \otimes \mathcal{B}_{\mathrm{in}}) \to \mathrm{L}(\mathcal{A}_{\mathrm{out}} \otimes \mathcal{B}_{\mathrm{out}})$*, the two-party hybrid protocol* $\widehat{\Pi}^{\mathscr{O}'}_{\mathscr{F}}$ *is statistically active secure without fairness for Bob.*

Since the simulator establishing Theorem 7.1 is poly-time, the result in [HSS11] allows us to conclude that instantiating all TCP oracles with computationally secure classical protocols preserves security of our protocol, provided the computational assumption is secure against quantum adversaries.

Our construction can be seen as a *compiler* that transforms protocols secure against specious adversaries into protocols for the same task secure against any adversary, in a spirit somewhat similar to Section 7.4 in [Gol04]. The technique we used allows us to force all players to behave speciously. It would be interesting to find out how general is this compiler in the computational setting.

## References


ABE08. D. Aharonov, M. Ben-Or, and E. Eban. Interactive proofs for quantum computations. In *Proceedings of Innovations in Computer Science*, 2008. arxiv.org/abs/0810.5375.

BCG+02. Howard Barnum, Claude Crépeau, Daniel Gottesman, Adam Smith, and Alain Tapp. Authentication of quantum messages. In *43rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 449–458, 2002.

BK05. Sergei Bravyi and Alexei Kitaev. Universal quantum computation with ideal clifford gates and noisy ancillas. *Physical Review A*, 71(022316), 2005. quant-ph/0403025.

BM02. Michael Ben-Or and Dominic Mayers. Quantum universal composability, November 2002. Presentation at "Quantum Information and Cryptography" Workshop, slides online available at http://www.msri.org/publications/ln/msri/2002/quantumcrypto/mayers/1/meta/aux/mayers.pdf.

Can00. Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.

DNS10. Frédéric Dupuis, Jesper Buus Nielsen, and Louis Salvail. Secure two-party quantum evaluation of unitaries against specious adversaries. In Tal Rabin, editor, *CRYPTO*, volume 6223 of *Lecture Notes in Computer Science*, pages 685–706. Springer, 2010.

GC99a. Daniel Gottesman and Isaac L. Chuang. Demonstrating the viability of universal quantum computation using teleportation and single-qubit operations. *Nature*, 402:390–393, November 1999.

GC99b. Daniel Gottesman and Isaac L. Chuang. Quantum teleportation is a universal computational primitive. arxiv.org/abs/quant-ph/9908010, August 1999.

Gol04. Oded Goldreich. *Foundations of Cryptography*, volume II: Basic Applications. Cambridge University Press, 2004.

Got97. Daniel Gottesman. *Stabilizer codes and quantum error correction*. PhD thesis, California Institute of Technology, 1997.

Got10. Daniel Gottesman. An introduction to quantum error correction and fault-tolerant quantum computation. In Jr. 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. arxiv.org/abs/0904.2557.

GW07. 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.

HSS11. Sean Hallgren, Adam Smith, and Fang Song. Classical cryptographic protocols in a quantum world. In Phillip Rogaway, editor, *CRYPTO*, volume 6841 of *Lecture Notes in Computer Science*, pages 411–428. Springer, 2011.

LN11. Carolin Lunemann and Jesper Buus Nielsen. Fully simulatable quantum-secure coin-flipping and applications. In Abderrahmane Nitaj and David Pointcheval, editors, *AFRICACRYPT*, volume 6737 of *Lecture Notes in Computer Science*, pages 21–40. Springer, 2011.

Reg05. Oded 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.

Sho96. Peter W. Shor. Fault-tolerant quantum computation. In *37th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 56–65, 1996.

Unr10. Dominique Unruh. Universally composable quantum multi-party computation. In *Advances in Cryptology—EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 486–505. Springer, 2010.

## A Two-Party Strategies

We define two-party strategies the same way than in [DNS10,GW07]. These strategies may have access to oracles implementing some functionalities, like a communication channel between parties or any other more evolved and non-trivial two-party functionality. The only difference with [DNS10] is that we explicitly allow for $\mathscr{A}$ and $\mathscr{B}$ to apply one final operation after the last oracle call. More precisely,

**Definition A.1.** *A* $m$–turn two party strategy with oracle calls *denoted* $\Pi^{\mathscr{O}} = (\mathscr{A}, \mathscr{B}, \mathscr{O}, m)$ *consists of:*

1. *input spaces* $\mathcal{A}_0$ *and* $\mathcal{B}_0$ *for parties* $\mathscr{A}$ *and* $\mathscr{B}$ *respectively, together with memory spaces* $\mathcal{A}_1, \ldots, \mathcal{A}_{m+1}$ *and* $\mathcal{B}_1, \ldots, \mathcal{B}_{m+1}$ *for* $\mathscr{A}$ *and* $\mathscr{B}$ *respectively. Spaces* $\mathcal{A}_i$ *and* $\mathcal{B}_i$ *can be written as* $\mathcal{A}_i = \mathcal{A}_i^{\mathscr{O}} \otimes \mathcal{A}'_i$ *and* $\mathcal{B}_i = \mathcal{B}_i^{\mathscr{O}} \otimes \mathcal{B}'_i$, $(1 \leq i \leq m)$, *and* $\mathscr{O} = (\mathscr{O}_1, \mathscr{O}_2, \ldots, \mathscr{O}_m)$ *is an* $m$*-tuple of quantum operations:* $\mathscr{O}_i : \mathrm{L}(\mathcal{A}_i^{\mathscr{O}} \otimes \mathcal{B}_i^{\mathscr{O}}) \mapsto \mathrm{L}(\mathcal{A}_i^{\mathscr{O}} \otimes \mathcal{B}_i^{\mathscr{O}})$, $(1 \leq i \leq m)$.
2. *an* $m+1$*-tuple of quantum operations* $(\mathscr{A}_1, \ldots, \mathscr{A}_{m+1})$ *for* $\mathscr{A}$, $\mathscr{A}_i : \mathrm{L}(\mathcal{A}_{i-1}) \mapsto \mathrm{L}(\mathcal{A}_i)$, $(1 \leq i \leq m+1)$,
3. *an* $m+1$*-tuple of quantum operations* $(\mathscr{B}_1, \ldots, \mathscr{B}_{m+1})$ *for* $\mathscr{B}$, $\mathscr{B}_i : \mathrm{L}(\mathcal{B}_{i-1}) \mapsto \mathrm{L}(\mathcal{B}_i)$, $(1 \leq i \leq m+1)$.

## B Discussion of our Security Definition

In extending the model in [DNS10] the following changes have been made.

1. The adversary is no more restricted in how it deviates from the protocol, i.e., it can deviate from the protocol as it wants, except that it has to run in polynomial time.
2. The simulator is now allowed to do an input substitution. This is needed as an active attacker might replace its original input by some alternative input and then run the input honestly. This will be indistinguishable from an honest party running with the alternative input, hence the protocol will and must complete successfully. To simulate this "attack" the simulator must be able to ask the ideal functionality to use the alternative input on behalf of corrupted party.
3. In [DNS10], it was required that each step of the protocol could be simulated. This is needed with passive security, as the fact that the view at the end can be simulated is not the same as the view at an earlier point can be simulated, due to no-cloning (cf., [DNS10]). However, an active adversary is allowed to stop its attack at any intermediary step and output its state at that step. Hence security at the end against all adversary implies security at all intermediary steps against all adversaries.

4. When we compare the outputs of the simulation and the protocol, we will compare the entire view, not just the view of the corrupted party. This is usual in defining active security, as it models privacy *and* correctness. In [DNS10] only the view of the adversary and the simulator was compared, which is sufficient to model privacy, as it shows that whatever the adversary learns can be computed given what it is allowed to learn. When comparing also the output of the honest party in the two settings, we also show that whatever influence the adversary can have on the output of the honest party in the protocol, it could also have on the output of the honest party in the simulation. This captures that basically only input substitution is possible, which is the definition of correctness. Furthermore, privacy and correctness cannot meaningfully be separated into separate notions, which is why we capture them using one definition (cf., the discussion on page 150 in [Can00]).
5. Finally, in [DNS10] a simulator contained an explicit bit $q$ which specified whether it would like to call the ideal functionality or not. This was needed to allow the simulator to preserve its original input. An active adversary has the power to just run the ideal functionality on a dummy input and move its original input to an ancilla if it so desires, so we can remove the explicit bit $q$.

Our model is reminiscent of previous models of quantum secure two-party computation, but as opposed to the UC models from [BM02,Unr10,HSS11] our model only guarantees sequential security. That said, since our simulators do not access the code of the adversary or rewind the adversary, our protocol might very well be UC secure in one of the models from [BM02,Unr10,HSS11]. However, here we want to focus on what we consider the main novelty, namely that actively secure 2-party quantum computation is possible at all, and how, and we prefer to avoid cluttering that presentation by dealing with the connoisseur details needed to obtain and prove UC security. We think this is best done in the above simple model.

## C Security of the Clifford-Based QAS

For completeness, we provide the proof of security of the Clifford-based authentication scheme of [ABE08]:

**Theorem C.1 (Security of Clifford-based QAS).** *The Clifford-based QAS defined in Definition 2.3 is $\varepsilon$-secure for $\varepsilon = 6 \times 2^{-n}$.*

*Proof.* Let $U_{CR}$ be the attack unitary, and let us decompose it into the Pauli basis on $C$:

$$U_{CR} = \sum_P P_C \otimes U_R^P \ .$$

We also define the following:

$$\begin{aligned} \mathscr{U}^{\mathrm{acc}}_{R\to R}(\rho) &= U_R^{\mathbb{1}} \rho {U_R^{\mathbb{1}}}^\dagger \\ \mathscr{U}^{\mathrm{rej}}_{R\to R}(\rho) &= \sum_{P\neq\mathbb{1}} U_R^P \rho {U_R^P}^\dagger \\ \Omega_S &= \pi_S \ . \end{aligned}$$

We now compute the state given that the decoder accepts:

$$
\begin{aligned}
&P_{\mathrm{acc}}\mathbb{E}_k\mathscr{D}_k\left(U_{CR}\mathscr{E}_k(\psi_{SR})U_{CR}^\dagger\right)P_{\mathrm{acc}}\\
&=\mathbb{E}_k\sum_{P,P'}\mathrm{tr}_A\left[P_{\mathrm{acc}}C_k^\dagger(P_C\otimes U_R^P)C_k(\psi_{SR}\otimes|0^n\rangle\langle 0^n|_A)C_k^\dagger({P'_C}^\dagger\otimes {U_R^{P'}}^\dagger)C_kP_{\mathrm{acc}}\right]\\
&\overset{(a)}{=}\mathbb{E}_k\sum_{P}\mathrm{tr}_A\left[P_{\mathrm{acc}}C_k^\dagger(P_C\otimes U_R^P)C_k(\psi_{SR}\otimes|0^n\rangle\langle 0^n|_A)C_k^\dagger({P_C}^\dagger\otimes {U_R^{P}}^\dagger)C_kP_{\mathrm{acc}}\right]\\
&=U_R^{\mathbb{1}}\psi_{SR}{U_R^{\mathbb{1}}}^\dagger+\mathbb{E}_k\sum_{P\neq\mathbb{1}}\mathrm{tr}_A\left[P_{\mathrm{acc}}C_k^\dagger(P_C\otimes U_R^P)C_k(\psi_{SR}\otimes|0^n\rangle\langle 0^n|_A)C_k^\dagger({P_C}^\dagger\otimes {U_R^{P}}^\dagger)C_kP_{\mathrm{acc}}\right]\\
&=\mathscr{U}^{\mathrm{acc}}(\psi_{SR})+\sum_{P\neq\mathbb{1}}\mathbb{E}_{\tilde{P}\neq\mathbb{1}}\mathrm{tr}_A\left[P_{\mathrm{acc}}(\tilde{P}_C\otimes U_R^P)(\psi_{SR}\otimes|0^n\rangle\langle 0^n|_A)(\tilde{P}_C^\dagger\otimes {U_R^{P}}^\dagger)P_{\mathrm{acc}}\right]\\
&=\mathscr{U}^{\mathrm{acc}}(\psi_{SR})+\frac{2^{-n}|\mathfrak{P}|}{|\mathfrak{P}|-1}\sum_{P\neq\mathbb{1}}U_R^P(\pi_S\otimes\psi_R){U_R^P}^\dagger-\frac{1}{|\mathfrak{P}|-1}\sum_{P\neq\mathbb{1}}U_R^P\psi_{SR}{U_R^P}^\dagger\\
&=\mathscr{U}^{\mathrm{acc}}(\psi_{SR})+\frac{2^{-a}|\mathfrak{P}|}{|\mathfrak{P}|-1}\mathscr{U}^{\mathrm{rej}}(\psi_R)\otimes\pi_S-\frac{1}{|\mathfrak{P}|-1}\mathscr{U}^{\mathrm{rej}}(\psi_{SR})\ ,
\end{aligned}
$$

where $(a)$ comes from Lemma B.3 in [ABE08], and where $\mathfrak{P}$ denotes the set of all Pauli operators on $n$ qubits. Likewise, the state given that the decoder rejects is given by:

$$
\begin{aligned}
&P_{\mathrm{rej}}\mathbb{E}_k\mathscr{D}_k\left(U_{CR}\mathscr{E}_k(\psi_{SR})U_{CR}^\dagger\right)P_{\mathrm{rej}}\\
&\qquad=\mathbb{E}_k\sum_{P,P'}\mathrm{tr}_A\left[P_{\mathrm{rej}}C_k^\dagger(P_C\otimes U_R^R)C_k(\psi_{SR}\otimes|0^n\rangle\langle 0^n|_A)C_k^\dagger({P'_C}^\dagger\otimes {U_R^{P'}}^\dagger)C_kP_{\mathrm{rej}}\right]\\
&\qquad\overset{(a)}{=}\mathbb{E}_k\sum_{P}\mathrm{tr}_A\left[P_{\mathrm{rej}}C_k^\dagger(P_C\otimes U_R^R)C_k(\psi_{SR}\otimes|0^n\rangle\langle 0^n|_A)C_k^\dagger({P_C}^\dagger\otimes {U_R^{P}}^\dagger)C_kP_{\mathrm{rej}}\right]\\
&\qquad=\mathbb{E}_k\sum_{P\neq\mathbb{1}}\mathrm{tr}_A\left[P_{\mathrm{rej}}C_k^\dagger(P_C\otimes U_R^P)C_k(\psi_{SR}\otimes|0^n\rangle\langle 0^n|_A)C_k^\dagger({P_C}^\dagger\otimes {U_R^{P}}^\dagger)C_kP_{\mathrm{rej}}\right]\\
&\qquad=\sum_{P\neq\mathbb{1}}\mathbb{E}_{\tilde{P}\neq\mathbb{1}}\mathrm{tr}_A\left[P_{\mathrm{rej}}(\tilde{P}_C\otimes U_R^P)(\psi_{SR}\otimes|0^n\rangle\langle 0^n|_A)(\tilde{P}_C^\dagger\otimes {U_R^{P}}^\dagger)P_{\mathrm{rej}}\right]\\
&\qquad=\frac{(1-2^{-n})|\mathfrak{P}|}{|\mathfrak{P}|-1}\sum_{P\neq\mathbb{1}}U_R^P(\pi_S\otimes\psi_R){U_R^P}^\dagger-\frac{1}{|\mathfrak{P}|-1}\sum_{P\neq\mathbb{1}}U_R^P\psi_{SR}{U_R^P}^\dagger\\
&\qquad=\frac{(1-2^{-n})|\mathfrak{P}|}{|\mathfrak{P}|-1}\mathscr{U}^{\mathrm{rej}}(\psi_R)\otimes\pi_S-\frac{1}{|\mathfrak{P}|-1}\mathscr{U}^{\mathrm{rej}}(\psi_{SR})\\
&\qquad=\mathscr{U}^{\mathrm{rej}}(\psi_R)\otimes\pi_S-\frac{2^{-a}|\mathfrak{P}|-1}{|\mathfrak{P}|-1}\mathscr{U}^{\mathrm{rej}}(\psi_R)\otimes\pi_S-\frac{1}{|\mathfrak{P}|-1}\mathscr{U}^{\mathrm{rej}}(\psi_{SR})\ .
\end{aligned}
$$

Now, it is easy to see that the trace distance in Definition 2.2 can be upper-bounded by:

$$
\mathrm{tr}\left[2\times\frac{2^{-n}|\mathfrak{P}|}{|\mathfrak{P}|-1}\mathscr{U}^{\mathrm{rej}}(\psi_R)\otimes\pi_S+\frac{2}{|\mathfrak{P}|-1}\mathscr{U}^{\mathrm{rej}}(\psi_{SR})\right]\leqslant 2\times\frac{2^{-n}|\mathfrak{P}|}{|\mathfrak{P}|-1}+\frac{2}{|\mathfrak{P}|-1}
$$

$$= 2 \times \frac{2^{-n}4^{n+s}}{4^{n+s}-1} + \frac{2}{4^{n+s}-1}$$
$$\leqslant 6 \times 2^{-n} \ .$$

□

## D R-gate teleportation and magic state distillation

The gate teleportation protocol for the R-gate works as follows. Suppose one wants to execute R on wire w. One would need to initialize a wire m in the *magic state* $|M\rangle := \frac{1}{\sqrt{2}}\left(|0\rangle + e^{i\pi/4}|1\rangle\right)$. The protocol then consists of a CNOT from m to w, and then of a measurement of w in the computational basis. If the result is 0, one does nothing, and if the result is 1, we apply the gate $A := e^{-i\pi/4}PX$ to m. We then rename m to w, and one can check that an R-gate has been applied to w.

We therefore need a supply of magic states for our protocol. However, we need some way of generating them securely. For this, we turn to ideas from fault-tolerant computation, more specifically from [BK05]. There, they consider a model of computation in which we can perform perfect Clifford gates, but where we only have a supply of noisy magic states. We therefore need to distill good magic states from noisy ones. We will use this technique to ensure that we have a supply of good magic states: we will first let Alice prepare a large supply of magic states, and then we will have Bob distill a smaller number of states from this untrusted supply. Bob will first test a random sample of the states to make sure that the error rate is sufficiently low, and he will then use the distillation protocol from [BK05] to get the final states.

Here we will simply state the protocol from [BK05] without attempting to explain how and why it works. The protocol is based on a CSS code which is constructed from two linear classical codes $\mathcal{L}_1$ and $\mathcal{L}_2$, whose generator matrices are given in Figure D.1 along with the stabilizer generators of the resulting CSS codes.

Here's how one step of the magic state distillation works:

1. Dephase all qubits (apply $A$ to each qubit with probability $1/2$).
2. Randomly permute all the qubits.
3. Measure $\mu$ (the syndrome of $\mathcal{L}_2$).
4. Compute the vector $w$ as $w = \sum_g \mu(g)g$, where the sum ranges over stabilizer generators of $\mathcal{L}_2$.
5. Apply $A(w)^\dagger$.
6. Measure $\eta$ (syndrome of $\mathcal{L}_1$ with $X$'s)
7. If $\eta \neq 0$, declare an error.
8. Apply the decoding operation for the $X/Z$ CSS code.

It can then be shown that, if the input has an error rate of $\varepsilon$, then the output error rate is $\varepsilon_{\text{out}} \leqslant 40\varepsilon^3$ whenever $\varepsilon \leqslant 0.041$ (see [BK05], Equation (38)). This means that after repeating this recursively $\ell$ times, we have a final error rate of $\varepsilon_f \leqslant \frac{1}{\sqrt{40}}\left(\sqrt{40}\varepsilon\right)^{3^\ell}$.

We now need to ensure that, if the actual error rate was above $0.041$, then the adversary would have been caught with overwhelming probability. Note that since the alleged magic states were dephased in the $A$-basis, this corresponds to a purely classical sampling problem: if one has a set

$$\begin{bmatrix} 1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0 \\ 0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&\\ 0&0&0&1&1&1&1&0&0&0&0&1&1&1&1&\\ 0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&1 \end{bmatrix}$$

(a) Generator matrix of code $\mathcal{L}_1$

$$\begin{bmatrix} 0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&\\ 1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&\\ 1&1&1&0&0&0&0&1&1&1&1&0&0&0&0&\\ 1&1&1&1&1&1&1&0&0&0&0&0&0&0&0&\\ 1&1&0&1&1&1&0&1&1&1&0&1&1&1&0&\\ 1&1&1&1&0&1&0&1&1&1&1&1&0&1&0&\\ 1&1&1&1&1&1&1&1&0&1&0&1&0&1&0&\\ 1&1&1&1&1&0&0&1&1&1&1&1&1&0&0&\\ 1&1&1&1&1&1&1&1&1&0&0&1&1&0&0&\\ 1&1&1&1&1&1&1&1&1&1&1&0&0&0&0&1 \end{bmatrix}$$

(b) Generator matrix of codde $\mathcal{L}_2$

$$\begin{bmatrix} X&I&X&I&X&I&X&I&X&I&X&I&X&I&X \\ I&X&X&I&I&X&X&I&I&X&X&I&I&X&X \\ I&I&I&X&X&X&X&I&I&I&I&X&X&X&X \\ I&I&I&I&I&I&I&X&X&X&X&X&X&X&X \\ Z&I&Z&I&Z&I&Z&I&Z&I&Z&I&Z&I&Z \\ I&Z&Z&I&I&Z&Z&I&I&Z&Z&I&I&Z&Z \\ I&I&I&Z&Z&Z&Z&I&I&I&I&Z&Z&Z&Z \\ I&I&I&I&I&I&I&Z&Z&Z&Z&Z&Z&Z&Z \\ I&I&Z&I&I&I&Z&I&I&I&Z&I&I&I&Z \\ I&I&I&I&Z&I&Z&I&I&I&I&I&Z&I&Z \\ I&I&I&I&I&I&I&I&Z&I&Z&I&Z&I&Z \\ I&I&I&I&I&Z&Z&I&I&I&I&I&I&Z&Z \\ I&I&I&I&I&I&I&I&I&Z&Z&I&I&Z&Z \\ I&I&I&I&I&I&I&I&I&I&I&Z&Z&Z&Z \end{bmatrix}$$

(c) Stabilizer generators of the CSS code used for magic state distillation

Fig. D.1: Description of the code used for magic state distillation.

of $n$ states, with a fraction of at least $0.041$ being faulty, what is the probability $p$ that sampling $k$ of them reveals only good states? This can be calculated as follows:

$$p \;=\; \binom{0.959n}{k}\binom{n}{k}^{-1} \;\leqslant\; \frac{e^{0.959nh(k/0.959n)}}{\frac{1}{n+1}e^{nh(k/n)}} \;=\; (n+1)e^{n(0.959h(k/0.959n)-h(k/n))}\,.$$

Now, it can be shown that the right-hand side is upper-bounded by $(n+1)e^{-0.1n}$ whenever $k/n \geqslant 0.8$.

## E Description of the subprotocols

The ancillas swaddled during `Initialization` also need to have their initial state verified. We now describe the verification procedure `VerifyAncilla` that verifies that a swaddled ancilla is indeed in state $|0\rangle$. This subprotocol relies on the ideal functionality $\mathsf{id}_{\text{INTCNOT}}$ defined as follows:

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

where $t$ is the description of a Clifford $T \in \mathfrak{C}_{2n+1}$ such that

$$T = \begin{cases} B'(A'\otimes \mathbb{1}_n)C(A^\dagger \otimes \mathbb{1}_n)B^\dagger & \text{if Alice holds the inner key,} \\ A'(\mathbb{1}_n \otimes B')C(\mathbb{1}_n \otimes B^\dagger)A^\dagger & \text{if Alice holds the outer key,} \end{cases}$$

where $C$ is a CNOT from the data qubit as control and Bob's first dummy qubit as target, and $A'$ and $B'$ are Cliffords of the appropriate size chosen uniformly at random.

`VerifyAncilla`$(\mathfrak{s}(\mathtt{w}))$**, with** $\mathtt{w} \in \mathcal{A}$**:**
1. Alice sends $\mathfrak{s}(\mathtt{w})$ to Bob.
2. Bob runs `TestSwaddling`$(\mathfrak{s}(\mathtt{w}))$.
3. Alice and Bob call $(K'_{\mathtt{w},a},(t,K'_{\mathtt{w},b})) \leftarrow \mathsf{id}_{\text{INTCNOT}}((id_A(\mathtt{w}),id_B(\mathtt{w})),(id_A(\mathtt{w}),id_B(\mathtt{w})))$. Bob executes $T$ on $\mathfrak{s}(\mathtt{w})$.
4. Bob sends $\mathfrak{s}(\mathtt{w})$ back to Alice.
5. Alice runs `TestSwaddling`$(\mathfrak{s}(\mathtt{w}))$ and sends $\mathfrak{s}(\mathtt{w})$ to Bob.
6. Bob runs `TestSwaddling`$(\mathfrak{s}(\mathtt{w}))$.
7. Bob sends $\mathfrak{s}(\mathtt{w})$ back to Alice.
8. Alice runs `TestSwaddling`$(\mathfrak{s}(\mathtt{w}))$.

In the above, the idea is to measure the swaddled state in the computational basis by first executing a CNOT with the swaddled state as the control and one of Bob's dummies as the target. Bob then performs the measurement implicitly when he tests his dummies in step 6.

The next procedure implements the magic state distillation outlined in Appendix D.

`DistillMagic`$(\mathfrak{s}(\mathtt{w}))$**, with** $\mathtt{w} \in \mathcal{A}$**:**
1. Alice creates $N$ qubits in state $|M\rangle$.
2. Swaddle all of them.
3. Apply $A$ to each qubit with probability $1/2$.
4. Measure $\mu$ (the syndrome of $\mathcal{L}_2$).
5. Compute the vector $w$ as $w = \sum_g \mu(g)g$, where the sum ranges over stabilizer generators of $\mathcal{L}_2$.
6. Apply $A(w)^\dagger$ to the qubits.
7. Measure $\eta$ (syndrome of $\mathcal{L}_1$ with $X$'s).
8. If $\eta \neq 0$, declare an error.
9. Apply the decoding operation for the $X/Z$ CSS code.

In the above, the measurement are done by calling the subprotocol `Measure`, given further down.

`RGate`($\mathfrak{s}(\mathtt{w})$) **with** $\mathtt{w} \in \mathcal{A}$**:**
1. Alice performs a swaddled CNOT with $\mathtt{m}_w$ as a control, and $\mathtt{w}$ as target.
2. Alice sends $\mathfrak{s}(\mathtt{w})$ to Bob. Bob runs `TestSwaddling`($\mathfrak{s}(\mathtt{w})$) on it.
3. Alice runs `Measure`($\mathfrak{s}(\mathtt{w})$).
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 $\mathtt{m}_{a,i}$ to $\mathtt{w}$.

To see that such a procedure indeed performs an R-gate, we will analyze its behavior when the $\mathtt{w}$ is in some arbitrary pure state $\alpha|0\rangle + \beta|1\rangle$; hence, if we include the magic state, the initial state here is

$$\frac{1}{\sqrt{2}}(|0\rangle + e^{i\pi/4}|1\rangle)(\alpha|0\rangle + \beta|1\rangle) \ .$$

After the CNOT, we get

$$\frac{1}{\sqrt{2}}(\alpha|00\rangle + \beta|01\rangle + e^{i\pi/4}\alpha|11\rangle + e^{i\pi/4}\beta|10\rangle) \ .$$

Measuring the second qubit then yields either $\alpha|0\rangle + e^{i\pi/4}\beta|1\rangle$ if the measurement result was 0, and $\beta|0\rangle + e^{i\pi/4}\alpha|1\rangle$ if the result was 1. This is the desired result in the first case, and performing $e^{i\pi/4}XP^\dagger$ in the second case also yields the same state.

We now know how to perform all of these operations on swaddled states; the only difficulty is to ensure that Alice doesn't lie about the measurement result. Hence, we first copy the measurement result into another swaddled register that Bob gets, ensuring that he also gets the measurement result.

`Measure`($\mathfrak{s}(\mathtt{w})$)**, with** $\mathtt{w} \in \mathcal{A}$**:**
1. Let $\mathtt{a}$ be an ancilla in state $|0\rangle$ prepared during `Initialization`, and let Alice call `CNOT`($\mathfrak{s}(\mathtt{w}), \mathfrak{s}(\mathtt{a})$).
2. Alice sends $\mathfrak{s}(\mathtt{a})$ to Bob.
3. Bob runs `TestDummies`($\mathfrak{s}(\mathtt{a})$).
4. Alice and Bob call $\mathsf{id}_{\text{OPEN}}(id_B(\mathtt{w}), id_B(\mathtt{w}))$ and $\mathsf{id}_{\text{OPEN}}(id_A(\mathtt{a}), id_A(\mathtt{a}))$.
5. Alice decrypts $\mathtt{w}$ and measures it in the computational basis, and Bob decrypts $\mathtt{a}$ and measures it in the computational basis.

This subprotocol measures $\mathtt{w}$ in the computational basis and ensures that both Alice and Bob get the result.

## F Additional definitions

**Definition F.1 (Inner swaddling operation).** *We define $\mathsf{C}_{I,a,\mathtt{w}}$, Alice's inner swaddling operation for wire $\mathtt{w}$, as follows:*

$$\mathsf{C}_{I,a,\mathtt{w}}(\rho) := \sum_{C_a \in \mathfrak{C}(k+1)} \left(|k_a\rangle\langle k_a|^{\mathcal{K}_a} \otimes C_{k_a}^{\mathfrak{s}(\mathtt{w})}\right) \rho \left(|k_a\rangle\langle k_a|^{\mathcal{K}_a} \otimes C_{k_a}^{\mathfrak{s}(\mathtt{w})^\dagger}\right),$$

*where $k_a$ is a classical description of $C_a$. Likewise, $\mathsf{C}_{I,b,\mathtt{w}}$ is Bob's inner swaddling operation and is defined the same way.*

**Definition F.2 (Outer swaddling operation).** *We define $\mathsf{C}_{O,a,\mathtt{w}}$, Alice's outer swaddling operation for wire $\mathtt{w}$, as follows:*

$$\mathsf{C}_{O,a,\mathtt{w}}(\rho) := \sum_{C_a \in \mathfrak{C}(2k+1)} \left( |k_a\rangle\langle k_a|^{\mathcal{K}_a} \otimes C_{k_a}^{\mathfrak{s}(\mathtt{w})} \right) \rho \left( |k_a\rangle\langle k_a|^{\mathcal{K}_a} \otimes C_{k_a}^{\mathfrak{s}(\mathtt{w})^\dagger} \right),$$

*where $k_a$ is a classical description of $C_a$. Likewise, $\mathsf{C}_{O,b,\mathtt{w}}$ is Bob's outer swaddling operation and is defined the same way.*

**Definition F.3 (Swaddling pattern).** *For every turn $s$ of the protocol, let $\mathfrak{A}_{I,s}$ (resp. $\mathfrak{B}_{I,s}$) be the set of wires for which Alice (resp. Bob) holds the inner key, and $\mathfrak{A}_{O,s}$ (resp. $\mathfrak{B}_{O,s}$) be the set of wires for which Alice holds the outer key. Wires that are not swaddled yet are in none of these sets, and wires which only have an inner swaddling are only in the corresponding inner set. We call these sets the* swaddling pattern *at turn $s$.*

**Definition F.4.** *Let $\mathsf{C}_s$ be the operation that takes all the unswaddled wires and swaddles according to the swaddling pattern at turn $s$:*

$$\mathsf{C}_s := \left[ \left( \bigotimes_{\mathtt{w} \in \mathfrak{A}_{O,s}} \mathsf{C}_{O,a,\mathtt{w}} \right) \otimes \left( \bigotimes_{\mathtt{w} \in \mathfrak{B}_{O,s}} \mathsf{C}_{O,b,\mathtt{w}} \right) \right] \circ \left[ \left( \bigotimes_{\mathtt{w} \in \mathfrak{A}_{I,s}} \mathsf{C}_{I,a,\mathtt{w}} \right) \otimes \left( \bigotimes_{\mathtt{w} \in \mathfrak{B}_{I,s}} \mathsf{C}_{I,b,\mathtt{w}} \right) \right].$$

*Furthermore, we define Alice's total swaddling operation:*

$$\mathsf{C}_{a,s} := \left( \bigotimes_{\mathtt{w} \in \mathfrak{A}_{O,s}} \mathsf{C}_{O,a,\mathtt{w}} \right) \circ \left( \bigotimes_{\mathtt{w} \in \mathfrak{A}_{I,s}} \mathsf{C}_{I,a,\mathtt{w}} \right),$$

*and likewise for Bob's total swaddling operation $\mathsf{C}_{b,s}$.*

## G Proving forcingness

In the following, we will not give explicit bounds for the various negligible functions, and assume that states that have negligible trace distance to each other are equal. Explicit bounds can be derived from Theorem 2.4, which gives an upper bound on the error of the the Clifford-based QAS.

**Lemma G.1.** `Initialization` *is $\mathscr{G}$-forcing for any $\tilde{\mathscr{A}}$ and any $\tilde{\mathscr{B}}$, where $\mathscr{G}$ tests the ancillas in the $|0\rangle$ state and distills magic states (see Lemmas G.3 and G.2).*

*Proof.* We first prove forcingness against $\tilde{\mathscr{A}}$. Let the state just before Alice sends her qubits to Bob in the first call to `Swaddle` be

$$\rho_1^{\mathcal{ABR}} = \tilde{\mathscr{E}}_0 \left( \rho_0^{\mathcal{ABR}} \right),$$

for some operation $\tilde{\mathscr{E}}_0$ on Alice's systems. At this point, Alice commits to a Clifford $C_A$, sends $n+1$ qubits to Bob and receives $\mathfrak{s}(\mathtt{w})$ from Bob. Alice then gets the qubits back from Bob, along with more qubits in the second call to `Swaddle`. At this point, the global state is

$$\rho_2^{\mathcal{ABR}} = \mathsf{C}_{b,s} \circ \tilde{\mathscr{E}}_0 \left( \rho_0^{\mathcal{ABR}} \right).$$

She will then send back some of these qubits to Bob. Now, since we are looking at the case where there is no abortion before the end, the fact that she passes Bob's `TestDummies` means that her attack can be represented by a CP map $\mathscr{U}^{\mathrm{acc}}$ on her qubits that she is not sending, as in Definition 2.2. The state after sending the qubits back to Bob is therefore

$$\rho_3^{\mathcal{ABR}} = \mathscr{U}^{\mathrm{acc}} \circ \mathsf{C}_{b,s} \circ \tilde{\mathscr{E}}_0 \left(\rho_0^{\mathcal{ABR}}\right).$$

and the protocol is therefore $\mathbb{1}$-forcing for $\tilde{\mathscr{A}}$ so far. The rest of the protocol only involves calls to other subprotocols which are proven to be forcing later on.

The proof of forcingness against $\tilde{\mathscr{B}}$ follows the same argument. □

**Lemma G.2.** `VerifyAncilla(`$\mathfrak{s}$`(`$\mathtt{w}$`))` *is $\mathscr{G}$-forcing for any $\tilde{\mathscr{A}}$ and any $\tilde{\mathscr{B}}$, where $\mathscr{G}(\cdot) = |0\rangle\langle 0| \cdot |0\rangle\langle 0|$.*

*Proof.* We first prove forcingness against $\tilde{\mathscr{A}}$. Assume that the state at the beginning of the protocol (in the no-early-abort case) is

$$\rho^{\mathcal{ABR}} = \tilde{\mathscr{E}} \circ \mathsf{C}_{b,s} \circ \mathscr{G}_0 \circ \tilde{\mathscr{E}}_0 \left(\rho_0^{\mathcal{ABR}}\right).$$

Since the state passes Bob's `TestDummies` in step 2, one can assume that $\tilde{\mathscr{A}}$'s attack in step 1 can be represented by some $\mathscr{U}_1^{\mathrm{acc}}$ on her systems. Furthermore, Bob implemented a measurement of the logical qubit in the computational basis, via his own dummy: since his dummy was initialized to $|0\rangle$ the CNOT that was performed from the logical qubit to the dummy, followed by measuring all his dummies in the computational basis means that the logical qubit was also measured. Since for forcingness, we only consider the case in which all `TestDummies` are passed, it follows that the operation $\mathscr{G}$ defined above was applied. Again, in step 5, Alice must have sent something that passes Bob's `TestDummies`, so her operation can be represented by some $\mathscr{U}_2^{\mathrm{acc}}$ on her systems. Finally, in step 8, she can again do an arbitrary operation $\tilde{\mathscr{E}}_f$ on her systems. Hence, the final state looks like

$$\rho_f^{\mathcal{ABR}} = \tilde{\mathscr{E}}' \circ \mathsf{C}_{b,s} \circ \mathscr{G} \circ \mathscr{G}_0 \circ \tilde{\mathscr{E}}_0 \left(\rho_0^{\mathcal{ABR}}\right),$$

where $\tilde{\mathscr{E}}' := \tilde{\mathscr{E}}_f \circ \mathscr{U}_2^{\mathrm{acc}} \circ \mathscr{U}_1^{\mathrm{acc}} \circ \tilde{\mathscr{E}}$.

We now prove forcingness against $\tilde{\mathscr{B}}$. Assume that the state at the beginning is

$$\rho^{\mathcal{ABR}} = \tilde{\mathscr{E}} \circ \mathsf{C}_{a,s} \circ \mathscr{G}_0 \circ \tilde{\mathscr{E}}_0 \left(\rho_0^{\mathcal{ABR}}\right).$$

Since Bob passes the `TestDummies` in step 5, his attack must correspond to a $\mathscr{U}_1^{\mathrm{acc}}$ on his other systems; likewise at step 8 with $\mathscr{U}_2^{\mathrm{acc}}$. The final state can therefore be expressed as

$$\rho_f^{\mathcal{ABR}} = \tilde{\mathscr{E}}' \circ \mathsf{C}_{a,s} \circ \mathscr{G}_0 \circ \tilde{\mathscr{E}}_0 \left(\rho_0^{\mathcal{ABR}}\right),$$

where $\tilde{\mathscr{E}}' := \mathscr{U}_2^{\mathrm{acc}} \circ \mathscr{U}_1^{\mathrm{acc}} \circ \tilde{\mathscr{E}}$. Finally, since Alice is honest, she has indeed prepared the ancilla in state $|0\rangle$, so adding $\mathscr{G}$ in the above expression leaves it invariant, and we have

$$\rho_f^{\mathcal{ABR}} = \tilde{\mathscr{E}}' \circ \mathsf{C}_{a,s} \circ \mathscr{G} \circ \mathscr{G}_0 \circ \tilde{\mathscr{E}}_0 \left(\rho_0^{\mathcal{ABR}}\right)$$

as the final states. □

**Lemma G.3.** $\texttt{DistillMagic}(\mathfrak{s}(\mathtt{w}))$ *is $\mathscr{G}$-forcing for any $\tilde{\mathscr{A}}$ and any $\tilde{\mathscr{B}}$, where $\mathscr{G}$ is the magic state distillation protocol described in Appendix D.*

*Proof.* The distillation protocol is simply a combination of other subprotocols, so the forcingness of the subprotocols used guarantee the forcingness of the distillation protocol. □

**Lemma G.4.** $\texttt{CNOT}(\mathfrak{s}(\mathtt{w_c}), \mathfrak{s}(\mathtt{w_t}))$ *with $\mathtt{w}_c$ at Alice, is $\mathscr{G}$-forcing for any $\tilde{\mathscr{A}}$ and any $\tilde{\mathscr{B}}$, where $\mathscr{G}$ is a CNOT on the appropriate wires.*

*Proof.* We start by proving forcingness for $\tilde{\mathscr{A}}$. Assume that the state at the beginning of the protocol (in the no-early-abort case) is

$$\rho^{\mathcal{ABR}} = \tilde{\mathscr{E}} \circ \mathsf{C}_{b,s} \circ \mathscr{G}_0 \circ \tilde{\mathscr{E}}_0 \left(\rho_0^{\mathcal{ABR}}\right).$$

Since the state passes Bob's joint `TestDummies` in step 3, one can assume that $\tilde{\mathscr{A}}$'s attack in step 2 can be represented by some $\mathscr{U}^{\mathrm{acc}}$ on her systems. Bob then applies the CNOT honestly, so it follows that the operation $\mathscr{G}$ defined above was applied. Finally, in step 8, she can again do an arbitrary operation $\tilde{\mathscr{E}}_f$ on her systems. Hence, the final state looks like

$$\rho_f^{\mathcal{ABR}} = \tilde{\mathscr{E}}' \circ \mathsf{C}_{b,s} \circ \mathscr{G} \circ \mathscr{G}_0 \circ \tilde{\mathscr{E}}_0 \left(\rho_0^{\mathcal{ABR}}\right),$$

where $\tilde{\mathscr{E}}' := \tilde{\mathscr{E}}_f \circ \mathscr{U}^{\mathrm{acc}} \circ \tilde{\mathscr{E}}$.

We now prove forcingness for $\tilde{\mathscr{B}}$; we will suppose that Bob holds $\mathtt{w}_t$, if not one can easily see that the proof is a slightly simpler version of the same argument. Assume that the state at the beginning of the protocol is

$$\rho^{\mathcal{ABR}} = \tilde{\mathscr{E}} \circ \mathsf{C}_{a,s} \circ \mathscr{G}_0 \circ \tilde{\mathscr{E}}_0 \left(\rho_0^{\mathcal{ABR}}\right).$$

Since the state passes Alice's `TestDummies` in step 1, one can assume that $\tilde{\mathscr{B}}$'s attack in step 2 can be represented by some $\mathscr{U}_1^{\mathrm{acc}}$ on his systems. Alice then sends $\mathfrak{s}(\mathtt{w}_c)$ and $\mathfrak{s}(\mathtt{w}_c)$ to Bob, and since Bob passes `TestDummies` in step 5, one can represent his attack by some $\mathscr{U}_2^{\mathrm{acc}}$ on his systems. Hence, the final state looks like

$$\rho_f^{\mathcal{ABR}} = \tilde{\mathscr{E}}' \circ \mathsf{C}_{a,s} \circ \mathscr{G} \circ \mathscr{G}_0 \circ \tilde{\mathscr{E}}_0 \left(\rho_0^{\mathcal{ABR}}\right),$$

where $\tilde{\mathscr{E}}' := \mathscr{U}_2^{\mathrm{acc}} \circ \mathscr{U}_1^{\mathrm{acc}} \circ \tilde{\mathscr{E}}$. □

**Lemma G.5.** $\texttt{Measure}(\mathfrak{s}(\mathtt{w}))$*, with $\mathtt{w}$ at Alice, is $\mathscr{G}$-forcing for any $\tilde{\mathscr{A}}$ and any $\tilde{\mathscr{B}}$, where $\mathscr{G}$ performs a measurement in the computational basis, and stores the output in registers in both $\mathcal{A}$ and $\mathcal{B}$.*

*Proof.* Since `CNOT` is forcing, Alice must have actually performed it, and since Bob runs `TestDummies` on $\mathfrak{s}(\mathtt{a})$, Alice must actually send it. Finally, once they measure, the honest party's measurement ensures that the adversary's state also collapses, so that the register only contains the classical measurement result. The same holds for Bob. □

**Lemma G.6.** $\texttt{RGate}(\mathfrak{s}(\mathtt{w}))$ *with $\mathtt{w}$ at Alice, is $\mathscr{G}$-forcing for any $\tilde{\mathscr{A}}$ and any $\tilde{\mathscr{B}}$, where $\mathscr{G}(\cdot) := R \cdot R^\dagger$ on $\mathtt{w}$.*

*Proof.* This subprotocol is simply a composition of other subprotocols that have already been shown to be forcing. □

**Lemma G.7.** $\texttt{OneQubitClifford}(\texttt{C}, \mathfrak{s}(\texttt{w}))$ *with* $\texttt{w}$ *at Alice, is* $\mathscr{G}$*-forcing for any* $\tilde{\mathscr{A}}$ *and any* $\tilde{\mathscr{B}}$*, where* $\mathscr{G}(\cdot) := C \cdot C^\dagger$ *on* $\texttt{w}$*.* □

*Proof.* Suppose that the state at the beginning in the no-early-abort case is

$$\rho^{\mathcal{ABR}} = \tilde{\mathscr{E}} \circ \mathsf{C}_{b,s} \circ \mathscr{G}_0 \circ \tilde{\mathscr{E}}_0 \left(\rho_0^{\mathcal{ABR}}\right).$$

Since this subprotocol doesn't actually involve Bob at all, we can simply let $\tilde{\mathscr{E}}_f$ be the operation applied by Alice on her qubits, and define $\tilde{\mathscr{E}}' := \tilde{\mathscr{E}}_f \circ \tilde{\mathscr{E}}$. The final state is therefore

$$\rho_f^{\mathcal{ABR}} = \tilde{\mathscr{E}}' \circ \mathsf{C}_{b,s} \circ \mathscr{G} \circ \mathscr{G}_0 \circ \tilde{\mathscr{E}}_0 \left(\rho_0^{\mathcal{ABR}}\right).$$

The same reasoning applies to $\tilde{\mathscr{B}}$. □