Concurrent composition in the bounded quantum storage model

Dominique Unruh

Saarland University

Abstract. We define the BQS-UC model, a variant of the UC model, that deals with protocols in the bounded quantum storage model. We present a statistically secure commitment protocol in the BQS-UC model that composes concurrently with other protocols and an (a-priori) polynomially-bounded number of instances of itself. Our protocol has an efficient simulator which is important if one wishes to compose our protocol with protocols that are only computationally secure. Combining our result with prior results, we get a statistically BQS-UC secure constant-round protocol for general two-party computation without the need for any setup assumption.

Keywords: Bounded quantum storage, composability, two-party computation.

1 Introduction

Since the inception of quantum key distribution by Bennett and Brassard [2], it has been known that quantum communication permits to achieve protocol tasks that are impossible given only a classical channel. For example, a quantum key distribution scheme [2] permits to agree on a secret key that is statistically secret, using only an authenticated but not secret channel. (By statistical security we mean security against computationally unbounded adversaries, also known as information-theoretical security.) In contrast, when using only classical communication, it is easy to see that such a secret key can always be extracted by a computationally sufficiently powerful adversary. In light of this result, one might hope that quantum cryptography allows to circumvent other classical impossibility results, possibly even allowing for statistically secure multi-party computation protocols. Yet, Mayers [13] showed that also in the quantum setting, even statistically secure commitment schemes are impossible, let alone general multi-party computation. This is unfortunate, because from commitments one can build OT (Bennett, Brassard, Crépeau, and Skubiszewska [3]), and from OT general multi-party computation (Kilian [11]). A way to work around this impossibility was found by Damgård, Fehr, Salvail, and Schaffner [6]. They showed that if we assume that the quantum memory available to the adversary is bounded (we speak of *bounded quantum storage* (BQS)), we can construct statistically secure commitment and OT schemes. Although such a result is not truly unconditional, it avoids hard-to-justify complexity-theoretic assumptions. Also, it achieves long-term security: even if the adversary can surpass the memory bound after the protocol execution, this will not allow him to retroactively break the protocol.

Yet, we still have not reached the goal of statistically secure multi-party computation. Although we have protocols for commitment and OT, we cannot simply plug them into the protocols by Bennett et al. [3] and by Kilian [11]. The reason is that it is not clear under which circumstances protocols in the BQS model may be composed. For example, Dziembowski and Maurer [7] constructed a protocol that is secure in the classical bounded storage model, but that looses security when composed with a computationally secure protocol. To overcome this remaining difficulty, works by Wehner and Wullschleger [17] and by Fehr and Schaffner [8] give security definitions in the BQS model that enable secure sequential composition. Both works also present secure OT protocols in their respective settings. Based on these, we can construct secure multi-party computation protocols in the BQS model. There are, however, a few limitations. First, since only sequential composition is supported, all instances of the OT protocol used by the multi-party computation need to be executed one after another, leading to a high round-complexity. Second, interactive functionalities such as a commitment are difficult to use: the restriction to sequential composability requires that we have to commit and immediately open a commitment before being allowed to execute the next commitment. Third, the security proof of their OT protocols uses a computationally unlimited simulator. As discussed in [15], a protocol with an unlimited simulator cannot be composed with a computationally secure protocol. Fourth, since we have no concurrent composability, it is not clear what happens if the protocols are executed in an environment where we do not have total control about which protocols are executed at what time.

To overcome the limitations of sequential composition *in the classical setting*, Canetti [4] introduced the Universal Composability (UC) model. In this model, protocols can be arbitrarily composed, even concurrently with other protocols and with copies of themselves. The UC model has been adapted to the quantum setting by Ben-Or and Mayers [1] and by Unruh [14,15]. In light of the success of the UC model, it seems natural to combine the ideas of the UC model with those of the BQS model in order to allow for concurrent composition.

1.1 Our contribution.

We define the notion of BQS-UC-security, which is an extension of quantum-UC-security [15]. We have composability in the following sense: If π is a secure realization of a functionality $\mathcal{F}$, and $\sigma^{\mathcal{F}}$ securely realizes $\mathcal{G}$ by using one instance of $\mathcal{F}$, then σ^{π}, the result of replacing $\mathcal{F}$ by π, still securely realizes $\mathcal{G}$. In contrast to quantum-UC-security, however, BQS-UC-security does not allow for concurrent self-composition: if π is secure, this does not automatically imply that two concurrent instances of π are secure.[1]

[1] The reader may wonder how it can be that σ and π may compose in general while π and π do not. Might not σ and π be the same protocol? The reason lies in the exact conditions of the composition theorem: In order to compose, σ needs to be secure against adversaries with a higher quantum memory bound than π tolerates. Thus σ and π cannot be the same protocol.

In order to get protocols that even self-compose concurrently, we design a commitment scheme π_{COM} such that n concurrent instances of π_{COM} securely realize n instances of the commitment functionality in the presence of a-memory bounded adversaries. Here a and n are arbitrary (polynomially-bounded), but the protocol depends on a and n.

The challenging part in the construction of π_{COM} is that BQS-UC-security requires the following: There must be an *efficient* simulator (which is allowed to have more quantum memory than the adversary) that can extract the committed value (extractability) or change it after the commit phase (equivocality). Prior constructions of commitment schemes in the BQS model required computationally unbounded simulators. Also, the fact that we directly analyze the concurrent composition of several instances of π_{COM} requires care: In the proof, we have hybrid networks in which instances of both π_{COM} and of the simulator occur. Since the simulator uses more quantum memory than π_{COM} tolerates, one needs to ensure that the simulator cannot be (mis)used by the adversary to break the commitment.

Finally, using the composition theorem and π_{COM}, for any two-party functionality $\mathcal{G}$, we get a statistically secure protocol π realizing $\mathcal{G}$ in the BQS model.[2] The protocol is secure even when running n concurrent instances of the protocol. (Again, this holds for any n and any memory bound, but the protocol depends on n and the memory-bound.) The protocol is constant-round. It does not use any quantum memory or quantum computation and thus is in the reach of today's technology.

A full version of this paper with complete proofs and details can be found at [16].

2 Bounded quantum storage UC

2.1 The BQS-UC model

To understand the definition of BQS-UC-security, we first have to understand the idea underlying UC security. In the UC model, a protocol π emulates (realizes, implements, is as secure as) another protocol ρ if any attack on π can also happen on ρ. Thus, if ρ is secure by definition (e.g., because it contains only one trusted machine, a so-called ideal functionality), π must also be secure. To formalize this, we introduce the concept of an adversary and a simulator. We require that for any adversary attacking π (real model), there is a simulator attacking ρ (ideal model) such that the real and the ideal model are indistinguishable. To define indistinguishability, we introduce another machine, the environment. Its task is to try and distinguish between the real and the ideal model. The environment provides the inputs to the protocol parties, gets their outputs, and may talk to the adversary/simulator. We then get the following definition: π UC-emulates ρ

[2] We are restricted to two-party functionalities because our construction uses a subprotocol by Wolf and Wullschleger [18,19] to reverse the direction of an OT; this protocol only makes sense in a two-party setting.

if for any adversary Adv there is a simulator Sim such that for all environments $\mathcal{Z}$, the probability that $\mathcal{Z}$ outputs 1 is approximately the same in the networks $\pi, \mathrm{Adv}, \mathcal{Z}$ and $\rho, \mathrm{Sim}, \mathcal{Z}$.

To translate this to the quantum setting, we only need to change the machine model to allow for quantum machines instead of classical machines. Given networks S, S' of quantum machines with $\mathcal{Z} \in S, S'$, we say that S and S' are ε-*close* if $|P - P'| \leq \varepsilon$ where P is the probability that $\mathcal{Z}$ outputs 1 in an execution of S, and P' is defined analogously. Networks are *negligible-close* if ε is negligible, and *perfectly close* if $\varepsilon = 0$. We call a machine M a-memory bounded if it keeps at most a qubits of quantum memory between activations. (An activation is the computation performed by a machine between receiving a message and sending the immediate response.) We do not impose any limitations on the computation-time or memory-use during a single activation of M. We write $\mathrm{QM}(M)$ for the memory bound of M (i.e., $\mathrm{QM}(M)$ is the smallest a such that M is a-memory bounded). A protocol π is a network not containing adversary, simulator, or environment. The set of corruptible protocol parties is denoted $\mathit{parties}_\pi$. Given $C \subseteq \mathit{parties}_\pi$, we denote by π^C the protocol where all parties in C have been replaced by *corruption parties* which are controlled by the adversary. For details on the machine and the network model, we refer to the full version [16] or to [15].

We can now formulate BQS-UC-security. Intuitively, a protocol is BQS-UC-secure if it is UC-secure for memory-bounded adversaries. To formulate this, we need to explicitly parametrize the definition over a memory bound a. Then we require that the total quantum memory used by environment and adversary is bounded by a. The reason why we include the environment's memory is that the latter can be involved in the actual attack: If only the adversary's memory was bounded, the adversary could use the environment as an external storage to perform the attack (see also our discussion on page 5).[3]

It remains to decide whether the simulator should be memory bounded. If we allow the simulator to be unbounded, composition becomes difficult: In some cases, the simulator of one protocol plays the role of the adversary of a second protocol. Thus, if simulators where not memory bounded, the second protocol would have to be secure against unbounded adversaries. However, if we require the simulator to be a-memory bounded, we will not be able to construct non-trivial protocols: In order to perform a simulation, the simulator needs to have some advantage over an honest protocol participant (in the computational UC setting, e.g., this is usually the knowledge of some trapdoor). In our setting, the advantage of the simulator will be that he has more quantum memory than the adversary. Thus we introduce a second parameter s which specifies the amount of quantum memory the simulator may use for the simulation. More precisely, we allow the simulator to use $s + \mathrm{QM}(\mathrm{Adv})$ qubits because the simulator will usually internally simulate the adversary Adv as a black-box and therefore have to additionally reserve sufficient quantum memory to store the adversary's state.

[3] This is captured more formally by the completeness of the so-called dummy-adversary (see the full version [16]), which shows that one can even shift the complete attack into the environment.

Definition 1 (BQS-UC-security). *Fix protocols π and ρ. Let $a, s \in \mathbb{N}_0 \cup \{\infty\}$ (possibly depending on the security parameter). We say π (a,s)-*BQS-UC-emulates[4] *ρ iff for every set $C \subseteq parties_\pi$ and for every adversary* Adv *there is a simulator* Sim *with* $\mathrm{QM}(\mathrm{Sim}) \leq s + \mathrm{QM}(\mathrm{Adv})$ *such that for every environment $\mathcal{Z}$ with* $\mathrm{QM}(\mathcal{Z}) + \mathrm{QM}(\mathrm{Adv}) \leq a$*, the networks* $\pi^C \cup \{\mathrm{Adv}, \mathcal{Z}\}$ *(called the real model) and* $\rho^C \cup \{\mathrm{Sim}, \mathcal{Z}\}$ *(called the ideal model) are negligible-close. We furthermore require that if* Adv *is quantum-polynomial-time, so is* Sim.

In most cases, the behavior of the ideal protocol ρ is described by a single machine $\mathcal{F}$, the so-called ideal functionality. We can think of this functionality as a trusted third party that perfectly implements the desired protocol behavior. In order to apply Definition 1 to ideal functionalities (e.g., π BQS-UC-emulates $\mathcal{F}$) we have to be able to consider an ideal functionality as a protocol. Following [4,15], we do this by introducing dummy-parties. That is, the protocol $\mathcal{F}$ consists of the functionality $\mathcal{F}$ together with a *dummy-party* $\tilde{A}$ for every party A. The dummy-parties just forward inputs/outputs between the functionality $\mathcal{F}$ and the environment. They can, however, be corrupted by the adversary/simulator. This allows the adversary/simulator to control the inputs/outputs of that party. When we write π BQS-UC-emulates $\mathcal{F}$, we always assume the presence of dummy-parties in the ideal model. For details, we refer to the full version [16] or to [15].

On the memory bound of the environment. In Definition 1, we impose the memory bound on both the adversary and the environment. In this, we differ from the modeling by Wehner and Wullschleger [17]. In their definition, the environment (which is implicit in the definition of the indistinguishability $\equiv_\varepsilon$ of quantum channels) provides the input state to protocol and adversary, then gets the outputs of protocol and adversary, and finally the environment has to guess whether it interacted with the real or the ideal model. During the interaction of the protocol, the environment is not allowed to communicate with any other machine. Between its two activations, the environment is allowed to keep an arbitrarily large quantum state.[5] The interesting point here is that, in contrast to our Definition 1, Wehner and Wullschleger do not impose the memory bound on the environment, only on the adversary. They motivate unlimited environments by pointing out that it is more realistic to assume that a particular memory bound (say, 100 qubits) applies to a particular adversary (e.g., a smart card) than to the whole environment (i.e., all computers world-wide). We believe, however, that this reasoning has to be applied with care: Only when we have the guarantee

[4] Since we only consider statistical security in this work, we omit the qualifier "statistical". Similarly, when we speak about classical-UC-security and quantum-UC-security, we mean the statistical variant of that notion.

[5] Note that strictly speaking, the formalism of [17, full version] does not model an environment with quantum memory: For quantum channels Λ, Λ' they define $\Lambda \equiv_\varepsilon \Lambda'$ iff for all quantum states ρ, the trace distance between $\Lambda(\rho)$ and $\Lambda'(\rho)$ is at most ε. To model environments with quantum memory, we should instead require that for all Hilbert spaces $\mathcal{H}$ and all quantum states ρ, the trace distance between $(\Lambda \otimes id_{\mathcal{H}})(\rho)$ and $(\Lambda' \otimes id_{\mathcal{H}})(\rho)$ is at most ε. We believe that the latter was the intended meaning of $\equiv_\varepsilon$.

that the adversary (e.g., the smart card) cannot communicate with any other machines can we assume a smart card is limited to 100 qubits. Otherwise, we have to assume that the smart card effectively has (in the worst case) access to all the quantum memory of the environment. Thus, except in very specific cases, the memory bound we assume needs to be large enough to encompass the environment's memory as a whole. Thus, the bound we assume not to be surpassed by the adversary's memory needs to be large enough that it makes sense to assume that the environment does not surpass this bound either. But in this case, we can safely assume in Definition 1 that the environment is restricted by that memory bound.

We stress that even if our environment is memory bounded, we do take into account the fact that an environment can have a quantum state that is entangled with that of the adversary; we just limit this quantum state to the memory bound.

We get, however, an interesting variant of our model if we follow the approach of Wehner and Wullschleger as follows. We call a machine a-$\diamond$-memory-bounded[6] if its state between activations consists of two registers A and B. The register A contains at most a qubits, and register B is unlimited but is only accessed in the first and the last activation of B. We denote by $\mathrm{QM}_\diamond(\mathcal{Z})$ the $\diamond$-memory bound of $\mathcal{Z}$. We define (a,s)-$\diamond$-BQS-UC-emulation like (a,s)-BQS-UC-emulation (Definition 1), except that we use $\mathrm{QM}_\diamond(\mathcal{Z})$ instead of $\mathrm{QM}(\mathcal{Z})$. (But we still use QM(Adv) and QM(Sim).) We stress that our techniques also work for this definition. All results of this section still hold with essentially unmodified proofs (except that we always have to refer to the $\diamond$-memory bound of the environment instead memory bound). The results from Section 3 (BQS-UC commitments) are based on the existence of certain commitment schemes that are hiding with respect to memory bounded adversaries. We use the commitment scheme from [12] (see Theorem 6). To extend the results from Section 3 to $\diamond$-BQS-UC, we need schemes that are hiding with respect to $\diamond$-memory bounded adversaries instead. Besides that, the proofs of the results in Section 3 stay essentially the same (except for using $\diamond$-memory bounds instead of memory bounds).

2.2 Composition

For some protocol σ, and some protocol π, by σ^π we denote the protocol where σ invokes (up to polynomially many) instances of π. That is, in σ^π the machines from σ and from π run together in one network, and the machines from σ access the inputs and outputs of π. (That is, σ plays the role of the environment from the point of view of π. In particular, $\mathcal{Z}$ then talks only to σ and not to the sub-protocol π directly.) A typical situation would be that $\sigma^{\mathcal{F}}$ is some protocol that makes use of some ideal functionality $\mathcal{F}$, say a commitment functionality, and then σ^π would be the protocol resulting from implementing that functionality with some protocol π, say a commitment protocol. One would hope that such

[6] We use the symbol $\diamond$ because $\diamond$-memory-bounded environments essentially model indistinguishability with respect to the so-called $\diamond$-norm.

an implementation results in a secure protocol σ^π. That is, we hope that if π BQS-UC-emulates $\mathcal{F}$ and $\sigma^{\mathcal{F}}$ BQS-UC-emulates $\mathcal{G}$, then σ^π BQS-UC-emulates $\mathcal{G}$. Fortunately, this is the case, as long as we pick the memory bounds in the right way:

Theorem 2 (Composition Theorem). *Let π and σ be quantum-polynomial-time protocols and $\mathcal{F}$ and $\mathcal{G}$ be quantum-polynomial-time functionalities. Assume that σ invokes at most one subprotocol instance. Assume that π (a,s)-BQS-UC-emulates $\mathcal{F}$ and that $\sigma^{\mathcal{F}}$ $(a - \mathrm{QM}(\sigma) + s, s')$-BQS-UC-emulates $\mathcal{G}$. Then σ^π $(a - \mathrm{QM}(\sigma), s + s')$-BQS-UC-emulates $\mathcal{G}$.*

The proof of this theorem is very similar to that in [15], except that we have to keep track of the quantum memory used by various machines constructed in the proof.

Notice that in this composition theorem, the outer protocol σ is only allowed to invoke one instance of the subprotocol π. This stands in contrast to the universal composition theorem for classical-UC [4] and for quantum-UC [15] where any polynomially-bounded number of concurrent instances of π is allowed. In fact, this is not just a limitation of our proof technique.[7] For example, assume a protocol $\pi_{\mathrm{COM}}^{A\to B}$ that (a,s)-BQS-UC-emulates the commitment functionality $\mathcal{F}_{\mathrm{COM}}^{A\to B}$ with sender A and recipient B. Assume further that $\pi_{\mathrm{COM}}^{A\to B}$ does not use any functionalities as setup. As we will see later, such a protocol exists. Now let $\pi_{\mathrm{COM}}^{B\to A}$ be the protocol that results from exchanging the roles of A and B. Then $\pi_{\mathrm{COM}}^{B\to A}$ (a,s)-BQS-UC-emulates $\mathcal{F}_{\mathrm{COM}}^{B\to A}$. Consider the concurrent composition of $\pi_{\mathrm{COM}}^{A\to B}$ and $\pi_{\mathrm{COM}}^{B\to A}$. In this protocol, a corrupted Bob may reroute all messages between the Alice in the first protocol and Alice in the second protocol. Thus, if Alice commits to a random value v in the first protocol, Bob commits to the same value v in the second protocol without knowing it. It is easy to see that in a concurrent composition of $\mathcal{F}_{\mathrm{COM}}^{A\to B}$ and $\mathcal{F}_{\mathrm{COM}}^{B\to A}$, this is not possible. Thus the composition of $\pi_{\mathrm{COM}}^{A\to B}$ and $\pi_{\mathrm{COM}}^{B\to A}$ does not (a',s')-BQS-UC-emulate the composition of $\mathcal{F}_{\mathrm{COM}}^{A\to B}$ and $\mathcal{F}_{\mathrm{COM}}^{B\to A}$ (for any parameters a', s'). To convert this into an example of a protocol that does not even compose with itself, just consider the protocol $\pi_{\mathrm{COM}}^{A\leftrightarrow B}$ in which Bob may choose whether $\mathcal{F}_{\mathrm{COM}}^{A\to B}$ or $\mathcal{F}_{\mathrm{COM}}^{B\to A}$ should be executed. It might be possible to make $\pi_{\mathrm{COM}}^{A\leftrightarrow B}$ self-composable by adding suitable tags inside the messages, but the definition of BQS-UC-security does not enforce this.

Although BQS-UC-security does not guarantee for concurrent self-composability, individual protocols may have this property. In order to formulate this, we introduce the concept of the multi-session variant of a protocol. Given a protocol π and a polynomially-bounded n, we define π^n to be the protocol that executes n instances of π concurrently.

Then, from Theorem 2, we immediately get the following corollary:

[7] In the proof, the difficulty arises from a hybrid argument where the protocol π is executed together in one network with the protocol ρ and the corresponding simulator. Since the simulator may use more quantum memory than π is resistant against, we cannot guarantee security of π in this hybrid setting and the proof cannot proceed.

Corollary 3. *Let π and σ be quantum-polynomial-time protocols and $\mathcal{F}$ and $\mathcal{G}$ be quantum-polynomial-time functionalities. Let $n, m \geq 0$ be integers (depending on the security parameter). Assume that σ invokes at most m subprotocol instances. Assume that π^{nm} (a, s)-BQS-UC-emulates $\mathcal{F}^{nm}$ and that $(\sigma^{\mathcal{F}})^n$ $(a - n\mathrm{QM}(\sigma) + s, s')$-BQS-UC-emulates $\mathcal{G}^n$. Then $(\sigma^{\pi})^n$ $(a - n\mathrm{QM}(\sigma), s + s')$-BQS-UC-emulates $\mathcal{G}^n$.*

3 Commitments

3.1 Extractable commitments

In this section, we present the notion of online-extractable commitments in the BQS model. These will be used as a building block for constructing BQS-UC commitments in the next section.

Definition 4 ((ε, a)-BQS-hiding). *Given a commitment protocol π with sender Alice and recipient Bob, and an adversary B' corrupting Bob, we denote with $\langle A(m), B' \rangle_{B'}$ the output of B' in an interaction between Alice and B' where Alice commits to m.*

We call π (ε, a)-BQS-hiding *iff for all a-memory bounded B' and all $m_1, m_2 \in M$, we have that $\left|\Pr[\langle A(m_1), B' \rangle_{B'} = 1] - \Pr[\langle A(m_2), B' \rangle_{B'} = 1]\right| \leq \varepsilon$. Here M is the message space of the commitment scheme.*

Instead of the binding property, we will need a stronger property: online-extractability. This property guarantees that there is a machine (the *extractor*) that, when running as the recipient of the commit protocol, is able to output the committed value V already after the commit phase. This extractor should be indistinguishable from an honest recipient. Note that this does not contradict (ε, a)-BQS-hiding since we allow the extractor's quantum memory to contain more than a qubits. For our purposes, we will only need a definition of online-extractability that does not impose a memory bound on the adversary. We do, however, make the memory bound s of the extractor explicit.

Definition 5 ((ε, s)-online-extractable). *Given a commitment protocol π with sender Alice and recipient Bob, an* extractor *is a machine B_S that, after the commit phase, gives an output V' and then executes the (honest) code of Bob for the open phase and outputs a value V (the accepted value). (In particular, B_S needs to provide an initial state for the program of the open phase of Bob that matches the interaction so far.) We write $V = \bot$ if the open phase fails.*

For an adversary A', we denote with $\langle A', B \rangle_{A'}$ ($\langle A', B_S \rangle_{A'}$) the output of A' in an interaction between A' and Bob (B_S) where A' is given V after Bob (B_S) terminates.

We call π (ε, s)-online-extractable *iff there exists an s-memory bounded quantum-polynomial-time extractor B_S such that for all adversaries A', we have that $\left|\Pr[\langle A', B \rangle_{A'} = 1] - \Pr[\langle A', B_S \rangle_{A'} = 1]\right| \leq \varepsilon$ and in an interaction of A' and B_S, we have $\Pr[V \notin \{V', \bot\}] \leq \varepsilon$.*

Theorem 6 (Online-extractable commitments). *For any polynomially-bounded integers a and ℓ, there is a constant-round 0-memory bounded (ε, a)-BQS-hiding (ε, s)-online-extractable commitment scheme π for some exponentially-small ε and some polynomially-bounded s. The message space of π is $M = \{0,1\}^\ell$.*

A protocol with the properties from Theorem 6 was constructed in [12]. They did not, however, show that it is online-extractable. In the full version [16] we show how their proof of the binding property can be extended to online-extractability.

3.2 BQS-UC commitments

In this section, we present a commitment scheme π_{COM} that is BQS-UC-secure for memory bound a and for n concurrent instances of π_{COM}. The parameters a and n can be arbitrary, but π_{COM} depends on them. To state our result, we first define the ideal functionality for commitments.

Definition 7 (Commitment). *Let A and B be two parties. The functionality $\mathcal{F}_{\mathrm{COM}}^{A\to B,\ell}$ behaves as follows: Upon (the first) input $(\texttt{commit}, x)$ with $x \in \{0,1\}^{\ell(k)}$ from A, store x and send* `committed` *to B. Upon (the first) input* `open` *from A send $(\texttt{open}, x)$ to B (unless x is still undefined). All communication/input/output is classical. We call A the sender and B the recipient.*

Note that this definition also defined the behavior of the functionality in the case where A or B is corrupted. In this case, the adversary (or simulator) is allowed to send/receive the inputs/outputs in the name of A or B, respectively. For example, if A is corrupted, the adversary can decide when to commit to what message and when to open.

Intuition. The protocol π_{COM} is depicted in Figure 1. Before we prove its security, we first explain the underlying intuition. In order to prove the BQS-UC-security of π_{COM}, it is necessary to construct a simulator (that may use more quantum memory than the adversary) that achieves the following: When being in the role of the recipient, the simulator is able to extract the commitment after the commit phase. When being in the role of the sender, the simulator should be able to open the commitment to any value of his choosing (equivocality). The first requirement can easily be achieved by using the online-extractable commitment scheme from Theorem 6. That scheme, however, is not equivocal. In order to make our protocol equivocal, we intentionally weaken the binding property of the commitment. Instead of committing to a single value v, the sender commits using a commitment scheme C_2 to random values $\underline{R} := R_1, \dots, R_m$. Then he sends $v \oplus F(\underline{R})$ with F being a universal hash function and sends the syndrome σ of $\underline{R}$ with respect to a suitable linear code. In the open phase, the sender does not open all commitments R_i, but instead just sends $\underline{R}$ to the recipient. The recipient chooses a test set T, and the sender opens R_i for $i \in T$. The modified scheme is still binding: Assume the sender wishes to be able to open the

Parameters: Integers ℓ (the length of the committed value), m, $c < m$, b, d, $\kappa < m$. A b-block (m, κ, d)-linear code[8] where $\mathbf{S}(\omega) \in \{0,1\}^{(m-\kappa)b}$ denotes the syndrome of a codeword $\omega \in \{0,1\}^{mb}$. A family $\mathbf{F}$ of strongly universal hash functions $F : \{0,1\}^{mb} \to \{0,1\}^{\ell}$. All parameters may depend on the security parameter k.
Subprotocols: A commitment scheme C_1 with sender Bob, and a commitment scheme C_2 with sender Alice, both 0-memory bounded (not using quantum memory).[9]
Parties: The sender Alice A and the recipient Bob B.
Inputs: In the commit phase, Alice gets $(\texttt{commit}, v)$ with $v \in \{0,1\}^{\ell}$. In the open phase, Alice gets `open`. Bob gets no inputs.
Commit phase:
C1. Bob picks a random $T \subseteq \{1, \dots, m\}$ with $\#T = c$. Then Bob commits to T using C_1. (We assume some encoding of sets T that does not allow to encode sets with $\#T \neq c$.)
C2. Alice picks $R_1, \dots, R_m \in \{0,1\}^b$. For each i, Alice commits to R_i using C_2. (The commitments may be performed concurrently.)
C3. Alice picks a hash function $F \leftarrow \mathbf{F}$, computes $p := v \oplus F(R_1 \| \dots \| R_m)$, computes the syndrome $\sigma := \mathbf{S}(R_1 \| \dots \| R_m)$, and sends (F, σ, p) to Bob. (This may be done concurrently with the commitments to R_i.)
C4. Bob outputs `committed`.
Open phase:
O1. Alice sends $R_1 \| \dots \| R_m$ to Bob.
O2. Bob opens T using C_1.
O3. For each $i \in T$, Alice opens R_i using C_2. (The open phases may be executed concurrently.)
O4. Bob checks that the values R_i sent by Alice match the values R_i opened by Alice for all $i \in T$, and that $\sigma = \mathbf{S}(R_1 \| \dots \| R_m)$.
O5. Bob computes $v := p \oplus F(R_1 \| \dots \| R_m)$ and outputs $(\texttt{open}, v)$. (I.e., Bob accepts the opened value v.)

Fig. 1. Our commitment protocol π_{COM}.

commitment with two different values. Then he has to find values $\underline{R}' \neq \underline{R}$ that both pass the recipients checks in the open phase. If $\underline{R}'$ differs from $\underline{R}$ in many blocks R_i, with high probability the verifier will require that one of these R_i is opened and the sender will be caught. If $\underline{R}'$ and $\underline{R}$ differs in only few blocks, then $\underline{R} - \underline{R}'$ has a low Hamming weight and is not in the code. Hence the syndrome of $\underline{R} - \underline{R}'$ is not zero, and, since the code is linear, the syndromes of $\underline{R}'$ and $\underline{R}$ cannot both equal σ. Thus the sender is caught, too. Furthermore, our scheme is online-extractable if C_2 is online-extractable since the simulator can extract the committed values $\underline{R}$. However, we have not yet achieved the equivocality. In order to open the commitment to a different value, the sender needs to know T before sending $\underline{R}'$. To achieve this, the recipient commits to T before the commit phase (using an online-extractable commitment scheme C_2). A simulator wishing to change the value of the commitment simply extracts T. Then he knows which R_i can be changed without being detected and can thus change $F(R_1, \dots, R_m)$ to any value he wishes.

Difficulties with concurrent composition. The main difficulty in showing the BQC-UC-security of π_{COM} lies in coping with the fact that several (say n) instances of π_{COM} might run concurrently. Consider for example the case that Alice is corrupted. In this case, the adversary may produce the C_2-commitments to $R_1, \dots, R_m$ in n instances of Alice. The simulator needs to run the nm extractors to extract these commitments. Each of these extractors needs some quantum memory s_2. Thus our simulator needs nms_2 bits of quantum memory. On the other hand, we need to make sure that the C_1-commitments to T, produced by the simulator, are hiding. C_1 needs to be hiding against a_1-bounded adversaries with $a_1 > nms_2 \geq s_2$ (because we cannot be sure that the memory used by the simulator is not misused by the adversary). But then the extractor for C_1 needs to use $s_1 > a_1$ qubits; otherwise the adversary could run the extractor to break the protocol. Similarly, we can see that when Bob is corrupted, C_2 needs to be hiding against a_2-memory bounded adversaries with $a_2 > ns_1 \geq s_1$, and $s_2 > a_2$. Thus we need $a_1 > s_2 > a_2 > s_1 > a_1$ which is impossible.

Solving the difficulties. The way out is to carefully track the memory used by the simulators; it turns out that in the proof of security against corrupted Alice, we can make sure that the adversary is not able to "misuse" the memory of the simulator. When Alice is corrupted, we need to construct a simulator that extracts the values v of n concurrent commitments produced by Alice, while being indistinguishable from an execution of the honest recipient Bob. More precisely, we show that the simulator is indistinguishable if C_1 is a_1-BQS-hiding and C_2 is s_2-online-extractable and environment and adversary are a_1-memory-bounded. We do not require that $a_1 > s_2$, thus breaking the above-mentioned circularity in the choices of a_1, s_1, a_2, s_2.

Let B^* be defined like the honest Bob, except that instead of honestly running the recipient's code for C_2, B^* runs the extractor B_S for C_2 to extract the committed values $\underline{R}$ in the C_2-commitments. From $\underline{R}$, B^* computes a guess v' for the committed value v. B^* does not, however, use this guess at any point.

First, note that B^* is indistinguishable from honest Bob: This follows from the fact that the extractor for C_2 is indistinguishable from the recipient for C_2. Furthermore, as discussed in the section "Intuition" above, extracting v will be successful as long as Alice does not learn anything about T, i.e., as long as C_1 is hiding. But C_1 is only a_1-BQS-hiding. And B^* uses ms_2 qubits to run the extractors for C_2, so the total memory used in the network is $a_1 + ms_2$ which is beyond the memory bound tolerated by C_1. Fortunately, however, honest Bob runs the recipient of C_2 after the end of the commit phase and before the beginning of the open phase of C_1. Thus, from the point of view of C_1, the extractors are executed within a single atomic computation. And we defined BQS-hiding to hold even if the adversary uses unlimited memory within a single

[8] That is, a code where the code words consist of m blocks of b bit, that contains $2^{b\kappa}$ codewords, and where every non-zero codeword contains at least d nonzero blocks.

[9] Note that this does not refer to the memory bound of the adversary. We only state that honest Alice and Bob do not need to use quantum memory in C_1 and C_2.

activation. Thus the ms_2 qubits used by the extractors do not break the hiding property, and we get that B^* guesses the right v' with overwhelming probability.

This argument does, however, only work when a single instance of B^* is executed. If several instances of B^* are executed, one instance may run the commit or open phase of C_1 concurrently with another instance's extractors. Two show that n concurrent instances of B^* extract successfully, we use the following argument: For each j, we have that if only the j-th Bob instance is replaced by B^*, then B^* extracts correctly. Furthermore, Bob and B^* are indistinguishable, thus if we replace all the other instances of Bob by B^*, the j-th instance still extracts correctly. Thus, for any j, if there are n instances of B^*, then the j-th instance extracts correctly. Thus all instances of B^* extract correctly. And the instances of B^* are indistinguishable from the instances of Bob.

Finally, we can construct a simulator that runs B^* instead of Bob and uses the value v' extracted by B^* as input to the commitment functionality. Since $v = v'$ with overwhelming probability, this simulator is successful.

Thus we have shown that a_1 can be chosen independently of s_2. This allows to break the circularity in the choices of a_1, s_1, a_2, s_2: We first start with an arbitrary $a = a_1$. Then we pick an arbitrary a_1-BQS-hiding and s_1-online-extractable C_1, and then an arbitrary $a_2 := a + ns_1$-BQS-hiding and s_2-online-extractable commitment C_2. For the case of corrupted Bob, we then construct a simulator that uses ns_1 qubits and is secure against $a = a_2 - ns_1$-memory bounded environments and adversaries. And for the case of corrupted Alice, using the argument above, we get a simulator that uses nms_2 qubits and is secure against $a = a_1$-bounded environments and adversaries.

The analysis. We proceed with the formal analysis of π_{COM}. We first consider the case where the recipient is corrupted.

Lemma 8. *Assume that ε, δ are negligible, n, c are polynomially-bounded, and $2\kappa b - mb - 2cb - \ell$ is superlogarithmic (in the security parameter k). Assume that C_1 is (ε, s_1)-online-extractable and C_2 is $(\delta, a + ns_1)$-BQS-hiding. Assume that $\mathbf{F}$ is a family of affine strongly universal hash functions.*

Then π^n_{COM} (a, ns_1)-BQS-UC-emulates $(\mathcal{F}^{A\to B,\ell}_{\mathrm{COM}})^n$ for corrupted recipient B.

Proof. First, we describe the structure of the real and ideal model in the case that the party B (Bob) is corrupted:

In the real model, we have the environment $\mathcal{Z}$, the adversary Adv, the honest party A (Alice), the corruption party B^C. The adversary controls the corruption party B^C, so effectively he controls the communication between Alice and Bob. The environment provides Alice's inputs $(\texttt{commit}, v)$ and $\texttt{open}$. See Figure 2 (a).

In the ideal model, we have the environment $\mathcal{Z}$, the simulator Sim (to be defined below), the dummy-party $\tilde{A}$, the corruption party B^C, and the commitment functionality $\mathcal{F}_{\mathrm{COM}}$. The inputs $(\texttt{commit}, v)$ and $\texttt{open}$ of $\mathcal{F}_{\mathrm{COM}}$ are provided by the dummy-party $\tilde{B}$ and thus effectively by the environment $\mathcal{Z}$. The simulator Sim controls the corruption party B^C and hence gets the outputs $\texttt{committed}$ and $(\texttt{open}, v)$ of $\mathcal{F}_{\mathrm{COM}}$. See Figure 2 (b).

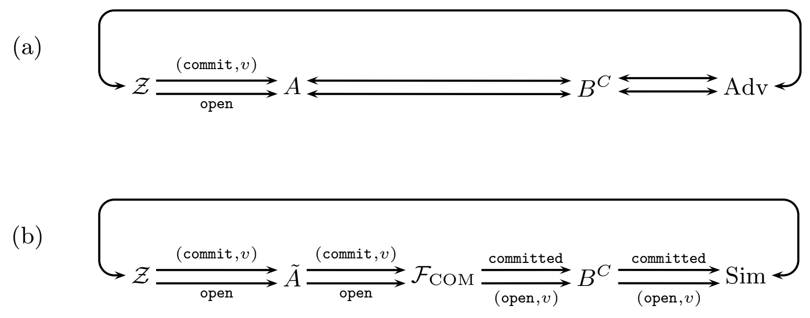

Fig. 2. Networks occurring in the proof of Lemma 8.

Fix an adversary Adv. To show Lemma 8, we need to find a simulator Sim with $\mathrm{QM}(\mathrm{Sim}) \leq ns_1$ such that, for any environment $\mathcal{Z}$ with $\mathrm{QM}(\mathcal{Z}) + \mathrm{QM}(\mathrm{Adv}) \leq a$, the real model and the ideal model are negligible-close. This simulator is described in Figure 3. We use the abbreviations $\underline{R} := R_1\|\dots\|R_m$ and $\underline{R}' := R'_1\|\dots\|R'_m$.

To show that the real and the ideal model are negligible-close, we start with the real model, and change the machines in the real model step-by-step until we end up with the ideal model. In each step, we show that the network before and after that step are negligible-close.

Game 1. We change the machine A as follows: Instead of executing the program of the honest recipient of C_1, A executes the extractor A_S. $\diamond$

Let T' denote the extracted value. The modified A does not use T'. Since there are up to n copies of A, and since C_1 is (ε, s_1)-online-extractable, the real model and Game 1 are $n\varepsilon$-close.

Game 2. We change the machine A to abort if the opening of T succeeds and reveals a value $T \neq T'$. $\diamond$

Since C_1 is (ε, s_1)-online-extractable, in each instance of A, this happens with probability at most ε, thus Game 1 and Game 2 are $n\varepsilon$-close.

Notice that the only machines that use quantum memory in Game 2 are $\mathcal{Z}$, Adv, and n copies of A_S. Since A_S is s_1-memory bounded, and $\mathrm{QM}(\mathcal{Z}) + \mathrm{QM}(\mathrm{Adv}) \leq a$ we have that the total amount of quantum memory used in Game 2 is bounded by $a + ns_1$.

Game 3. We change the machine A to commit to 0^b instead of R_i for each $i \notin T'$. $\diamond$

To see that Game 2 and Game 3 are negligible-close, we introduce an intermediate hybrid game, Game 3_j, in which only the first j of the commitments to $R_i, i \notin T$ are replaced by commitments to 0^b. Since at most $a + ns_1$ qubits of quantum memory are used in Game 2 and therefore also in Game 3_j, and since the C_2-commitments to $R_i, i \notin T$ are never opened, from the fact that C_2 is $(\delta, a + ns_1)$-BQS-hiding it follows that Game 3_j and Game 3_{j+1} are δ-close. Note that there are, in the whole game, up to n copies of A and thus

Commit phase (on input `committed`):
- When Bob commits to T using C_1, the simulator runs the extractor A_S for C_1 instead of the honest recipient's program. (Since C_1 has recipient Alice, we write A_S, not B_S.) Let T' denote the value extracted by A_S.
- The simulator picks $R_1, \dots, R_m \in \{0,1\}^b$. For each i, Sim commits (honestly) to R_i (if $i \in T$) or to 0^b (if $i \notin T$) using C_2.
- Sim picks a hash function $F \overset{R}{\leftarrow} \mathbf{F}$, picks a random $p \overset{R}{\leftarrow} \{0,1\}^\ell$, computes the syndrome $\sigma := \mathbf{S}(\underline{R})$, and sends (F, σ, p) to Bob.

Open phase (on input $(\mathtt{open}, v)$ with $v \in \{0,1\}^\ell$):
- Sim picks $\underline{R}' \in \{\underline{R}' : \forall i \in T'. R_i = R'_i, \sigma = \mathbf{S}(\underline{R}'), p \oplus F(\underline{R}') = v\}$ uniformly.[10]
- Sim sends $\underline{R}'$ to Bob.
- Sim waits for Bob to open T using C_1. If $T \neq T'$, Sim aborts.
- For each $i \in T'$, Sim (honestly) opens R_i using C_2.

Fig. 3. Simulator Sim for the case of corrupted Bob. The program described in this figure is executed for each instance of the n instances of π_{COM}. Communication with Bob is sent to an internally simulated instance of the adversary Adv.

up to nc C_2-commitments to some $R_i, i \notin T$. Thus Game 2 = Game 3_0 and Game 3 = Game 3_{nc} are $nc\delta$-close.

Game 4. We modify A to set $\underline{R}' := \underline{R}$ and to send $\underline{R}'$ instead of $\underline{R}$ to Bob in step O1. $\diamond$

This modification is for notational purposes only, Game 3 and Game 4 are perfectly close.

Game 5. We modify the way A chooses $F, \underline{R}, \underline{R}', \sigma, p$: In Game 4, we have $F \overset{R}{\leftarrow} \mathbf{F}$, $\underline{R} \overset{R}{\leftarrow} \{0,1\}^{mb}$, $\sigma := \mathbf{S}(\underline{R})$, $p := v \oplus F(\underline{R})$, $\underline{R}' := \underline{R}$. (We call this distribution $\mathcal{D}_1$.) In Game 5 we use $F \overset{R}{\leftarrow} \mathbf{F}$, $\underline{R} \overset{R}{\leftarrow} \{0,1\}^{mb}$, $p \overset{R}{\leftarrow} \{0,1\}^\ell$, $\sigma := \mathbf{S}(\underline{R})$, $\underline{R}' \overset{R}{\leftarrow} \{\underline{R}' : \forall i \in T'. R_i = R'_i, \sigma = \mathbf{S}(\underline{R}'), p \oplus F(\underline{R}') = v\} =: \mathcal{R}_{F,\underline{R},p}$. (We call this distribution $\mathcal{D}_2$.) $\diamond$

To show that Game 4 and Game 5 are negligible-close, we use the following claim:

Claim 1 *Let $R_T := (R_i)_{i \in T}$. For any $v \in \{0,1\}^\ell$, the statistical distance between $(F, R_T, \underline{R}', \sigma, p)$ chosen according to $\mathcal{D}_1$ and $(F, R_T, \underline{R}', \sigma, p)$ chosen according to $\mathcal{D}_2$ is at most $2^{cb+mb/2+\ell/2-\kappa b-1}$.*

The proof of this claim uses the fact that p is the result of applying F to a random variable $\underline{R}$ with high min-entropy. Due to the leftover-hash-lemma [9], p is indistinguishable from randomness. We refer to the full version [16] for the proof. Using Claim 1 and the fact that we have n instances of A, we immediately

[10] Note $\underline{R}'$ can be sampled efficiently since the conditions $\forall i \in T'. R_i = R'_i$, $\sigma = \mathbf{S}(\underline{R}')$, and $p \oplus F(\underline{R}') = v$ are a system of linear equations. This uses that $\mathbf{S}$ is the syndrome of a linear code, and that $\mathbf{F}$ is a family of affine functions.

(a)
$$\begin{tikzcd}
\text{Adv} \arrow[r, leftrightarrow, shift left] \arrow[r, leftrightarrow, shift right] & A^C \arrow[r, leftrightarrow, shift left] \arrow[r, leftrightarrow, shift right] & B \arrow[r, shift left, "\texttt{committed}"] \arrow[r, shift right, "{(\texttt{open},v)}"'] & \mathcal{Z} \arrow[lll, leftrightarrow, bend right=20]
\end{tikzcd}$$

(b)
$$\begin{tikzcd}
\text{Sim} \arrow[r, shift left, "{(\texttt{commit},v)}"] \arrow[r, shift right, "\texttt{open}"'] & A^C \arrow[r, shift left, "{(\texttt{commit},v)}"] \arrow[r, shift right, "\texttt{open}"'] & \mathcal{F}_{\text{COM}} \arrow[r, shift left, "\texttt{committed}"] \arrow[r, shift right, "{(\texttt{open},v)}"'] & \tilde{B} \arrow[r, shift left, "\texttt{committed}"] \arrow[r, shift right, "{(\texttt{open},v)}"'] & \mathcal{Z} \arrow[llll, leftrightarrow, bend right=20]
\end{tikzcd}$$

Fig. 4. Networks occurring in the proof of Lemma 9.

get that Game 4 and Game 5 are $n(2^{cb+mb/2+\ell/2-\kappa b-1})$-close because the values $(R_i)_{i\notin T}$ are never used by A (except indirectly through $\underline{R}'$, σ, and p).

Finally, note that by construction of Sim, Game 5 and the ideal model are perfectly close. Thus the real and the ideal model are γ-close with $\gamma := 2n\varepsilon + nc\delta + n(2^{cb+mb/2+\ell/2-\kappa b-1})$. Since ε, δ are negligible, and n, c are polynomially-bounded, and $2\kappa b - mb - 2cb - \ell$ is superlogarithmic, we have that γ is negligible. Thus π^n_{COM} (a, ns_1)-BQS-UC-emulates $(\mathcal{F}^{A\to B,\ell}_{\text{COM}})^n$ in the case of corrupted Bob. □

Lemma 9. *Assume that ε, δ are negligible, n is polynomially-bounded, and $(1 - \frac{d}{m})^c$ is negligible (in the security parameter k). Assume that C_1 is (ε, a)-BQS-hiding and that C_2 is (δ, s_2)-online-extractable. Assume that the code with syndrome $\mathbf{S}$ has efficient error-correction.*

Then π^n_{COM} (a, nms_2)-BQS-UC-emulates $(\mathcal{F}^{A\to B,\ell}_{\text{COM}})^n$ for corrupted sender A.

Proof. First, we describe the structure of the real and the ideal model in the case that the party A (Alice) is corrupted:

In the real model, we have the environment $\mathcal{Z}$, the adversary Adv, the corruption party A^C, and the honest party B (Bob). The adversary controls the corruption party A^C, so effectively he controls the communication between Alice and Bob. The environment gets Bob's outputs `committed` and $(\texttt{open}, v)$. See Figure 4 (a).

In the ideal model, we have the environment $\mathcal{Z}$, the simulator Sim (to be defined below), the corruption party A^C, the dummy-party $\tilde{B}$, and the commitment functionality $\mathcal{F}_{\text{COM}}$. The inputs $(\texttt{commit}, v)$ and `open` of $\mathcal{F}_{\text{COM}}$ are provided by the corruption party A^C and thus effectively by the simulator Sim. The environment $\mathcal{Z}$ controls the dummy-party $\tilde{B}$ and hence gets the outputs `committed` and $(\texttt{open}, v)$ of $\mathcal{F}_{\text{COM}}$. See Figure 4 (b).

Fix an adversary Adv. To show Lemma 9, we need to find a quantum-polynomial-time simulator Sim with QM(Sim) $\leq nms_2$ such that, for any environment $\mathcal{Z}$ with QM($\mathcal{Z}$) + QM(Adv) $\leq a$, the real model and the ideal model are negligible-close. This simulator is described in Figure 5. Note that Sim is quantum-polynomial-time: The extractor B_S is quantum-polynomial-time

Commit phase:

- Sim picks a random $T \subseteq \{1, \dots, m\}$ with $\#T = c$. Then Sim (honestly) commits to T using C_1.
- When Alice commits to $R_1, \dots, R_m$, the simulator runs the extractor B_S for C_2 instead of the honest recipient's program. Let $R'_1, \dots, R'_m$ denote the extracted values.
- Sim waits for (F, σ, p) from Alice.
- Sim computes an $\underline{R}^* \in \{0,1\}^{mb}$ with $\mathbf{S}(\underline{R}^*) = \sigma$ and $\omega(\underline{R}', \underline{R}^*) \le (d-1)/2$ (remember that ω is the *block-wise* Hamming distance), computes $v' := p \oplus F(\underline{R}^*)$, and sends $(\texttt{commit}, v')$ to $\mathcal{F}_{\mathrm{COM}}$. (If no such $\underline{R}^*$ exists, we set $v' := \perp$.)

Open phase:

- Sim waits for $\underline{R}$ from Alice.
- Sim (honestly) opens T using C_1.
- For each $i \in T$, Sim waits for Alice to open R_i using C_2.
- Sim checks that the values R_i sent by Alice match the values R_i opened by Alice for all $i \in T$, and that $\sigma = \mathbf{S}(R_1 \| \dots \| R_m)$.
- Sim sends $\texttt{open}$ to $\mathcal{F}_{\mathrm{COM}}$.

Fig. 5. Simulator Sim for the case of corrupted Alice. The program described in this figure is executed for each instance of the n instances of π_{COM}. Communication with Alice is sent to an internally simulated instance of the adversary Adv.

by definition, and computing $\underline{R}^*$ is possible in polynomial-time because the code with syndrome $\mathbf{S}$ has efficient error-correction. Since C_2 is (δ, s_2)-online-extractable and Sim uses m instances of B_S per copy of B, $\mathrm{QM}(\mathrm{Sim}) \le nms_2$. We use the abbreviations $\underline{R} := R_1 \| \dots R_m$ and similarly for $\underline{R}'$ and $\underline{R}^*$.

Before we proceed, we introduce two variants of the honest recipient B. The machine B^* behaves like B, but when Alice commits to $R_1, \dots, R_m$ using C_2, B^* runs the extractor B_S for C_2 instead of the honest recipient's program. Call the extracted values $R'_1, \dots, R'_m$. Further, B^* computes an $\underline{R}^*$ with $\mathbf{S}(\underline{R}^*) = \sigma$ and $\omega(\underline{R}', \underline{R}^*) \le (d-1)/2$ and then computes $v' := p \oplus F(\underline{R}^*)$. (If no such $\underline{R}^*$ exists, $v' := \perp$.) In the open phase, B^* behaves like B. In particular, B^* outputs $(\texttt{open}, v)$, not $(\texttt{open}, v')$. That is, v' is computed but never used.

The machine B^+ behaves like B^*, but outputs $(\texttt{open}, v')$ instead of $(\texttt{open}, v)$.

By definition of online-extractability, and since B^* does not use the value extracted by B_S, we have that B and B^* are δ-indistinguishable. More precisely, for any network S, we have that $S \cup \{B\}$ and $S \cup \{B^*\}$ are δ-close. Since online-extractability was defined with respect to non-memory bounded adversaries, this holds even if S is not memory bounded.

As in Lemma 8, we proceed by investigating a sequence of games.

Game 1. In the game Game 1_j, the j-th instance of B is replaced by an instance of B^*. (Note: only one instance is replaced, not the first j instances.) $\diamond$

We use the following claim:

Claim 2 *Let S be an a-memory bounded network. In an execution of $S \cup \{B^*\}$, let v, v' denote the values v, v' computed by B^*. Then* $\Pr[v \notin \{v', \perp\}] \le \varepsilon + (1 - \frac{d}{m})^c := \eta$.

We prove this claim below. Since C_1 and C_2 are 0-memory bounded, we have that the machine B is 0-memory bounded. $\mathrm{QM}(\mathcal{Z}) + \mathrm{QM}(\mathrm{Adv}) \leq a$. Thus we can apply Claim 2 to Game 1_j. Hence in Game 1_j, $\Pr[v_j \notin \{v'_j, \bot\}] \leq \eta$ where v_j, v'_j are the values v, v' computed by B^*. We write $v_j := \bot$ if the open phase fails or does not take place (and hence v_j is not computed by B^*).

Game 2. This game is defined like the real model, except that we use n instances of B^* instead of the n instances of B. ◇

Using the fact that B and B^* are δ-indistinguishable, we get that the real model and Game 2 are $n\delta$-close.

Again using that B and B^* are δ-indistinguishable, we get that $\big|\Pr[v_j \notin \{v'_j, \bot\} : \text{Game } 1_j] - \Pr[v_j \notin \{v'_j, \bot\} : \text{Game } 2]\big| \leq (n-1)\delta$. Thus $\Pr[v_j \notin \{v'_j, \bot\} : \text{Game } 2] \leq \eta + (n-1)\delta$. Since this holds for all $j = 1, \dots, n$, we get:

$$\Pr[\exists j.\ v_j \notin \{v'_j, \bot\} : \text{Game } 2] \leq n\eta + n(n-1)\delta. \tag{1}$$

Game 3. This game is defined like the real model, except that we use n instances of B^+ instead of the n instances of B. ◇

Notice that Game 2 and Game 3 only differ in the fact that in Game 2 we use instances of B^* and in Game 3 instances of B^+. By definition, B^* and B^+ only differ in the value they output: B^* outputs $(\mathtt{open}, v)$ and B^+ outputs $(\mathtt{open}, v')$. By (1), the probability that the values v, v' are different in some instance of B^* is bounded by $n\eta + n(n-1)\delta$. Hence Game 2 and Game 3 are $(n\eta + n(n-1)\delta)$-close.

Finally, note that by construction of Sim, Game 3 and the ideal model are perfectly close. Thus the real and the ideal model are γ-close with $\gamma := n\delta + n\eta + n(n-1)\delta = n^2\delta + n\varepsilon + n(1 - \frac{d}{m})^c$. Since δ, ε are negligible, n is polynomially-bounded, and $(1 - \frac{d}{m})^c$ is negligible, we have that γ is negligible. Thus π^n_{COM} (a, nms_2)-BQS-UC-emulates $(\mathcal{F}^{A \to B, \ell}_{\mathrm{COM}})^n$ in the case of corrupted Alice.

Proof of Claim 2. Let $\underline{R}$, $\underline{R}'$, $\underline{R}^*$ and T denote the corresponding values as computed by B^*. We abbreviate $R_T := (R_i)_{i \in T}$ and $R'_T := (R'_i)_{i \in T}$. By *Bad* we denote the event that $R_T = R'_T$ and $\mathbf{S}(\underline{R}) = \sigma$ and $\underline{R} \neq \underline{R}^*$. By construction of B^*, $v \neq \bot$ implies $R_T = R'_T$ and $\mathbf{S}(\underline{R}) = \sigma$. And $v \notin \{v', \bot\}$ implies $\underline{R} \neq \underline{R}^*$. Thus $v \notin \{v', \bot\}$ implies *Bad*. Therefore, to show Claim 2, it is sufficient to show $\Pr[Bad] \leq \eta$ in $S \cup \{B^*\}$. To show this, we again proceed using a sequence of games:

Game 4. An execution of $S \cup \{B^*\}$. ◇

Game 5. We change B^* to halt after receiving $\underline{R}$ from Alice. ◇

Then $\Pr[Bad : \text{Game } 4] = \Pr[Bad : \text{Game } 5]$.

Game 6. We change B^* to commit to some (arbitrary) fixed value T_0 instead of committing to T. ◇

We wish to apply the (ε, a)-BQS-hiding property of C_1 in order to show that $\big|\Pr[Bad : \text{Game } 5] - \Pr[Bad : \text{Game } 6]\big| \leq \varepsilon$. Let B_1 denote the sender in the commitment scheme C_1. By definition, to commit to T (or T_0), B^* internally runs B_1. We construct an adversary A_1' that interacts with B_1. This adversary simulates $S \cup \{B^*\}$ (with B^* as in Game 5) except for the machine B_1 inside B^*. Note that in Game 5, only the commit phase of C_1 is executed. We let A_1' output 1 iff *Bad* happens. We define $\hat{B}_1$ like B_1, except that $\hat{B}_1$ ignores its input and commits to T_0. Let P be the probability that A_1' outputs 1 when running with B_1, and let $\hat{P}$ be the probability that A_1' outputs 1 when running with $\hat{B}_1$. By construction, $P = \Pr[\mathit{Bad} : \text{Game } 5]$ and $\hat{P} = \Pr[\mathit{Bad} : \text{Game } 6]$. Thus we only have to show that $|P - \hat{P}| \leq \varepsilon$. To apply the (ε, a)-BQS-hiding property of C_1 we have to check that A_1' is a-memory bounded. A_1' simulates $\mathcal{Z}$, Adv, and B^*. We have $\mathrm{QM}(\mathcal{Z}) + \mathrm{QM}(\mathrm{Adv}) \leq a$ by assumption. But B^* contains the extractor B_S for C_2 which uses additional s_2 qubits of quantum memory. Yet, B_S is executed after the end of the commit phase of C_1. That is, B^* is executed within a single activation of A_1' (since B_1 is not activated any more after the commit phase). Note that, although A_1' might use more than a qubits during the activation in which B^* is simulated, it stores at most a qubits between activations. Thus A_1' is a-memory bounded (remember that our definition of "a-memory bounded" on page 4 only requires that the memory bound holds between activations). Hence $|P - \hat{P}| \leq \varepsilon$ and thus $\big|\Pr[\mathit{Bad} : \text{Game } 5] - \Pr[\mathit{Bad} : \text{Game } 6]\big| \leq \varepsilon$.

Game 7. We change B^* to choose T only after receiving $\underline{R}'$. ◇

Since T is not used earlier by B^*, $\Pr[\mathit{Bad} : \text{Game } 6] = \Pr[\mathit{Bad} : \text{Game } 7]$. Fix values $\underline{R}$, $\underline{R}'$ and σ with $\mathbf{S}(\underline{R}) = \sigma$. We distinguish two cases, depending on whether there exists an $\underline{R}^*$ with $\mathbf{S}(\underline{R}^*) = \sigma$ and $\omega(\underline{R}^*, \underline{R}') \leq (d-1)/2$. Case "$\underline{R}^*$ exists": Since $\mathbf{S}$ is the syndrome of a b-block (m, κ, d)-linear code, $\mathbf{S}(\underline{R} - \underline{R}^*) = 0$, hence $\underline{R} - \underline{R}^*$ is a codeword. Hence $\underline{R} = \underline{R}^*$ or $\omega(\underline{R}, \underline{R}^*) \geq d$. Using the triangle inequality and $\omega(\underline{R}^*, \underline{R}') \leq (d-1)/2$, we get that $\underline{R} = \underline{R}^*$ or $\omega(\underline{R}, \underline{R}') \geq d - (d-1)/2 \geq d/2$. Case "$\underline{R}^*$ does not exist": Since no $\underline{R}^*$ with $\mathbf{S}(\underline{R}^*) = \sigma$ and $\omega(\underline{R}^*, \underline{R}') \leq (d-1)/2$ exists, and since $\mathbf{S}(\underline{R}) = \sigma$, we have that $\omega(\underline{R}, \underline{R}') > (d-1)/2$. Hence $\omega(\underline{R}, \underline{R}') \geq d/2$.

Thus, for any fixed choice of $\underline{R}, \underline{R}', \sigma$, we have $\mathbf{S}(\underline{R}) \neq \sigma$ or $\underline{R} = \underline{R}^*$ or $\omega(\underline{R}, \underline{R}') \geq d/2$.

If $\underline{R} = \underline{R}^*$ or if $\mathbf{S}(\underline{R}) \neq \sigma$, the event *Bad* does not occur by definition.

If $\omega(\underline{R}, \underline{R}') \geq d/2$, we bound the probability of *Bad* occurring as follows: Let $D := \{i : R_i \neq R_i'\}$. Then, for random $T \subseteq [m]$ with $\#T = c$, we have $\Pr[\mathit{Bad}] \leq \Pr[R_T = R_T'] = \Pr[T \cap D = \varnothing] \leq (1 - \frac{\#D}{m})^c \leq (1 - \frac{d}{m})^c$.

Thus for any fixed $\underline{R}, \underline{R}', \sigma$ we have $\Pr[\mathit{Bad}] \leq (1 - \frac{d}{m})^c$. By averaging over the choice of $\underline{R}, \underline{R}', \sigma$, we get $\Pr[\mathit{Bad} : \text{Game } 7] \leq (1 - \frac{d}{m})^c$.

Summarizing, we have $\Pr[\mathit{Bad} : \text{Game } 4] \leq \varepsilon + (1 - \frac{d}{m})^c = \eta$. This shows Claim 2.

□

Using Reed-Solomon codes for $\mathbf{S}$, and the extractable commitments from Theorem 6 for C_1 and C_2, we can instantiate the parameters of π_{COM} to satisfy the conditions of Lemmas 8 and 9. Thus we get the following theorem:

Theorem 10. *Let ℓ, n, and a be polynomially-bounded. Then there are choices for the parameters of π_{COM} and a polynomially-bounded integer s such that π_{COM} is polynomial-time, constant-round and π^n_{COM} (a,s)-BQS-UC-emulates $(\mathcal{F}^{A\to B,\ell}_{\mathrm{COM}})^n$.*

General two-party computation. By combining known results [19,10,15], we get a constant-round protocol that quantum-UC-emulates any two-party functionality and uses only commitments from Alice to Bob. Combining this with our protocol π_{COM}, we get our final result (for details see the full version [16]):

Theorem 11 (BQS two-party computation). *Let $\mathcal{G}$ be a classical well-formed*[11] *probabilistic-polynomial-time functionality. Let n and a be polynomially-bounded. Then there is a polynomially-bounded s and a constant-round* 0*-memory bounded protocol π_{bqs2pc} not invoking any functionality such that π^n_{bqs2pc} (a,s)-BQS-UC-emulates $\mathcal{G}^n$.*

Acknowledgements. We thank Christian Schaffner and Jürg Wullschleger for valuable discussions. This work was funded by the Cluster of Excellence "Multimodal Computing and Interaction".

References

1. Ben-Or, M., Mayers, D.: General security definition and composability for quantum & classical protocols (Sep 2004), online available at `http://xxx.lanl.gov/abs/quant-ph/0409062`
2. Bennett, C.H., Brassard, G.: Quantum cryptography: Public-key distribution and coin tossing. In: IEEE International Conference on Computers, Systems and Signal Processing 1984. pp. 175–179. IEEE Computer Society (1984)
3. Bennett, C.H., Brassard, G., Crépeau, C., Skubiszewska, M.H.: Practical quantum oblivious transfer. In: Crypto '91. LNCS, vol. 576, pp. 351–366. Springer (1991)
4. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: FOCS 2001. pp. 136–145. IEEE Computer Society (2001), full and revised version is [5]
5. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. IACR ePrint Archive (Jan 2005), full and revised version of [4], online available at `http://eprint.iacr.org/2000/067.ps`
6. Damgård, I., Fehr, S., Salvail, L., Schaffner, C.: Cryptography in the bounded quantum-storage model. In: FOCS 2005. pp. 449–458 (2005), a full version is available at `http://arxiv.org/abs/quant-ph/0508222`
7. Dziembowski, S., Maurer, U.: On generating the initial key in the bounded-storage model. In: Cachin, C., Camenisch, J. (eds.) Advances in Cryptology, Proceedings of EUROCRYPT '04. Lecture Notes in Computer Science, vol. 3027, pp. 126–137. Springer-Verlag (2004), online available at `ftp://ftp.inf.ethz.ch/pub/crypto/publications/DziMau04b.pdf`

[11] Well-formedness describes certain technical restrictions stemming from the proof by Ishai et al. [10]: Whenever the functionality gets an input, the adversary is informed about the length of that input. Whenever the functionality makes an output, the adversary is informed about the length of that output and may decide when this output is to be scheduled.

8. Fehr, S., Schaffner, C.: Composing quantum protocols in a classical environment. In: TCC 2009. LNCS, vol. 5444, pp. 350–367. Springer (2009)
9. Håstad, J., Impagliazzo, R., Levin, L.A., Luby, M.: A pseudorandom generator from any one-way function. SIAM Journal on Computing 28(4), 1364–1396 (1999), full version online available at http://www.icsi.berkeley.edu/~luby/PAPERS/hill.ps
10. Ishai, Y., Prabhakaran, M., Sahai, A.: Founding cryptography on oblivious transfer – efficiently. In: CRYPTO 2008. LNCS, vol. 5157, pp. 572–591. Springer (2008), http://www.springerlink.com/content/0l5v1l524816u652/
11. Kilian, J.: Founding cryptography on oblivious transfer. In: STOC 1988. pp. 20–31. ACM (1988)
12. König, R., Wehner, S., Wullschleger, J.: Unconditional security from noisy quantum storage. arXiv:0906.1030v2 [quant-ph] (Jun 2009)
13. Mayers, D.: Unconditionally Secure Quantum Bit Commitment is Impossible. Physical Review Letters 78(17), 3414–3417 (1997), online available at http://arxiv.org/abs/quant-ph/9605044
14. Unruh, D.: Simulatable security for quantum protocols (Sep 2004), online available at http://arxiv.org/ps/quant-ph/0409125
15. Unruh, D.: Universally composable quantum multi-party computation. In: EUROCRYPT 2010. LNCS, Springer (2010), to appear, preprint on arXiv:0910.2912 [quant-ph]
16. Unruh, D.: Concurrent composition in the bounded quantum storage model. IACR ePrint 2010/229 (Feb 2011), full version of this paper
17. Wehner, S., Wullschleger, J.: Composable security in the bounded-quantum-storage model. In: ICALP 2008, track C. pp. 604–615. LNCS, Springer (2008), full version available at http://arxiv.org/abs/0709.0492v1
18. Wolf, S., Wullschleger, J.: Oblivious transfer is symmetric. In: Vaudenay, S. (ed.) EUROCRYPT 2006. Lecture Notes in Computer Science, vol. 4004, pp. 222–232. Springer (2006)
19. Wullschleger, J.: Oblivious-Transfer Amplification. Ph.D. thesis, ETH Zurich (March 2007), arXiv:cs/0608076v3 [cs.CR]