# Superposition Attacks on Cryptographic Protocols

Ivan Damgård, Jakob Funder, Jesper Buus Nielsen, Louis Salvail

Dept. of Computer Science, Aarhus University, Université de Montreal

**Abstract.** Attacks on cryptographic protocols are usually modeled by allowing an adversary to ask queries to an oracle. Security is then defined by requiring that as long as the queries satisfy some constraint, there is some problem the adversary cannot solve, such as compute a certain piece of information. Even if the protocol is quantum, the queries are typically classical, such as a choice of subset of players to corrupt. In this paper, we introduce a fundamentally new model of quantum attacks on protocols, where the adversary is allowed to ask several classical queries in quantum superposition. This is a strictly stronger attack than the standard one, and we consider the security of several primitives in this model. We show that a secret-sharing scheme that is secure with threshold $t$ in the standard model is secure against superposition attacks if and only if the threshold is lowered to $t/2$. This holds for all classical as well as a large class of quantum secret sharing schemes. We then consider zero-knowledge and first show that known protocols are not, in general, secure in our model by designing a superposition attack on the well-known zero-knowledge protocol for graph isomorphism. We then use our secret-sharing result to design zero-knowledge proofs for all of NP in the common reference string model. While our protocol is classical, it is sound against a cheating unbounded quantum prover and computational zero-knowledge even if the verifier is allowed a superposition attack. Finally, we consider multiparty computation and give a characterization of a class of protocols that can be shown secure, though not necessarily with efficient simulation. We show that this class contains non-trivial protocols that cannot be shown secure by running a classical simulator in superposition.

## 1 Extended Abstract

Attacks on cryptographic protocols are usually modeled by allowing an adversary to query an oracle, for instance the adversary specifies a subset of parties he wants to corrupt, and gets back their views of the protocol. Security is then defined by requiring that as long as the queries satisfy some constraint (for instance, the corrupted subset is not too large), there is some problem the adversary cannot solve, such as compute a certain piece of information. In previous work, such a choice of subset to corrupt is always done classically, even if the protocol is quantum.

Several previous works consider what happens to security of classical protocols if we allow the adversary to be quantum. The model usually considered is that the adversary is now a quantum machine, but otherwise plays exactly the same game as in a normal attack, i.e., he still communicates classically with the protocol he attacks. One example of this is the work of Watrous [Wat06], showing that a large class of zero-knowledge protocols are also zero-knowledge against a quantum verifier.

In this paper, we introduce a new model of quantum attacks on classical as well as quantum cryptographic protocols, where the adversary is allowed to ask several classical queries to the oracle in quantum superposition. In more concrete terms, we ask, for multiparty protocols: what happens if the adversary can be in superposition of having corrupted several different subsets? or, for zero-knowledge protocols: what happens if a quantum verifier can be in superposition of having issued several different challenges to the prover? As we will argue below, we believe such superposition attacks to be a valid physical concern, but they also form a very natural generalization from a theory point of view: in the literature on black-box quantum computing, quantum black-box access to a function is usually defined by extending classical black-box access such that queries are allowed to contain several inputs in superposition. Our superposition attacks extend conventional attacks in the same way.

At first sight, superposition attacks may seem rather exotic. However, it is not hard to see that in several scenarios, it is quite natural to consider these attacks:

Consider first protocols that handle quantum data, such as in several previous works on quantum secret sharing and multiparty computation (see, e.g., Ben-Or et al. [BCG$^+$05]). Such a protocol requires players to communicate quantum information and keep throughout the game a joint entangled quantum state that involves all the players. If the players can do this, we should assume that the adversary can do something of similar technological difficulty. It therefore seems fair to allow the adversary to be in superposition of having interacted with the quantum memory of different subsets of players, i.e., he may do a superposition attack. This can be seen as a natural final step that makes everything in the game be quantum.

We emphasize that we are of not claiming that all forms of corruption that may occur in practice can occur in superposition. For instance, bribing or coercing a human being is a macroscopic process that can hardly take place in superposition. However, corrupting a player can also mean that the adversary communicates with the player in a way that is physically different from what the implementation of the protocol expected, and in this way get more information about the private state of the player than he was supposed to. For instance, the effect of sending on a different frequency or a different number of particles than expected may be hard to predict and may depend heavily on the way the protocol is implemented. Several known attacks on quantum key distribution work in this way. Now, if the communication defined by the protocol is quantum in the first place, we see no reason why such an attack cannot be performed in superposition against different players.

Second, what about classical protocols? One might think that here, superposition attacks cannot be mounted. The argument would be that since honest players are classical, they would “automatically” do a measurement of anything they receive, thus forcing a collapse of any quantum state they are given. However, this may be a dangerous assumption in the future, considering the fact that classical computing equipment becomes smaller and approaches the quantum limit. If an adversary captures such a device containing a secret key, he may be able to cool it down, for instance, and get some quantum effects to happen during communication. On top of this, even honest players may in the future use quantum computing and communication, but may sometimes need to execute a classical protocol. Having to always do this on separate classical hardware would be an unpleasant limitation.

The best solution would therefore be to use protocols that are secure in both the classical and the quantum setting. However, as we shall see, we cannot hope that security in the classical setting implies security against superposition attacks. One solution would then be to explicitly require in the protocol that every message is measured[1]. Of course, such countermeasures can be used, but seem problematic from a practical point of view: an instruction to measure an incoming message makes little sense to someone implementing the protocol on a classical machine, and moving to quantum hardware, one must make sure that equipment is present to do the measurement. We believe a more elegant solution would be to ask for protocols that are truly “hardware-oblivious”, i.e., are secure in both a classical and a quantum setting, assuming only that the hardware communicates messages as specified in the protocol

*Contributions of the paper.* We first show that any classical secret-sharing scheme that is perfectly secure with threshold $t$ in the standard model is perfectly secure against superposition attacks if and only if the adversary's superposition is constrained to contain subsets of size at most $t/2$. If this condition is not satisfied, not only does perfect security fail, we show examples where the adversary may even learn the secret with certainty. We also consider quantum secret sharing schemes and show that the same results hold for a large class of schemes derived from classical linear secret sharing, this includes essentially all known schemes.

We then consider (classical) zero-knowledge protocols and first give strong evidence that known protocols are not, in general, secure against superposition attacks. We give such an attack on the well-known graph isomorphism protocol, showing how to extract from the prover the number of fixed points of the prover's secret permutation. A simple reduction shows that if this attack could be simulated, then there is an efficient quantum algorithm computing the isomorphism between two graphs, as long as the isomorphism is unique. Thus the protocol can only be zero-knowledge if graph isomorphism in most cases is easy to a quantum computer.

We then use our result on classical secret-sharing to construct zero-knowledge proofs for all of NP in the common reference string (CRS) model. While our protocol is classical, it is sound against a cheating

[1] or that a copy is taken of every message—from the adversary's point of view, such a copy can be assumed to be measured and this would collapse the superposition.

unbounded quantum prover and computational zero-knowledge against a quantum verifier, even if the verifier is allowed a superposition attack[2]. Since our simulation is straight-line, the protocol is also secure under concurrent composition. We stress that our construction does not make assumptions beyond what we need to protect against standard attacks, nor is it less efficient than known constructions. Therefore this construction is an affirmative answer to our question above: we can indeed have protocols that are oblivious to the type of hardware (classical or quantum) they are executed on.

Finally, we consider classical multiparty computation and we define a UC-style model for static and passive superposition attacks on classical MPC protocols. Given our result on secret-sharing schemes, it is natural to speculate that classical MPC protocols that are secure against $t$ corruptions, are secure against superposition attacks corrupting $t/2$ players. The situation turns out to be more complicated, however: We show that for the model that gives the adversary the most power (and hence is the most hostile to the simulator), simulation based security is not possible at all. However, putting a natural constraint on the adversary, we are able to give a characterization of a class of protocols that can be shown secure, though not necessarily with efficient simulation. We show that this class contains non-trivial protocols, where by non-trivial, we mean that although the protocol is secure against a classical attack, we can show that it cannot be proved secure against a superposition attack by simply running the classical simulator in superposition. The simulator that does exist is in some sense "more quantum" .

Whether more general positive results hold in this constrained model remains an open question. Likewise, the very natural question of security of *quantum* multiparty computation protocols against superposition attacks remains open. Note, however, that existing work on quantum multiparty computation is typically based on quantum secret sharing, where the adversary's choice of subset to corrupt is classical. The negative part of our result on secret sharing described above shows that such protocol are not necessarily secure against superposition attacks as they stand.

**This is the end of the extended abstract, the rest of the paper should be considered as an attachment.**

[2] Since we use the CRS model, the reader may ask why we do not use existing protocols for non-interactive zero-knowledge (NIZK), where the prover just sends a single message to the verifier. In this way, the adversary would not get a chance to do a superposition attack. However, the most general assumption under which NIZK is known to be possible with an efficient prover is existence of one-way trapdoor permutations. They in turn are only known to be realizable under assumptions that are easily broken by a quantum adversary, such as factoring. Therefore we do not consider NIZK a satisfactory solution.

## 2 Preliminaries

We will model players in protocols in two different ways: when we are not interested in computational limitations on parties, a player will be specified by a series of unitary transforms where the $i$'th transform is done on all qubits available to the party, after the i'th message has been received (in the form of a quantum register), and then some designated part of the storage is sent as the next outgoing message. We are limiting ourselves to perfect unitary transformation of the party's register because we are exactly considering the situation where an attacker manages to prevent coupling between the party and the environment.

In cases where we want to bound the computational complexity of a player, we consider players to be an infinite family of interactive quantum circuits, as in the model from [FS09], and then the complexity is the circuit size.

### 2.1 Running functions in superposition

Consider any function, $f : X \to Y$ and a register of qubits, $|\psi\rangle = \sum_x \alpha_x |x\rangle|0\rangle \in \mathcal{H}_X \otimes \mathcal{H}_Y$, where $\dim(\mathcal{H}_X) = |X|$ and $\dim(\mathcal{H}_Y) = |Y|$. To *run* $f$ on $|\psi\rangle$ means to apply the unitary transformation, $U_f$, such that $U_f \sum_x \alpha_x |x\rangle|0\rangle = \sum_x \alpha_x |x\rangle|f(x)\rangle$. In general the register in $\mathcal{H}_Y$, called the *response register*, can contain any superposition of values, not just 0. In this case, we have that, $U_f \sum_{x,a} \alpha_{x,a}|x\rangle|a\rangle = \sum_{x,a} \alpha_{x,a}|x\rangle|f(x) \oplus a\rangle$ where $\oplus$ is the bitwise xor.

## 3 Secret sharing

In (classical) secret sharing $n$ parties are sharing some secret value $s \in \mathbb{S}$ using randomness $r \in \mathcal{R}$, where $\mathbb{S}$ and $\mathcal{R}$ are the sets of possible secrets and randomness. We name the parties $P_1, \dots, P_n$. Let $[n] = \{1, \dots, n\}$. Each party, $P_i$, receives a *share* $v_i(s, r) \in \{0,1\}^k$, also called his *private view* . That is, $v_i : \mathbb{S} \times \mathcal{R} \to \{0,1\}^k$.

For $A \subset [n]$, let $v_A(s, r) = \{v_i(s, r)\}_{i \in A}$ be the string containing the concatenation of views for parties $P_i$ with $i \in A$. For convenience in the following we assume that each such string is padded, so that they have the same length regardless of the size of $A$. That is, $v_A : \mathbb{S} \times \mathcal{R} \to \{0,1\}^t$. An *adversary structure* $G$ is a family of subsets $G \subset 2^{[n]}$. A secret sharing scheme is perfectly secure against classical $G$-attacks if for any $A \in G$, the distribution of $v_A(s, r)$ does not depend on $s$. The adversary structure of the secret sharing scheme is the maximal $F \subseteq 2^{[n]}$ for which the scheme is perfectly secure against $F$ attacks. We will only consider so-called *perfect* secret-sharing schemes, where it holds for all $A \notin G$ that $v_A(s, r)$ uniquely determines the secret $s$. I.e., in a perfectly secure, perfect secret-sharing scheme a set of shares either carry no information on the secret or fully determines the secret.

We will model any passive attack on the scheme as a one-time query to an *corruption oracle*. The corruption oracle for a specific run of a secret sharing scheme $O(s, r, A) = v_A(s, r)$ is the function that for a specific choice of secret, randomness and set of parties, returns the private view of those parties. That is, $O : \mathbb{S} \otimes \mathcal{R} \otimes F \to \{0,1\}^t$.

### 3.1 Two-party bit sharing example

Before we present the full model for secret sharing we start with a small example. We consider the case of 2 parties sharing a single bit, $b \in \{0,1\}$, using a random bit, $r \in \{0,1\}$. Here $[n] = \{1,2\}$, $F = \{(1),(2)\}$, $v_1(b, r) = b \oplus r$, $v_2(b, r) = r$.

This scheme is of course secure against a classical attack, which we can model by giving an adversary one-time access to an oracle that will return one share ($r$ or $r \oplus b$) on request. However, it is well known from the Deutch-Jozsa algorithm that given quantum access to such an oracle, one can compute the xor of the two bits from only one query. Hence the scheme is not secure against a superposition attack. In the following we consider what happens in the general case.

### 3.2 Model for secret sharing

We will now give the full technical description of the model for superposition attacks on general secret sharing. To do this we first consider the state spaces needed to run the protocol and the attack on the protocol. First is the space that contains the shares for all the parties, $\mathcal{H}_{\mathsf{parties}}$. The state in the register for this space is unchanged throughout the attack and is

$$|parties\rangle_{\mathsf{p}} = \sum_{s\in\mathbb{S},r\in\mathcal{R}} \sqrt{p_s}\sqrt{p_r}\big|s,r,v_{[n]}(s,r)\big\rangle_{\mathsf{p}} = \sum_{s\in\mathbb{S},r\in\mathcal{R}} \sqrt{p_s}\sqrt{p_r}|s,r\rangle \bigotimes_{i=1}^{n} |v_i(s,r)\rangle \in \mathcal{H}_{\mathsf{parties}} ,$$

where $|s,r\rangle$ is the purification of the secret and randomness choice. This is purely for technical reasons and does not matter for the adversary as he never sees it (and hence they might as well be considered measured). Secondly is the space for the environment, $\mathcal{H}_{\mathsf{env}}$, which the adversary can use to choose his query and use as potential auxiliary register. The initial state for the environment is a general (pure) state,

$$|\psi\rangle_{\mathsf{e}} = \sum_{x} \alpha_x |x\rangle_{\mathsf{e}} \in \mathcal{H}_{\mathsf{env}} ,$$

where $x$ is in some set of arbitrary, but finite size. We will ommit this for readability.
Finally is the space holding the adversary's query to the corruption oracle, $\mathcal{H}_{\mathsf{query}}$. This is initially a 'blank' state,

$$|\omega\rangle_{\mathsf{q}} = |0,0\rangle_{\mathsf{q}} \in \mathcal{H}_{\mathsf{query}}.$$

The space for the entire model is hence, $\mathcal{H}_{\mathsf{total}} = \mathcal{H}_{\mathsf{parties}} \otimes \mathcal{H}_{\mathsf{env}} \otimes \mathcal{H}_{\mathsf{query}}$, and the intial state is,

$$|init\rangle_{\mathsf{t}} = \sum_{s\in\mathbb{S},r\in\mathcal{R}} \sqrt{p_s}\sqrt{p_r}\big|s,r,v_{[n]}(s,r)\big\rangle_{\mathsf{p}} \otimes \sum_{x} \alpha_x |x\rangle_{\mathsf{e}} \otimes |0,0\rangle_{\mathsf{q}} \in \mathcal{H}_{\mathsf{total}} .$$

The attack will be defined by two operations and an adversary structure, $F$. First the adversary needs to construct his query for the oracle. This includes choosing the superposition of subsets he will corrupt and associated values for the response registers. This is an arbitrary unitary operation. We will denote it, $U_{\text{QUERY}}^{\text{ADV},F}$,

$$U_{\text{QUERY}}^{\text{ADV},F} : \mathcal{H}_{\mathsf{env}} \otimes \mathcal{H}_{\mathsf{query}} \to \mathcal{H}_{\mathsf{env}} \otimes \mathcal{H}_{\mathsf{query}} .$$

After this unitary operation the state is

$$|query\rangle_{\mathsf{t}} = U_{\text{QUERY}}^{\text{ADV},F}|init\rangle_{\mathsf{t}} = \sum_{s\in\mathbb{S},r\in\mathcal{R}} \sqrt{p_s}\sqrt{p_r}\big|s,r,v_{[n]}(s,r)\big\rangle_{\mathsf{p}} \otimes \sum_{x,A\in F,a\in\{0,1\}^t} \alpha_{x,A,a}|x\rangle_{\mathsf{e}} \otimes |A,a\rangle_{\mathsf{q}} \in \mathcal{H}_{\mathsf{total}}$$

where we assume it is the identity on $\mathcal{H}_{\mathsf{parties}}$. Next the oracle, $O(s,r,A)$, is run. Let

$$U_O : \mathcal{H}_{\mathsf{parties}} \otimes \mathcal{H}_{\mathsf{query}} \to \mathcal{H}_{\mathsf{parties}} \otimes \mathcal{H}_{\mathsf{query}}$$

denote the unitary applying this function. The state afterwards is,

$$\begin{aligned}|final\rangle_{\mathsf{t}} &= U_O|query\rangle_{\mathsf{t}} \\ &= \sum_{s\in\mathbb{S},r\in\mathcal{R}} \sqrt{p_s}\sqrt{p_r}\big|s,r,v_{[n]}(s,r)\big\rangle_{\mathsf{p}} \otimes \sum_{x,A\in F,a\in\{0,1\}^t} \alpha_{x,A,a}|x\rangle_{\mathsf{e}} \otimes |A, a\oplus v_A(s,r)\rangle_{\mathsf{q}} \in \mathcal{H}_{\mathsf{total}} ,\end{aligned}$$

where we assume $U_O$ is padded with appropriate identities. Consider the final state the adversary sees for a specific secret, $s$,

$$\rho_s^{\text{ADV},F} = \sum_{r\in\mathcal{R}} p_r \big|\psi_r^{\text{ADV},F}\big\rangle\big\langle\psi_r^{\text{ADV},F}\big| ,$$

where $|\psi_r^{\text{ADV},F}\rangle = \sum_{x,A\in F,a\in\{0,1\}^t} \alpha_{x,A,a}|x\rangle_{\mathsf{e}} \otimes |A, a\oplus v_A(s,r)\rangle_{\mathsf{q}}$.

**Definition 1.** *A secret sharing scheme $S$ is* perfectly secure against superposition $F$-attacks *if, and only if, for all unitary matrices, $U_{\text{QUERY}}^{\text{ADV},F} : \mathcal{H}_{env} \otimes \mathcal{H}_{query} \to \mathcal{H}_{env} \otimes \mathcal{H}_{query}$ and all possible pairs of inputs, $s, s' \in \mathbb{S}$ it holds that $\rho_s^{\text{ADV},F} = \rho_{s'}^{\text{ADV},F}$.*

## 3.3 (In)security against Superposition Attacks on Classical Secret-Sharing

For an adversary structure $F$, we define $F^2 = \{A \mid \exists B, C \in F : A = B \cup C\}$.

**Theorem 1.** *Let $G$ be the classical adversary structure for $\mathcal{S}$. $\mathcal{S}$ is perfectly secure against superposition $F$-attacks if and only if $F^2 \subseteq G$.*

*Proof.* For the forward direction, consider the adversary's final state,

$$\begin{aligned}\rho_s^{\text{ADV}} &= \sum_{r \in \mathcal{R}} p_r |\psi_r^{\text{ADV}}\rangle\langle\psi_r^{\text{ADV}}| \\ &= \sum_{r \in \mathcal{R}, x, x', A, A' \in F, a, a' \in \{0,1\}^t} p_r \alpha_{x,A,a} \alpha^*_{x',A',a'} |x\rangle_{\mathsf{e}} \langle x'|_{\mathsf{e}} \otimes |A, a \oplus v_A(s,r)\rangle_{\mathsf{q}} \langle A', a' \oplus v_{A'}(s,r)|_{\mathsf{q}} .\end{aligned}$$

Now, for any fixed $A$, $A'$, $a$, $a'$ and $s$, consider the matrix $\sum_{r \in \mathcal{R}} p_r |A, a \oplus v_A(s,r)\rangle_{\mathsf{q}} \langle A', a' \oplus v_{A'}(s,r)|_{\mathsf{q}}$. The crucial observation now is that this matrix is in 1-1 correspondence with the joint distribution of $v_A(s,r)$ and $v_{A'}(s,r)$. Namely, its entries are indexed by pairs of strings $(\alpha, \beta)$, where $\alpha$, $\beta$ are strings of the same length. And furthermore the $(\alpha, \beta)$'th entry is the probability that the events $v_A(s,r) = \alpha \oplus a$ and $v_{A'}(s,r) = \beta \oplus a'$ occur simultaneously, where the probability is taken over a random $r$. Now, if $F^2 \subseteq G$, we have that $\mathcal{S}$ is perfectly secure against classical $F^2$-attacks. Therefore the joint distribution of $v_A(s,r)$ and $v_{A'}(s,r)$ does not depend on $s$, consequently each matrix

$$\sum_{r \in \mathcal{R}} p_r |A, a \oplus v_A(r,s)\rangle_{\mathsf{q}} \langle A', a' \oplus v_{A'}(s,r)|_{\mathsf{q}}$$

is independent of $s$ as well. Hence $\forall s, s' \in \mathbb{S} : \rho_s^{\text{ADV},F} = \rho_{s'}^{\text{ADV},F}$ as required.

For the only-if part, assume for contradiction that $F^2 \not\subseteq G$, i.e., there exist $A_0, A_1$ such that $A_0 \cup A_1 \notin G$. It follows that a secret shared using $\mathcal{S}$ is uniquely determined from shares in $A_0 \cup A_1$. Then consider the query $|\omega_\alpha\rangle = (|A\rangle|0\rangle + |A'\rangle|0\rangle)/\sqrt{2}$. By the same computation as above, we see that $\rho_s^{\text{ADV}}$ contains a submatrix of form $\sum_{r \in \mathcal{R}} |a_A \oplus v_A(s,r)\rangle\langle a_{A'} \oplus v_{A'}(s,r)|$, that corresponds to the joint distribution of shares in $A$ and $A'$. But since the secret is uniquely determined from these shares, it follows that this submatrix is different for different secrets, and hence we get that there exists a measurement with non-zero bias towards the secret, and so $\mathcal{S}$ is not perfectly secure against superposition $F$-attacks. This is exactly the result we saw in the small example in Section 3.1. $\square$

*Remark 1.* Note that the if-part of the theorem still holds (in a slightly weaker sense) if the privacy of the underlying secret-sharing sheme is only statistical, rather than perfect. To see this, consider the matrix $\sum_{r \in \mathcal{R}} p_r |A, a \oplus v_A(s,r)\rangle_{\mathsf{q}} \langle A', a' \oplus v_{A'}(s,r)|_{\mathsf{q}} - \sum_{r \in \mathcal{R}} p_r |A, a \oplus v_A(s',r)\rangle_{\mathsf{q}} \langle A', a' \oplus v_{A'}(s',r)|_{\mathsf{q}}$, for any two secrets $s, s'$. By statistical privacy of $\mathcal{S}$, the sum of the 1-norm of all entries is negligibly small in some security parameter. It follows that the same is true for $\rho_s^{\text{ADV}} - \rho_{s'}^{\text{ADV}}$. Then the trace norm is negligible as well, and hence the adversary can distinguish $s$ from $s'$ with only negligible advantage.

## 3.4 Simplified models for secret sharing

When formalizing superposition attacks on secret sharing we need to consider if we allow the adversary to put values of his choice into the response register. This provably makes a difference for the strength of the model, and both options can be justified from a physical perspective. We therefore cover both models.

In more detail, when the adversary runs a classical component in superposition, and sends it a message, then the reply will in general be a superposition. This opens the question of how the reply is delivered. In quantum information processing, it is customary that the result is xor'ed onto a response register $a$ supplied along with the input. I.e., for a function $f$ one is given a box which on classical input $|x\rangle|a\rangle$, the output is $|x\rangle|a \oplus f(x)\rangle$. This is convenient, as it is invertible, so the action of the box on a superposition is given by its actions on the classical inputs. This approach is reasonable in quantum information processing, as one is typically designing the boxes oneself. If $f$ is a database one can simply design the quantum version to supply the output by xor'ing it onto a response register. However, when considering an adversary attacking a protocol one may be able to argue that the adversary will not have enough control to be able to decide the *a priori* content of the response register. It may be more reasonable to assume that the "box" the adversary talks to will create the response register and return it to the adversary. We model this setting of *created response registers* by restricting the more general setting of *supplied response registers* by allowing only $a = 0$, in which case the adversary will always receive $|x\rangle|f(x)\rangle$. It is clear that Theorem 1 applies in this setting as well.

### 3.5 Attacks on Secret Sharing

Theorem 1, tells us that we cannot have perfect security if the condition $F^2 \subseteq G$ is violated, but nothing about how much information the adversary can actually gain on the secret in this case. Of course, if $F \not\subseteq G$, the adversary can trivially learn the secret. But if $F \subseteq G$ one might hope that the adversary only learns a small amount of information, and even that this could become negligible by increasing the amount of randomness used to create the shares. However, in the lemma below we show that, under a simple assumption on the secret sharing scheme, an attacker can distinguish between two possible secrets with considerable bias. The attack works even in the restricted setting of created response registers. The proof is found in Appendix A.

**Lemma 1.** *Let $G$ be the classical adversary structure for $\mathcal{S}$. If there exist two subsets, $A_0$, $A_1 \in G$ such that (1) $A_0 \cup A_1 \notin G$ and (2) any secret, $s \in \mathbb{S}$, combined with the shares in $A_0$ ($A_1$) uniquely determine the choice of randomness, $r \in \mathcal{R}$, then the following holds.*
*For any two secrets $s, s' \in \mathbb{S} : s \neq s'$, there exists a query (with $a = 0$) that will allow an adversary to distinguish between $s$ and $s'$ with probability $p_{guess}$, where $p_{guess} \geq \frac{3}{4}$.*

An example of such a scheme is a Shamir secret sharing scheme that is classically secure against $t$ corrupted players. Here any subset of $t$ players combined with the secret will give $t + 1$ points on a polynomial of degree at most $t$ and hence uniquely determine the choice of randomness.

## 4 Quantum Secret Sharing

In this section we study the security of *quantum* secret sharing against superposition attacks. The situation turns out to be more complicated than for the classical case, and security depends on the exact model for what the adversary is allowed to do.

The model we will use for quantum secret sharing is that a secret is a quantum state $|sec\rangle = \sum_{s \in \mathbb{S}} \alpha_s |s\rangle$, i.e., some superposition over the possible classical choices of secret. The secret sharing scheme is then a unitary transform that maps each basis state $|s\rangle$ plus an ancilla of appropriate size to a state

$$|\Phi_s\rangle = \sum_{v_1,\dots,v_n \in \{0,1\}^k} \alpha_{v_1,\dots,v_n} |v_1\rangle \cdots |v_n\rangle \ ,$$

where we think of the content of the $i$'th register as the share of the $i$'th player. A quantum secret sharing scheme is secure against adversary structure $F$, if any subset of shares corresponding to a subset $A \in F$ contains no information on the secret state, but any subset of shares $B$ where $B \notin F$ allows reconstruction of the secret. It follows from the no-cloning theorem that security can only hold if $F$ has the property that for any $B \notin F$, the complement $\bar{B}$ is in $F$.

An example of quantum secret sharing can be derived from Shamir's secret sharing scheme. Here we assume that $n = 2t+1$, and the adversary structure contains all subsets of size at most $t$, and $\mathbb{S} = \mathbb{F}_p$ for the finite field $\mathbb{F}_p$. We then define $|\Phi_s\rangle = \frac{1}{\sqrt{M}} \sum_{f \in P_s} |f(1)\rangle \cdots |f(n)\rangle$, where $P_s$ is the set of polyomials of degree at most $t$ with $f(0) = s$, and $M$ the number of such polynomials.

There are several ways to define what it means that an adversary gets access to some of the shares. The simplest form of attack we consider is called a *share capture attack*, where the adversary essentially steals a subset of the shares (or subsets of shares in superposition). Since it seems natural to assume that some evidence of the absence of shares would be left after this, we assume that the players whose shares are captured are left with erasure symbols $\perp$ instead of shares. Some notation to describe this more formally: we will let $\mathbf{v} = v_1, \ldots, v_n$ in the following, and $|\mathbf{v}_A\rangle$ will stand for the basis state $|\mathbf{v}_A\rangle = |w_1\rangle \cdots |w_n\rangle$, where $w_i = v_i$ if $i \in A$ and $w_i = \perp$ otherwise. Likewise, for the Shamir based example, we let $|f(A)\rangle = |u_1\rangle \cdots |u_n\rangle$ where $u_i = f(i)$ if $i \in A$ and $u_i = \perp$ otherwise.

In an $F$-share capture attack, the adversary prepares a query $\sum_{A \in F} \alpha_A |A\rangle |\perp\rangle \cdots |\perp\rangle$ where the last part of the register will contain the captured shares. Then a unitary transform $U$ is executed on the shares and the adversary's query. $U$ is specified by the following action on basis states for shares and player subsets:

$$U(|v_1\rangle \cdots |v_n\rangle |A\rangle |\perp\rangle \cdots |\perp\rangle) = |\mathbf{v}_{\bar{A}}\rangle |A\rangle |\mathbf{v}_A\rangle \ .$$

We define $U$ to act as the identity on all basis vectors not involved in the above expression. This clearly makes $U$ be unitary. In the actual attack, $U(|\Phi_s\rangle \sum_{A \in F} \alpha_A |A\rangle |\perp\rangle \cdots |\perp\rangle)$ is computed and the adversary is given the last two registers, containing the corrupted set(s) and the captured shares. We say that the scheme is secure against $F$-share capture attacks if it always holds that the state the adversary gets is independent of $|s\rangle$. By linearity, security trivially extends to superpositions over different $|s\rangle$'s.

Now the following proposition is easy to show:

**Proposition 1.** *Any quantum secret sharing scheme that is secure for adversary structure $F$ is also secure against $F$-share capture attacks.*

*Proof.* The state the adversary gets after the attack can be computed by taking the global state

$$U(|\Phi_s\rangle \sum_{A \in F} \alpha_A |A\rangle |\perp\rangle \cdots |\perp\rangle)$$

and tracing out the part that the players keep. Note that this part is a superposition over basis states of form $|\mathbf{v}_{\bar{A}}\rangle$. From such a state, the set $A$ is determined by the positions of the $\perp$'s. Note also that after we trace out the players, their part can be assumed to be measured without loss of generality. This will therefore collapse the adversary's superposition over subsets to some particular choice of $A$. It follows that the attack is equivalent to choosing subset $A \in F$ with probability $|\alpha_A|^2$ and asking for shares in this subset only. This reveals no information by the assumed security. □

A more interesting result emerges if we allow the adversary slightly more power. We will consider what we call a *capture-and-restore* attack. Here, the adversary prepares a query as before and $U$ is executed. Now the adversary does some local computation. For convenience (and without loss of generality), we assume that a unitary transform $V$ is executed which on input $|\mathbf{v}_A\rangle$ and ancilla $|\Psi\rangle$ outputs $V(|\mathbf{v}_A\rangle|\Psi\rangle) = |\mathbf{v}_A\rangle|\Psi_{v_A}\rangle$. Finally the adversary keeps the register containing $|\Psi_{v_A}\rangle$ and $U$ is executed again on the rest. Note here that $U = U^{-1}$, so the second transformation "puts the shares back in place". It seems hard to argue why the adversary would be limited to only one application of $U$, so there is good motivation to consider also this attack. We talk about security of $F$-capture and restore attacks in the same way as above. In this case, it turns out that superposition attacks help the adversary, and for the Shamir based scheme, we get a result similar to what we have for classical secret sharing:

**Theorem 2.** *Let $G$ be the adversary structure containing sets of size at most $t$, for which the Shamir-based secret sharing scheme is secure. Then this scheme is perfectly secure against $F$-capture and restore attacks if and only if $F^2 \subseteq G$.*

The proof is similar to the proof of Theorem 1 and can be found in Appendix A. The theorem generalizes easily to any quantum secret sharing scheme derived from a classical linear scheme. We conjecture it generalizes to any quantum scheme. However, the only other quantum schemes we are aware of are schemes that use quantum shares for classical data. For instance, the 4 Bell-states can be used to share 2 classical bits among two players who receive only one qubit each. This scheme can also be broken under a superposition attack.

## 5 Zero-Knowledge

In this section we first give evidence that standard ZK protocols are not, in general, secure against superposition attacks. We then present a construction in the CRS model that leads to ZK protocols for NP that are indeed secure against superposition attacks.

### 5.1 An Attack Against the Graph Isomorphism Protocol

Consider the well-known protocol for proving graph isomorphism: The common input is two graphs $(G_0, G_1)$ on $n$ nodes, where the prover $P$ knows a permutation $\pi$ such that $\pi(G_0) = G_1$. $P$ will send to the verifier $V$ a graph $H = \sigma(G_0)$ where $\sigma$ is a random permutation on $n$ nodes. $V$ sends a random bit $b$, and $P$ returns the permutation $\rho = \pi^b\sigma^{-1}$. Finally $V$ checks that $\rho(H) = G_b$.

This protocol is perfect zero-knowledge against a classical as well as a quantum verifier, but in the quantum case the proof assumes that the verifier is limited to sending a classical bit $b$ as challenge to the prover (see [Wat06] for a precise definition of zero-knowledge when the verifier is quantum). We consider what happens if a superposition attack is allowed, and we give a concrete attack that allows to extract non-trivial information from the prover. We will only consider input graphs for which the permutation $\pi$ is uniquely defined from $G_0, G_1$. This is mostly for simplicity, but note also that it is well-known that deciding whether the permutation is unique is equivalent to deciding graph isomorphism in the *general* case. Hence, assuming the latter is hard, algorithms trying to compute an isomorphism between $G_0$ and $G_1$ are unlikely to have an easier time when the permutation is unique.

To analyze what happens, we need to assume a concrete way in which $P$ communicates his permutation to $V$ in the final message. We will assume a natural method, namely to send $\rho$, $P$ sends $\rho(0), \dots, \rho(n-1)$ in the classical case. We generalize this to the quantum case in the standard way: $V$ supplies a register $|b\rangle|i_0\rangle_0 \cdots |i_{n-1}\rangle_{n-1}$ and $P$ returns $|b\rangle|i_0 + \rho_b(0) \bmod n\rangle_0 \cdots |i_{n-1} + \rho_b(n-1) \bmod n\rangle_{n-1}$, where $\rho_b$ is the correct answer to $b$. By adding the answer into the response register, we ensure that the operation of $P$ is unitary[3]. The proof of the result below is found in Appendix A. It works by first constructing a concrete attack that the verifier might execute and concluding that this attack would allow to decide if the prover's secret permutation has a fixed point or not. Then, if a simulator existed for such a verifier, a simple reduction[4] allows us to show:

**Theorem 3.** *If the graph isomorphism protocol is zero-knowledge against superposition attacks, then graph isomorphism can be computed efficiently by a quantum algorithm for all inputs where the permutation is uniquely defined.*

### 5.2 A Superposition-Secure Zero-Knowledge Protocol

In this section, we present a zero-knowledge proof for any NP problem in the common reference string model. The proof is sound for an unbounded prover (quantum or not) and is computationally zero-knowledge for a polynomially bounded quantum verifier, even if superposition attacks are allowed.

[3] It is an open question whether our result holds for the case of created response registers mentioned earlier.

[4] We are grateful to Elad Verbin for pointing this reduction out to us.

For the protocol, we need a commitment scheme with special properties: we require a *keyed* commitment scheme $\texttt{Commit}_{\texttt{pk}}$, where the corresponding public key $\texttt{pk}$ is generated by one of two possible key-generation algorithms: $\mathcal{G}_{\texttt{H}}$ or $\mathcal{G}_{\texttt{B}}$. For a key $\texttt{pkH}$ generated by $\mathcal{G}_{\texttt{H}}$, the commitment scheme $\texttt{Commit}_{\texttt{pkH}}$ is unconditionally hiding, whereas the other generator, $\mathcal{G}_{\texttt{B}}$, actually produces a key *pair* $(\texttt{pkB}, \texttt{sk})$, so that the secret key $\texttt{sk}$ allows to efficiently extract $m$ from $\texttt{Commit}_{\texttt{pkB}}(m, r)$, and as such $\texttt{Commit}_{\texttt{pkB}}$ is unconditionally binding. Furthermore, we require that keys $\texttt{pkH}$ and $\texttt{pkB}$ produced by the two generators are computationally indistinguishable, for any family of polynomial size quantum circuits. We call such a commitment scheme a *dual-mode* commitment scheme.[5] As a candidate for implementing such a system, we propose the public-key encryption scheme of Regev [Reg05], which is based on a worst-case lattice assumption and is not known to be breakable even by (efficient) quantum algorithms. Regev does not explicitly state that the scheme has the property we need, but this is implicit in his proof that the underlying computational assumption implies semantic security.[6]

## 5.3 The Model

We now describe the framework for our protocol: the proof system is specified w.r.t. a language $L$, and we have a prover $P$ and a verifier $V$, both are assumed classical (when playing honestly). They get as input a common reference string $CRS$ chosen with a prescribed distribution $\sigma$ and a string $x$. $P$ and $V$ interact and at the end $V$ outputs $accept$ or $reject$. The first two properties we require are standard: *Completeness:* if $x \in L$ and $P, V$ follow the protocol, $V$ outputs $accept$ with probability 1. *Soundness:* if $x \notin L$ (but $CRS$ is chosen according to $\sigma$) then for any prover $P^*$, $V$ outputs $accept$ with probability negligible (in the length of $x$) when interacting with $P^*$ on input $x$ and $CRS$.

For zero-knowledge, we extend the capabilities of a cheating verifier $V^*$ so it may do a superposition attack. For simplicity, we give our definition of superposition zero-knowledge only for 3-move public-coin protocols, i.e., conversations are assumed to have the form $(a, e, z)$, where $e$ is a random challenge issued by the verifier. It is not hard to extend the definition but the notation becomes more cumbersome. First, $V^*$ is assumed to be a quantum machine, and the protocol is executed as follows: $V^*$ receives $x, CRS$ and $P$'s first message $a$. Now, instead of sending a classical challenge $e$, $V^*$ is allowed to send a query $\sum_{e,y} \alpha_{e,y}|e\rangle|y\rangle$. We assume the the prover will process the query following his normal algorithm in superposition, so the verifier will get the same two registers back, in state $\sum_{e,y} \alpha_{e,y}|e\rangle|y + z(x, e, \rho)\rangle$, where $z(x, e, \rho)$ is $P$'s response to challenge $e$ on input $x$ and internal randomness $\rho$. Finally, $V^*$ outputs 0 or 1. Let $p_{real}(x)$ be the probability that 1 is output. We say that the proof system is *superposition zero-knowledge* if there exists a polynomial time quantum machine, the simulator $S$, such that the following holds for any cheating verifier $V^*$ and $x \in L$: $S$ interacts with $V^*$ on input $x$, and we let $p_{\text{SIM}}(x)$ be the probability that $V^*$ outputs 1. Then $|p_{real}(x) - p_{\text{SIM}}(x)|$ is negligible (in the length of $x$).

Note that, as usual in the CRS model, $S$ only gets $x$ as input and may therefore generate the reference string itself.

## 5.4 The Protocol

We now describe the basic ideas behind our protocol: we will let the CRS contain the following: $\texttt{pkB}$, $c = \texttt{Commit}_{\texttt{pkB}}(0), \texttt{pkB}'$, where the public keys are both generated by $\mathcal{G}_{\texttt{B}}$. Then, using a standard trick, we will let $P$ show that either $x \in L$ or $c$ contains a 1. Since of course the latter statement is false, $P$ still needs to convince us that $x \in L$. The simulator, on the other hand, can construct a reference string where $c$ does contain 1 and simulate by following the protocol. The CRS will look the same to the verifier so we just need that the change of witness used is not visible in the proof, i.e., the proof should be so-called *witness indistinguishable*. In this way, we can simulate without rewinding, and this allows $V^*$ to be quantum.

[5] The notions of dual-mode *cryptosystems* and of meaningful/meaningless encryptions, as introduced in [PVW08] and [KN08], are similar in spirit but differ slightly technically.

[6] The proof compares the case where the public key is generated normally to a case where it is chosen with no relation to any secret key. It is then argued that the assumption implies that the two cases are computationally indistinguishable, and that in the second case, a ciphertext carries essentially no information about the message. This argument implies what we need.

However, standard techniques for witness indistinguishability are not sufficient to handle a superposition attack. For this, we need to be more specific about the protocol: a first attempt (which does not give us soundness) is that $P$ will secret-share his witness $w$ (where for the honest prover, $w$ will be a witness for $x \in L$), to create shares $s_1, \ldots, s_n$ where we assume the scheme has $t$-privacy. Then $P$'s first message is a set of commitments $a = (\texttt{Commit}_{\texttt{pkB}'}(s_1, r_1), \ldots, \texttt{Commit}_{\texttt{pkB}'}(s_n, r_n))$. The verifier's challenge $e$ will point out a random subset of the commitments, of size $t/2$, and the prover opens the commitments requested. Intuitively, this is zero-knowledge by Theorem 1: since we limit the number of shares the verifier can ask for to half the threshold of the secret sharing scheme, the state $V^*$ gets back contains no information on the secret $w$.

On the other hand, this protocol is of course not sound, the verifier cannot check that the prover commits to meaningful shares of anything. To solve this, we make use of the "MPC in the head" technique from [IKOS09]: Here, we make use of an $n$-party protocol in which the witness $w$ is secret-shared among the players, and a multiparty computation is done to check whether $w$ is correct with respect to the claim on the the public input, namely in our case $x \in L$ or the $c$ from the $CRS$ contains 1. Finally all players output $accept$ or $reject$ accordingly. It is assumed that the protocol is secure against active corruption of $t$ players where $t$ is $\Theta(n)$. We will call this protocol $\pi_{L,CRS}$ in the following. Several examples of suitable protocols can be found in [IKOS09]. In their construction, the prover emulates an execution of $\pi$ in his head, and we let $v_{\pi_{L,CRS}}(i, \rho)$ denote the view of virtual player $i$, where $\rho$ is the randomness used. The prover then commits to $v_{\pi_{L,CRS}}(i, \rho)$, for $i = 1, \ldots, n$ and the verifier asks the prover to open $t$ randomly chosen views that are checked for consistency and adherence to $\pi_{L,CRS}$. It is shown in [IKOS09] that if no valid witness exists for the public input, then the verifier will detect an error with overwhelming probability.

Now, observe that the process of emulating $\pi$ can be thought of as a secret sharing scheme, where the prover's witness $w$ is shared and each $v_\pi(i, \rho)$ is a share: indeed any $t$ shares contain no information on $w$ by $t$-privacy of the protocol. Therefore combining this with our rudimentary idea from before gives us the solution.

**Superposition-secure zero-knowledge proof for any $NP$-language $L$.**
The public input is $x$, of length $k$ bits. The distribution $\sigma$ generates the common reference string as $\texttt{pkB}, c = \texttt{Commit}_{\texttt{pkB}}(0), \texttt{pkB}'$, where the public keys are both generated by $\mathcal{G}_{\texttt{B}}$ on input $1^k$.

1. The prover $P$ emulates $\pi_{L,CRS}$ to generate $v_{\pi_{L,CRS}}(i, \rho)$ and sends $\texttt{Commit}_{\texttt{pkB}'}(v_{\pi_{L,CRS}}(i, \rho), r_i)$, for $i = 1, \ldots, n$, to the verifier $V$.
2. $V$ sends a challenge $e$ designating a random subset of the commitments of size $t/2$.
3. $P$ opens the commitments designated by $e$, $V$ checks the opened views according to the algorithm described in [IKOS09], and accepts or rejects according to the result.

**Theorem 4.** *If $(\mathcal{G}_{\texttt{B}}, \mathcal{G}_{\texttt{H}}, \texttt{Commit})$ form a secure dual-mode commitment scheme, then the above protocol is complete, sound and superposition zero-knowledge.*

*Proof.* Completeness is trivial by inspection of the protocol. Soundness follows immediately from the soundness proof in [IKOS09], we just have to observe that the fact that the prover opens $t/2$ and not $t$ views makes no difference, in fact the proof holds as long as $\Theta(n)$ views are opened. For zero-knowledge, we describe a simulator $S$: It will generate a common reference string as $\texttt{pkH}, c = \texttt{Commit}_{\texttt{pkH}}(1), \texttt{pkH}'$ where both public keys are generated by $\mathcal{G}_{\texttt{H}}$ on inout $1^k$. It then plays the protocol with $V^*$, answering its quantum queries by following the protocol. This is possible since $c$ now contains a 1, so $S$ knows a valid witness. To show that $V^*$ cannot distinguish the simulation from the protocol, we define series of games

**Game 0** The protocol as described above, but where $P$ talks to $V^*$ doing a superposition attack.
**Game 1** As Game 0, but the CRS is generated as $\texttt{pkH}, c = \texttt{Commit}_{\texttt{pkH}}(0), \texttt{pkH}'$, where both public keys are generated by $\mathcal{G}_{\texttt{H}}$.
**Game 2** As Game 1, but the CRS is generated as $\texttt{pkH}, c = \texttt{Commit}_{\texttt{pkH}}(1), \texttt{pkH}'$.

**Game 3** As Game 2, but $P$ uses as witness the fact that $c$ contains a 1.

Now, Game 0 and Game 1 are computationally indistinguishable by assumption on the dual-mode commitment scheme. Game 1 and Game 2 are statistically indistinguishable by the fact that commitments done using $\mathtt{pkH}$ are statistically hiding. Game 2 and Game 3 are statistically indistinguishable by Theorem 1 and the fact that commitments done using $\mathtt{pkH}'$ are statistically hiding. In the last step one notes that if you take a secret-sharing scheme meeting the conditions of Theorem 1 and you augment each share with a commitment, under a statistically hiding commitment scheme, to all the other shares, then you obtain a secret-sharing scheme that still meets the conditions of Theorem 1, except that privacy is only statistical. Then you note that the protocol can be seen as using such an augmented secret-sharing scheme where the prover's witness is the secret, and apply Remark 1. Finally, note that Game 3 is exactly the same game as the simulation. $\square$

## 6 Multiparty computation

We give a short summary of our results on multiparty computation. The details can be found in Appendix B. To consider the security of multiparty computation under superposition attacks we define a UC-style model that captures security under static and passive attacks. We use a notion where the environment chooses inputs to and gets outputs from the parties, and also attacks the protocol, i.e., he may issue a query $|q\rangle$ where he asks to corrupt a subset of players, possibly several in superposition. In the real process, he gets directly $|q\rangle$ back where the views of the corrupted players have been added in, including their inputs and outputs, their randomness and all messages sent and received. In the ideal process, the query goes to a simulator, who sends it to an ideal functionality. The functionality only adds in the inputs and outputs of corrupt players and the simulator must patch in views that match. Finally the environment gets the patched view back, and we now demand that its final states in the real and the ideal process are indistinguishable. Again we can make a distinction between *created response registers*, where the views are returned in newly created registers, and *supplied response registers*, where the environment provides the registers in which the (patched) views should be returned. This gives rise to two distinct MPC models, which we call the *CRR model* and the *SRR model* below.

Our first result is that in the SRR model, one can construct settings where simulation seems to be impossible for purely technical reasons. Consider the dummy 4-party function $d(x_1, x_2, x_3, x_4) = (\lambda, \lambda, \lambda, \lambda)$, where $\lambda$ is the empty string, i.e., the function which gives no outputs on any of its inputs. Consider protocol $\delta$, where parties $P_2, P_3, P_4$ runs as follows: On input $x_i$, output $\lambda$ and terminate, and where $P_1$ runs as follows: On input $x_1$, create a random secret sharing $(s_1, \dots, s_4)$ of $x_1$, send $s_i$ to $P_i$ and then output $\lambda$ and terminate. If we pick a secret sharing scheme which classically tolerates 2 corrupted parties, then the secret sharing scheme is secure against corruption of 1 party under superposition attacks. We would therefore expect the protocol $\delta$ to be a secure implementation of $d$ against corruption of 1 party under superposition attacks. It turns out that $\delta$ is *not* a secure implementation of $d$ against 1 corruption in the SRR model. The reason is that the environment can put the supplied register in a uniform superposition over all values. Then when the inputs and outputs are added to the register by the ideal functionality it will have the same state regardless of $x_1$. So the simulator gets no information on $x_1$ when $P_1$ is corrupted and hence cannot simulate the shares that $P_1$ sends. The simulator could try to process the supplied register to get rid of the problem, but for every simulator there exists an environment that can anticipate what the simulator will do and can invert its operation. Technically our result only applies to simulators which act unitarily on the provided register, but it is hard to imagine that it should help in constructing a simulator that it is allowed to send a mixed state. The details are in (Appendix B.3).

We then show that in the CRR model, $\delta$ *is* indeed a secure evaluation of $d$ against corruption of 1 player, as we would expect; (The details are in Appendix B.7) Together, these results suggest that the "correct" model is the CRR model.

We then proceed to investigate general feasibility of secure function evaluation against superposition attacks in the CRR model. This problem, however, turns out to be far from trivial. As an example of this, suppose the protocol in question is classically secure against corruptions of sets of size $t$. Now, since the protocol is a classical process that in our model is run in superposition over the inputs, one could expect that we could get a simulator for a superposition attack with $t/2$ corruptions by running the classical simulator in superposition. This turns out to be false. Specifically, we show that even though the above protocol $\delta$ is a classical secure implementation of $d$ against $t = 2$ corruptions, it cannot be proven superposition attack secure against $t/2 = 1$ corruptions using a simulator which consists of running a classical simulator in superposition. The problem seems to come from the fact that in the MPC model, also the dealer, $P_1$, may be corrupted[7], which can be used to force the way to simulate the shares of different corrupt subsets to be inter-consistent, which in turn is the same a being able to simultaneously simulate a corruption of all other players than the dealer. But this is impossible, as they have three shares, which is sufficient to determine $x_1$. The details are in (Appendix B.4).

Not all hope is lost, however. The fact that simple classical-simulation-in-superposition is insufficient to prove superposition attack security does not rule out simulators which are "more quantum" than this. And, indeed, we can show that a large class of protocols can in fact be proven superposition attack secure, although the simulator may not always be efficient. For deterministic functions $f$ we give, in classical terms, a complete characterization of the class of classical MPC protocols which securely evaluate $f$ under superposition attack.

The characterization goes as follows. Let $f$ be a deterministic function, let $\pi$ be a protocol, let $A \subseteq \{1, \ldots, n\}$ denote a subset of corrupted parties, let $s = (s_1, \ldots, s_n)$ denote a vector of inputs for $\pi$ and let $\mathbb{S}$ denote the set of possible input vectors. For two input vectors $s$ and $s'$ let $F_{s,s'} = \{A \in F | s_A = s'_A \wedge f_A(s) = f_A(s')\}$ be the subset of allowed corruptions where the corrupted parties have the same inputs and outputs in $f$. Finally, let $r = (r_1, \ldots, r_n)$ denote a vector of random inputs for $\pi$ and let $\mathcal{R}$ denote the space of such vectors. Then it holds that the protocol $\pi$ is a perfectly secure evaluation of $f$ against superposition $F$-attacks in the CRR model iff it is correct and there exist a family of permutations, $\{\pi_{s,s',A} : \mathcal{R} \to \mathcal{R}\}_{s,s' \in \mathbb{S}, A \in F_{s,s'}}$ with the following two properties,

1. $\forall s, s' \in \mathbb{S}, \forall A \in F_{s,s'}, \forall r \in \mathcal{R} : |v_A(s, \pi_{s,s',A}(r))\rangle = |v_A(s', \pi_{s',s,A}(r))\rangle$.
2. $\forall s, s', s'' \in \mathbb{S}, \forall A \in F_{s,s'}, A' \in F_{s,s''}$ :
$$\sum_{r \in \mathcal{R}} |v_A(s, r)\rangle\langle v_{A'}(s, r)| = \sum_{r \in \mathcal{R}} |v_A(s, \pi_{s,s',A}(r))\rangle\langle v_{A'}(s, \pi_{s,s'',A'}(r))|.$$

(The proof is in in Appendix B.8.) Note that property (1) is exactly the statement that a (not necessarily efficient) simulator exists in the classical model. As an example of using the characterization, the proof in Appendix B.3 that the dummy protocol is superposition attack secure against 1 corruption is derived via the general characterization. Extending our characterization to probabilistic functions $f$ and making a characterization of when polynomial time simulation is possible are open problems.

## References

[BCG+05] Michael Ben-Or, Claude Crépeau, Daniel Gottesman, Avinatan Hassidim, and Adam Smith. Secure multiparty quantum computation with (only) a strict honest majority. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 249–260, 2005.

[CJW04] Anthony Chefles, Richard Jozsa, and Andreas Winter. On the existence of physical transformations between sets of quantum states. *International Journal of Quantum Information*, pages 11–21, 2004. http://arxiv.org/abs/quant-ph/0307227.

[FS09] Serge Fehr and Christian Schaffner. Composing quantum protocols in a classical environment. In *Theory of Cryptography Conference (TCC)*, volume 5444 of *Lecture Notes in Computer Science*, pages 350–367. Springer, 2009.

[IKOS09] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge proofs from secure multiparty computation. *SIAM J. Comput.*, 39(3):1121–1152, 2009.

[KN08] Gillat Kol and Moni Naor. Games for exchanging information. In *Theory of Cryptography Conference (TCC)*, volume 4948 of *Lecture Notes in Computer Science*, pages 423–432. Springer, 2008.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In *Advances in Cryptology—CRYPTO '08*, volume 5157 of *Lecture Notes in Computer Science*, pages 554–571. Springer, 2008.

[7] (This is in contrast to the pure secret-sharing model where only shareholders can be corrupted.)

[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.

[Wat06] John Watrous. Zero-knowledge against quantum attacks. In *38th Annual ACM Symposium on Theory of Computing (STOC)*, pages 296–305, 2006. full version available at http://www.cs.uwaterloo.ca/ watrous/papers.html.

## A Proofs

*Proof (of Lemma 1).* The adversary will need no auxiliary register, so let his state simply be the query register. He constructs the following (pure state) query

$$|\omega\rangle = \frac{1}{\sqrt{2}}(|A_0, 0\rangle + |A_1, 0\rangle) ,$$

where $|\omega\rangle \in \mathcal{H}_{\mathsf{query}}$ and $A_0 \neq A_1$ are subsets chosen according to the lemma. The final state the adversary sees for different secrets is then

$$\rho_s^{\mathrm{ADV}} = \sum_{r\in\mathcal{R}} p_r |\psi_r^{\mathrm{ADV}}\rangle\langle\psi_r^{\mathrm{ADV}}| ,$$

where $|\psi_r^{\mathrm{ADV}}\rangle = \sum_{A\in\{A_0,A_1\}} \frac{1}{\sqrt{2}} |A, v_A(s,r)\rangle_{\mathsf{q}}$ and $r \in \mathcal{R}$.

It is well-known that the adversary's probability of distinguishing between two such states, $\rho^s_{\mathrm{ADV}}$ and $\rho^{s'}_{\mathrm{ADV}}$, is $p_{guess} = \frac{1}{2} + \frac{1}{4} \times |\rho^s_{\mathrm{ADV}} - \rho^{s'}_{\mathrm{ADV}}|_{\mathrm{tr}}$, where $|\ldots|_{\mathrm{tr}}$ denotes the trace norm. Define the difference between the two states as the matrix $\varDelta$.

$$\begin{aligned}
\varDelta &= \rho^s_{\mathrm{ADV}} - \rho^{s'}_{\mathrm{ADV}} \\
&= \frac{1}{2} \sum_{A,A'\in\{A_0,A_1\},r\in\mathcal{R}} p_r |A, v_A(s,r)\rangle_{\mathsf{q}} \langle A', v_{A'}(s,r)|_{\mathsf{q}} - \frac{1}{2} \sum_{A,A'\in\{A_0,A_1\},r\in\mathcal{R}} p_r |A, v_A(s',r)\rangle_{\mathsf{q}} \langle A', v_{A'}(s',r)|_{\mathsf{q}} \\
&= \frac{1}{2} \sum_{A,A'\in\{A_0,A_1\}} |A\rangle\langle A'| \left( \sum_{r\in\mathcal{R}} p_r |v_A(s,r)\rangle_{\mathsf{q}} \langle v_{A'}(s,r)|_{\mathsf{q}} - \sum_{r\in\mathcal{R}} p_r |v_A(s',r)\rangle_{\mathsf{q}} \langle v_{A'}(s',r)|_{\mathsf{q}} \right) .
\end{aligned}$$

Since the distribution of $v_A(s,r)$ (for $A \in \{A_0, A_1\}$) is independent of the secret we have that for all $s, s' \in \mathbb{S}$

$$\sum_{r\in\mathcal{R}} |v_A(s,r)\rangle\langle v_A(s,r)| = \sum_{r\in\mathcal{R}} |v_A(s',r)\rangle\langle v_A(s',r)|,$$

which means we only need to consider $A, A' \in \{A_0, A_1\}$ where $A \neq A'$. So,

$$\begin{aligned}
\varDelta = &\frac{1}{2} |A_0\rangle\langle A_1| \otimes \left( \sum_{r\in\mathcal{R}} p_r |v_{A_0}(s,r)\rangle_{\mathsf{q}} \langle v_{A_1}(s,r)|_{\mathsf{q}} - \sum_{r\in\mathcal{R}} p_r |v_{A_0}(s',r)\rangle_{\mathsf{q}} \langle v_{A_1}(s',r)|_{\mathsf{q}} \right) \\
&+ \frac{1}{2} |A_1\rangle\langle A_0| \otimes \left( \sum_{r\in\mathcal{R}} p_r |v_{A_1}(s,r)\rangle_{\mathsf{q}} \langle v_{A_0}(s,r)|_{\mathsf{q}} - \sum_{r\in\mathcal{R}} p_r |v_{A_1}(s',r)\rangle_{\mathsf{q}} \langle v_{A_0}(s',r)|_{\mathsf{q}} \right) .
\end{aligned}$$

Now define the two submatrices,

$$S = \sum_{r\in\mathcal{R}} p_r |v_{A_0}(s,r)\rangle_{\mathsf{q}} \langle v_{A_1}(s,r)|_{\mathsf{q}} - \sum_{r\in\mathcal{R}} p_r |v_{A_0}(s',r)\rangle_{\mathsf{q}} \langle v_{A_1}(s',r)|_{\mathsf{q}} \tag{1}$$

$$S^\dagger = \sum_{r\in\mathcal{R}} p_r |v_{A_1}(s,r)\rangle_{\mathsf{q}} \langle v_{A_0}(s,r)|_{\mathsf{q}} - \sum_{r\in\mathcal{R}} p_r |v_{A_1}(s',r)\rangle_{\mathsf{q}} \langle v_{A_0}(s',r)|_{\mathsf{q}}$$

such that $\Delta$ is the $2 \times 2^t$ by $2 \times 2^t$ matrix

$$\Delta = \frac{1}{2}\begin{pmatrix} 0 & S \\ S^\dagger & 0 \end{pmatrix} . \tag{2}$$

It is well-known that if $\Delta$ is of the form (2) and $\frac{1}{2}S$ has singular values $s_1 \geq \cdots \geq s_p$ then $\Delta$ has eigenvalues $\pm s_1, \ldots, \pm s_p$. Since $\Delta$ is Hermitian, the trace norm is the sum of the absolute eigenvalues. From this we conclude that $|\Delta|_{\mathrm{tr}} = |S|_{\mathrm{tr}}$ and we can reduce the problem to that of finding the trace norm of S. Let

$$\begin{aligned} S &= M_s - M_{s'} \\ M_s &= \sum_{r \in \mathcal{R}} p_r |v_{A_0}(s,r)\rangle\langle v_{A_1}(s,r)| \, . \end{aligned}$$

Note that for the state to be normalized it must be that

$$\sum_{i,j \in \{0,1\}^t} [M_s]_{i,j} = \sum_{i,j \in \{0,1\}^t, r} p_r \delta_{v_{A_0}(s,r),i} \delta_{v_{A_1}(s,r),j} = 1 \, .$$

Now, define the matrix, $\tilde{M}_s$,

$$\tilde{M}_s = \sum_{r \in \mathcal{R}} |v_{A_0}(s,r)\rangle\langle v_{A_1}(s,r)| + \sum_{i=|\mathcal{R}|}^{2^t-1} |i\rangle\langle i| \, .$$

It's straightforward to check that $\tilde{M}_s \tilde{M}_s^T = \mathbb{I}$,

$$\begin{aligned} &\left( \sum_{r \in \mathcal{R}} |v_{A_0}(s,r)\rangle\langle v_{A_1}(s,r)| + \sum_{i=|\mathcal{R}|}^{2^t-1} |i\rangle\langle i| \right) \times \left( \sum_{r' \in \mathcal{R}} |v_{A_1}(s,r')\rangle\langle v_{A_0}(s,r')| + \sum_{i=|\mathcal{R}|}^{2^t-1} |i\rangle\langle i| \right) \\ &\qquad = \sum_{r,r' \in \mathcal{R}} |v_{A_0}(s,r)\rangle\langle v_{A_0}(s,r')| \langle v_{A_1}(s,r)| |v_{A_1}(s,r')\rangle + \sum_{i=|\mathcal{R}|}^{2^t-1} |i\rangle\langle i| \\ &\qquad = \sum_{r \in \mathcal{R}} |v_{A_0}(s,r)\rangle\langle v_{A_0}(s,r)| + \sum_{i=|\mathcal{R}|}^{2^t-1} |i\rangle\langle i| = \mathbb{I} \, , \end{aligned}$$

where we used that both $|v_{A_0}(s,r)\rangle$ and $|v_{A_1}(s,r)\rangle$ has a 1-1 correspondence to $r \in \mathcal{R}$ (for fixed $s$). Note that $\sum_{i=|\mathcal{R}|}^{2^t-1} |i\rangle\langle i|$ is simply used to pad the subspace to ensure that the matrix is unitary. It will not be of any importance in the following calculations and can simply be ignored. It is well-known that

$$|S|_{\mathrm{tr}} = \mathsf{max}_{\mathsf{U}}\{|\operatorname{tr}(SU)|\} \, ,$$

where $U$ is any unitary matrix. That is, any specific matrix $U$ is going to give a lower bound on the trace norm. Both $\tilde{M}_s$ and $\tilde{M}_s^T$ are such unitary matrices,

$$\begin{aligned} |S|_{\mathrm{tr}} &= \mathsf{max}_{\mathsf{U}}\{|\operatorname{tr}(SU)|\} \geq |\operatorname{tr}(S\tilde{M}_s^T)| \\ &= |\operatorname{tr}((M_s - M_{s'})\tilde{M}_s^T)| \\ &= |\sum_{i,j \in \{0,1\}^t} [M_s]_{i,j}[\tilde{M}_s]_{i,j} - \sum_{i,j \in \{0,1\}^t} [M_{s'}]_{i,j}[\tilde{M}_s]_{i,j}| \\ &= |1 - \sum_{i,j \in \{0,1\}^t} [M_{s'}]_{i,j}[\tilde{M}_s]_{i,j}| \, . \end{aligned}$$

Now note that

$$\sum_{i,j\in\{0,1\}^t} [M_{s'}]_{i,j}[\tilde{M}_s]_{i,j} = \sum_{i,j\in\{0,1\}^t, r,r'\in\mathcal{R}} p_r \times \delta_{v_{A_0}(s,r),i} \times \delta_{v_{A_1}(s,r),j} \times \delta_{v_{A_0}(s',r'),i} \times \delta_{v_{A_1}(s',r'),j} \ .$$

However, the pair $(v_{A_0}(s,r), v_{A_1}(s,r))$ uniquely defines $s$ and hence the sum is 0 unless $s = s'$. Therefore for $s \neq s' : |S|_{\mathrm{tr}} \geq 1$. Putting it together we get,

$$p_{guess} = \frac{1}{2} + \frac{1}{4}|\Delta|_{\mathrm{tr}} = \frac{1}{2} + \frac{1}{4}|S|_{\mathrm{tr}} \geq \frac{1}{2} + \frac{1}{4} \times 1 = \frac{3}{4} \ ,$$

which completes the proof. □

*Proof (Theorem 3).* We describe the attack a verifier might execute. We first specify the state we will send to $P$:

$$\frac{1}{\sqrt{2}} \left(|0\rangle + |1\rangle\right) F_n(|1 + r \bmod n\rangle_0)\, F_n(|1\rangle_1) \cdots F_n(|1\rangle_{n-1}) \ ,$$

where $F_n$ is the quantum Fourier transform over $\mathbb{Z}_n$ and $r$ is a random non-zero value in $\mathbb{Z}_n$. The state can also be written as

$$\frac{1}{\sqrt{2}} \left(|0\rangle F_n(|1 + r \bmod n\rangle_0)\, F_n(|1\rangle_1) \cdots F_n(|1\rangle_{n-1}) + |1\rangle F_n(|1 + r \bmod n\rangle_0)\, F_n(|1\rangle_1) \cdots F_n(|1\rangle_{n-1})\right) .$$

To see what happens to this state under the operation done by $P$, consider the $k$'th register of those where $F_n$ is applied, in the summand corresponding to challenge bit $b$. Before $P$'s operation, it is in state

$$F_n(|i_k\rangle_k) = \sum_{j=0}^{n-1} \omega_n^{i_k j} |j\rangle_k \ ,$$

where $\omega_n$ is the principal $n$'th root of unity and $i_0 = 1 + r \bmod n$ and $i_k = 1$ otherwise. After the operation, we have the new state

$$\sum_{j=0}^{n-1} \omega_n^{i_k j} |j + \rho_b(k)\rangle_k = \sum_{j'=0}^{n-1} \omega_n^{i_k (j' - \rho_b(k))} \left|j'\right\rangle_k = \omega_n^{-i_k \rho_b(k)} \sum_{j'=0}^{n-1} \omega_n^{i_k j'} \left|j'\right\rangle_k = \omega_n^{-i_k \rho_b(k)} F_n(|i_k\rangle_k) \ ,$$

by a simple substitution of variables. Plugging this into the above overall state, we get that the state $P$ returns can be written as

$$\frac{1}{\sqrt{2}} \cdot \big(|0\rangle\, \omega_n^{-(1+r)\rho_0(0)} F_n(|1 + r \bmod n\rangle) \omega_n^{-\rho_0(1)} F_n(|1\rangle) \cdots \omega_n^{-\rho_0(n-1)} F_n(|1\rangle)$$
$$+ |1\rangle\, \omega_n^{-(1+r)\rho_1(0)} F_n(|1 + r \bmod n\rangle) \omega_n^{-\rho_1(1)} F_n(|1\rangle) \cdots \omega_n^{-\rho_1(n-1)} F_n(|1\rangle)\big) .$$

Collecting some terms, we get

$$\frac{1}{\sqrt{2}} \cdot \big(|0\rangle\, \omega_n^{-r\rho_0(0) - \sum_k \rho_0(k)} F_n(|1 + r \bmod n\rangle) F_n(|1\rangle) \cdots F_n(|1\rangle)$$
$$+ |1\rangle\, \omega_n^{-r\rho_1(0) - \sum_k \rho_1(k)} F_n(|1 + r \bmod n\rangle) F_n(|1\rangle) \cdots F_n(|1\rangle)\big) .$$

Note that since $\rho_0, \rho_1$ are permutations, we have $\sum_k \rho_0(k) = \sum_k \rho_1(k) = n(n-1)/2$. Using this and the fact that $\rho_0 = \sigma^{-1}, \rho_1 = \pi\sigma^{-1}$, we get that our state is of form

$$\frac{1}{\sqrt{2}} \omega_n^{-n(n-1)/2 - r\sigma^{-1}(0)} \left(|0\rangle + \omega_n^{r(\sigma^{-1}(0) - \pi(\sigma^{-1}(0)))} |1\rangle\right) F_n(|1 + r \bmod n\rangle) F_n(|1\rangle) \cdots F_n(|1\rangle) \ .$$

In the final step of the attack, we measure the first bit of the state in the basis $(|0\rangle+|1\rangle/\sqrt{2}, (|0\rangle-|1\rangle/\sqrt{2}$. Call the measurement results 0 respectively 1.

Let us consider the distribution we can expect of the measurement result: $\sigma^{-1}(0)$ is uniform in $\mathbb{Z}_n$. Furthermore, it is clear that $r(\sigma^{-1}(0) - \pi(\sigma^{-1}(0)))$ is 0 if $\sigma^{-1}(0)$ is a fixed point of $\pi$ and uniform in $\mathbb{Z}_n^*$ otherwise. It follows that the distribution of the final measurement result depends only on the number of fixed points of $P$'s secret $\pi$.

Furthermore, if $\sigma^{-1}(0)$ is a fixed point of $\pi$, we will measure 0 with probability 1, if not, we get result 1 with non-negligible probability. It follows by a Chernoff bound that in a polynomial number of queries we can decide except with negligible error probability whether $\pi$ has a fixed point or not. In fact, it is not hard to see that the distribution of the measurement result for different numbers of fixed points have non-negligible statistical distance, so in a polynomial number of queries one can even get a reliable estimate of the number of fixed points of $\pi$.

Now, if the protocol was zero-knowledge in our model, a simulator would exist that on input any two isomorphic graphs would create a state indistinguishable from the state our attack creates with the help of the prover. Thus, running the simulator a sufficient number of times followed by the measurements described above, we get an oracle that tells us whether the permutation that takes one graph to the other has a fixed point.

Using such an oracle, we can compute the permutation: let $G' = \theta(G_1)$ for a random permutation $\theta$ and ask the oracle if the permutation mapping $G_0$ to $G'$ has a fixed point. If the answer is no, we know that, e.g., $\theta(\pi(0)) \neq 0$ or equivalently $\pi(0) \neq \theta^{-1}(0)$. Since we know $\theta$, we can exclude one possibility for $\pi(0)$. Repeating this, we eventually find $\pi(0)$ and can compute other values of $\pi$ in the same way. This terminates in polynomial time since a random permutation has no fixed points with constant probability (about $1/e$), and if this happens, the value we can exclude is uniform among the potential values.

*Proof (of Theorem 2).* We first consider the global state after the attack, assuming basis state $|s\rangle$ has been shared, which using the above notation is

$$\frac{1}{\sqrt{M}} \sum_{f \in P_s, A \in F} \alpha_A |f(A)\rangle |A\rangle |\Psi_{f(A)}\rangle ,$$

where $P$ is the set of all players, and $|\Psi_{f(S)}\rangle$ is the state computed by the adversary from $|f(A)\rangle$. To get the state actually held by the adversary, we trace out the part held by the players and using that $\langle f(P)|f'(P)\rangle = 0$ for $f \neq f'$, we get the mixed state $\rho_s$:

$$\rho_s = \frac{1}{M} \sum_{f \in P_s, A, A' \in F} \alpha_A \alpha^*_{A'} |A\rangle\langle A'| |\Psi_{f(A)}\rangle\langle \Psi_{f(A')}| = \frac{1}{M} \sum_{A, A' \in F} \alpha_A \alpha^*_{A'} |A\rangle\langle A'| \sum_{f \in P_s} |\Psi_{f(A)}\rangle\langle \Psi_{f(A')}| .$$

For the if-part of the theorem, assume that $A \cup A' \in G$. Then, when $f$ varies over all polynomials, the non-$\perp$ entries in $|f(A)\rangle, |f(A')\rangle$ assume all possible values the same number of times. Since (for any fixed value of the ancilla) $|\Psi_{f(A)}\rangle\langle\Psi_{f(A')}|$ depends only on these values, we find that $\rho_s = \sum_{f \in P_s} |\Psi_{f(A)}\rangle\langle\Psi_{f(A')}|$ and hence $\rho_s$ does not depend on $s$.

For the only-if part, we consider that the adversary could simply "copy" the shares, that is, define $V$ such that $|\Psi_{f(A)}\rangle = |f(A)\rangle$, and set the query to be $(|A\rangle + |A'\rangle)/\sqrt{2}$ for sets $A, A'$ where $A \cup A'$ has size larger than $t$. Now, $s$ is uniquely determined from $|f(A)\rangle, |f(A')\rangle$ and hence the matrices $\sum_{f \in P_s} |\Psi_{f(A)}\rangle\langle\Psi_{f(A')}|$ are different from different values of $s$, so we can distinguish (with some probability) between sharings of different basis states. □

## B Multiparty Computation

In this section we consider the models for MPC protocols. A classical passive attack on a multiparty computation protocol looks a lot like the attacks on secret sharing: you query for a subset and get back the party's entire view of the protocol. Of course, you can generalize this to a superposition attacks in the same way. And you can ask if there is some adversary structure for which the protocol would be secure against such an attack, assuming classical security.

Security for MPC protocols is usually defined as an adversary's inability to distinguish between an attack in the *real world* where he's allowed access to a corruption oracle of some subset of the parties and an *ideal world* where the attack is *simulated* towards the adversary using the *ideal functionality* which models the intended behavior of the protocol. We hence need to describe both models for the real and for the ideal world. They will need different spaces and different operations to execute. As before, we have $n$ parties running the protocol. We name the parties $P_1, \ldots, P_n$. Let $[n] = \{1, \ldots, n\}$. An adversary structure is $F \subset 2^{[n]}$. Each party, $P_i$, has local input, $s_i \in \mathbb{S}_i$ which is supplied by the adversary and chooses randomness, $r_i \in \mathcal{R}_i$. We use $s \in \mathbb{S}$ and $r \in \mathcal{R}$ to denote the concatenation of each of the $s_i$ respectively the $r_i$. Note that when $s$ and $r$ are fixed, the execution of a protocol is deterministic, i.e., it is fixed what all parties send and receive. We use $v_i(s,r)$ and $o_i(s,r)$ to denote the private view respectively the output or party $P_i$, when the protocol has been run on inputs $s$ and using randomness $r$. Note these are functions and not general quantum operations as the parties, even the corrupted ones, are expected to run the protocol honestly. Concretely, the view $v_i(s,r)$ if $P_i$ is $v_i(s,r) = (s_i, r_i, msg_i(s,r))$, where $msg_i(s,r)$ is the concatenation of the messages that $P_i$ receive from the other parties when the protocol is run on input vector $s$ with randomness vector $r$. For $A \subset [n]$, let $v_A(s,r) = \{v_i(s,r)\}_{i\in A}$, $s_A = \{s_i\}_{i\in A}$ and $o_A(s,r) = \{o_i(s,r)\}_{i\in A}$ be strings containing the concatenation of views, inputs respectively outputs for parties $P_i$ with $i \in A$.

For convenience, in the following we assume that each such string is padded, so that they have the same length ($t$ bits) regardless of the size of $A$.

## B.1 MPC model in the 'Real world'

First we will consider the case of running and attacking the protocol in the real world. As earlier, all actions taken by the parties and the adversary will be considered purified so the overall state remains pure throughout. Consider the following space,

$$\mathcal{H}_{\mathsf{total}} = \mathcal{H}_{\mathsf{parties}} \otimes \mathcal{H}_{\mathsf{in}} \otimes \mathcal{H}_{\mathsf{out}} \otimes \mathcal{H}_{\mathsf{env}} \otimes \mathcal{H}_{\mathsf{query}} \ .$$

- $\mathcal{H}_{\mathsf{parties}}$ contains the private views and purification of the randomness for the parties.
- $\mathcal{H}_{\mathsf{in}}$ contains the input the parties will use to run the protocol.
- $\mathcal{H}_{\mathsf{out}}$ is where the output will be stored after running the protocol.
- $\mathcal{H}_{\mathsf{env}}$ is the environment and is used to store auxiliary input and any auxiliary register needed by the adversary. The dimension is therefore arbitrary, though finite.
- $\mathcal{H}_{\mathsf{query}}$ is where the query to, and response from, the oracle will be stored.

Each of these subspaces are, of course, of appropriate (and finite) dimension.

We will break the superposition run, and attack, of a MPC protocol in the real world down into four unitaries. After each unitary we will consider the change to the description of the state. In the beginning all the registers are blank (i.e., have value 0), except the environment, which might contain some (purified) auxiliary input for the adversary. The initial state is hence,

$$|init\rangle_{\mathsf{t}}^{\mathrm{RW}} = \sum_x \alpha_x |0\rangle_{\mathsf{p}} |0\rangle_{\mathsf{i}} |0\rangle_{\mathsf{o}} |x\rangle_{\mathsf{e}} |0\rangle_{\mathsf{q}} \ ,$$

where $|init\rangle_{\mathsf{t}}^{\mathrm{RW}} \in \mathcal{H}_{\mathsf{total}}$, $|0\rangle_{\mathsf{p}} \in \mathcal{H}_{\mathsf{parties}}$, $|0\rangle_{\mathsf{i}} \in \mathcal{H}_{\mathsf{in}}$, $|0\rangle_{\mathsf{o}} \in \mathcal{H}_{\mathsf{out}}$, $|x\rangle_{\mathsf{e}} \in \mathcal{H}_{\mathsf{env}}$ and $|0\rangle_{\mathsf{q}} \in \mathcal{H}_{\mathsf{query}}$, as should be expected from the notation. The superscript, RW, specifies that it's in the real world. The first unitary is applied by the adversary and supplies the inputs to the parties. This is an arbitrary unitary operation. We will denote it, $U_{\mathrm{IN}}^{\mathrm{ADV}}$,

$$U_{\mathrm{IN}}^{\mathrm{ADV}} : \mathcal{H}_{\mathsf{in}} \otimes \mathcal{H}_{\mathsf{env}} \to \mathcal{H}_{\mathsf{in}} \otimes \mathcal{H}_{\mathsf{env}} \ .$$

The result of this is that the input registers are now filled. These are in superposition over all possible inputs. The state after the first unitary is therefore,

$$|1\rangle_{\mathsf{t}}^{\text{RW}} = \sum_{x,s} \alpha_{x,s} |0\rangle_{\mathsf{p}} |s\rangle_{\mathsf{i}} |0\rangle_{\mathsf{o}} |x\rangle_{\mathsf{e}} |0\rangle_{\mathsf{q}} \,,$$

where $s = s_{[n]} = (s_1, \ldots, s_n)$ is the vector of inputs. The classical protocol $\pi$ is now run honestly without intervention from the adversary. This is a classical function $\pi$ run in superposition of the possible inputs and produces a corresponding superposition of private views and outputs for the parties. We will denote this unitary, $U^{\text{PRO}}_{\text{RUN},\pi}$,

$$U^{\text{PRO}}_{\text{RUN},\pi} : \mathcal{H}_{\mathsf{parties}} \otimes \mathcal{H}_{\mathsf{in}} \otimes \mathcal{H}_{\mathsf{out}} \to \mathcal{H}_{\mathsf{parties}} \otimes \mathcal{H}_{\mathsf{in}} \otimes \mathcal{H}_{\mathsf{out}} \,.$$

Recall that we are purifying all actions, hence also the choice of randomness when the protocol is run. The state after the second unitary is therefore,

$$|2\rangle_{\mathsf{t}}^{\text{RW}} = \sum_{x,s,r} \alpha_{x,s} \sqrt{p_r} \big|v_{[n]}(s,r)\big\rangle_{\mathsf{p}} |s\rangle_{\mathsf{i}} \big|o_{[n]}(s,r)\big\rangle_{\mathsf{o}} |x\rangle_{\mathsf{e}} |0\rangle_{\mathsf{q}} \,. \tag{3}$$

Next the adversary needs to construct his query to the oracle. This includes choosing the superposition of subsets he will corrupt and associated values for the response registers for input, output and view. For simplicity we will sometimes refer to the last three values by $a$. That is, $a = (a_{\mathsf{qi}}, a_{\mathsf{qo}}, a_{\mathsf{qv}})$. This is an arbitrary unitary operation. We will denote it, $U^{\text{ADV},F}_{\text{QUERY}}$,

$$U^{\text{ADV},F}_{\text{QUERY}} : \mathcal{H}_{\mathsf{env}} \otimes \mathcal{H}_{\mathsf{query}} \to \mathcal{H}_{\mathsf{env}} \otimes \mathcal{H}_{\mathsf{query}} \,,$$

where the $F$ in the notation specifies that all subset with non-zero amplitude in the constructed query are from the adversary structure $F$. The state is now,

$$|3\rangle_{\mathsf{t}}^{\text{RW}} = \sum_{x,s,r,A,a} \alpha_{x,s,A,a} \sqrt{p_r} \big|v_{[n]}(s,r)\big\rangle_{\mathsf{p}} |s\rangle_{\mathsf{i}} \big|o_{[n]}(s,r)\big\rangle_{\mathsf{o}} |x\rangle_{\mathsf{e}} |A, a\rangle_{\mathsf{q}} \,.$$

The next unitary is applied by the oracle, that, for each corrupted subset in the query, fills the input, output and view into the response register supplied by the adversary. This is a classical function on each corrupted subset and view in the superposition which fills the input, output and view into the response register. We will denote this unitary, $U^{\text{RW}}_{\text{RES}}$,

$$U^{\text{IW}}_{\text{RES}} : \mathcal{H}_{\mathsf{parties}} \otimes \mathcal{H}_{\mathsf{in}} \otimes \mathcal{H}_{\mathsf{out}} \otimes \mathcal{H}_{\mathsf{query}} \to \mathcal{H}_{\mathsf{parties}} \otimes \mathcal{H}_{\mathsf{in}} \otimes \mathcal{H}_{\mathsf{out}} \otimes \mathcal{H}_{\mathsf{query}} \,.$$

The state after the fourth unitary is

$$|4\rangle_{\mathsf{t}}^{\text{RW}} = \sum_{x,s,r,A,a} \alpha_{x,s,A,a} \sqrt{p_r} \big|v_{[n]}(s,r)\big\rangle_{\mathsf{p}} |s\rangle_{\mathsf{i}} \big|o_{[n]}(s,r)\big\rangle_{\mathsf{o}} |x\rangle_{\mathsf{e}} \big|A, a_{\mathsf{qi}} \oplus s_A, a_{\mathsf{qo}} \oplus o_A(s,r), a_{\mathsf{qv}} \oplus v_A(s,r)\big\rangle_{\mathsf{q}} \,,$$

where $o(s,r) = (o_1, \ldots, o_n)$ where $o_i$ is the output of party number $i$ in the protocol $\pi$ run on inputs $s$ and randomness $r$. The adversary receives the response register and must now guess if he's in the real or ideal world. He can do this using the input register, his auxiliary register and the query register. To see the adversary's final state we need to trace out the register holding the view of all the parties,

$$\begin{aligned}
\rho^{\text{RW}}_{\text{ADV}} = \operatorname{tr}_{\mathsf{p}}(|4\rangle\langle 4|_{\mathsf{t}}^{\text{RW}}) &= \sum_{r,r',s,s'} \sqrt{p_r}\sqrt{p_{r'}} \big|\psi^{\text{ADV}}_{r,s}\big\rangle\big\langle\psi^{\text{ADV}}_{r',s'}\big| \operatorname{tr}_{\mathsf{p}}\big(\big|v_{[n]}(s,r)\big\rangle_{\mathsf{p}}\big\langle v_{[n]}(s',r')\big|_{\mathsf{p}}\big) \\
&= \sum_{r,r',s,s'} \sqrt{p_r}\sqrt{p_{r'}} \big|\psi^{\text{ADV}}_{r,s}\big\rangle\big\langle\psi^{\text{ADV}}_{r',s'}\big| \big\langle v_{[n]}(s,r)\big|_{\mathsf{p}} \big|v_{[n]}(s',r')\big\rangle_{\mathsf{p}}\big) \\
&= \sum_{r,s} p_r \big|\psi^{\text{ADV}}_{r,s}\big\rangle\big\langle\psi^{\text{ADV}}_{r,s}\big| \,,
\end{aligned}$$

where $|\psi^{\text{ADV}}_{r,s}\rangle = \sum_{x,A,a} \alpha_{x,s,A,a} |s\rangle_{\mathsf{i}} |o_{[n]}(s,r)\rangle_{\mathsf{o}} |x\rangle_{\mathsf{e}} |A, a_{\mathsf{qi}} \oplus s_A, a_{\mathsf{qo}} \oplus o_A(s,r), a_{\mathsf{qv}} \oplus v_A(s,r)\rangle_{\mathsf{q}}$.

It is interesting to note that even though the input register was supplied by the adversary he only sees a mixed state. This happens as the parties keep a private copy of their input and therefore collapses the input register from the point of view of the adversary. While it is standard to assume the players keep such a copy (i.e. they remember their own input), you could consider a model where they do not. In this case the input register would *not* collapse and hence the adversary would have a superposition, instead of a mixed state, over the choice of input. It is an open problem how this would effect security.

We will sum up these steps below in Figure 1.

Inital state:

$$|init\rangle^{\text{RW}}_{\mathsf{t}} = \sum_{x} \alpha_x |0\rangle_{\mathsf{p}} |0\rangle_{\mathsf{i}} |0\rangle_{\mathsf{o}} |x\rangle_{\mathsf{e}} |0\rangle_{\mathsf{q}}$$

1.

$$U^{\text{ADV}}_{\text{IN}} : \mathcal{H}_{\mathsf{in}} \otimes \mathcal{H}_{\mathsf{env}} \to \mathcal{H}_{\mathsf{in}} \otimes \mathcal{H}_{\mathsf{env}}$$
$$|1\rangle^{\text{RW}}_{\mathsf{t}} = \sum_{x,s} \alpha_{x,s} |0\rangle_{\mathsf{p}} |s\rangle_{\mathsf{i}} |0\rangle_{\mathsf{o}} |x\rangle_{\mathsf{e}} |0\rangle_{\mathsf{q}}$$

2.

$$U^{\text{PRO}}_{\text{RUN}} : \mathcal{H}_{\mathsf{parties}} \otimes \mathcal{H}_{\mathsf{in}} \otimes \mathcal{H}_{\mathsf{out}} \to \mathcal{H}_{\mathsf{parties}} \otimes \mathcal{H}_{\mathsf{in}} \otimes \mathcal{H}_{\mathsf{out}}$$
$$|2\rangle^{\text{RW}}_{\mathsf{t}} = \sum_{x,s,r} \alpha_{x,s} \sqrt{p_r} |v_{[n]}(s,r)\rangle_{\mathsf{p}} |s\rangle_{\mathsf{i}} |o_{[n]}(s,r)\rangle_{\mathsf{o}} |x\rangle_{\mathsf{e}} |0\rangle_{\mathsf{q}}$$

3.

$$U^{\text{ADV},F}_{\text{QUERY}} : \mathcal{H}_{\mathsf{env}} \otimes \mathcal{H}_{\mathsf{query}} \to \mathcal{H}_{\mathsf{env}} \otimes \mathcal{H}_{\mathsf{query}}$$
$$|3\rangle^{\text{RW}}_{\mathsf{t}} = \sum_{x,s,r,A,a} \alpha_{x,s,A,a} \sqrt{p_r} |v_{[n]}(s,r)\rangle_{\mathsf{p}} |s\rangle_{\mathsf{i}} |o_{[n]}(s,r)\rangle_{\mathsf{o}} |x\rangle_{\mathsf{e}} |A,a\rangle_{\mathsf{q}}$$

4.

$$U^{\text{RW}}_{\text{RES}} : \mathcal{H}_{\mathsf{parties}} \otimes \mathcal{H}_{\mathsf{in}} \otimes \mathcal{H}_{\mathsf{out}} \otimes \mathcal{H}_{\mathsf{query}} \to \mathcal{H}_{\mathsf{parties}} \otimes \mathcal{H}_{\mathsf{in}} \otimes \mathcal{H}_{\mathsf{out}} \otimes \mathcal{H}_{\mathsf{query}}$$
$$|4\rangle^{\text{RW}}_{\mathsf{t}} = \sum_{x,s,r,A,a} \alpha_{x,s,A,a} \sqrt{p_r} |v_{[n]}(s,r)\rangle_{\mathsf{p}} |s\rangle_{\mathsf{i}} |o_{[n]}(s,r)\rangle_{\mathsf{o}} |x\rangle_{\mathsf{e}} |A, a_{\mathsf{qi}} \oplus s_A, a_{\mathsf{qo}} \oplus o_A(s,r), a_{\mathsf{qv}} \oplus v_A(s,r)\rangle_{\mathsf{q}}$$

**Fig. 1.** Purified run of multiparty computation in the real world with superposition attacks

### B.2 MPC model in the 'Ideal world'

Secondly we consider the ideal world, where a simulator is to simulate a real attack for the adversary using only the ideal functionality. The space for this model is,

$$\mathcal{H}_{\mathsf{total}} = \mathcal{H}_{\mathsf{ideal}} \otimes \mathcal{H}_{\mathsf{in}} \otimes \mathcal{H}_{\mathsf{out}} \otimes \mathcal{H}_{\mathsf{sim}} \otimes \mathcal{H}_{\mathsf{env}} \otimes \mathcal{H}_{\mathsf{query}} \,.$$

– $\mathcal{H}_{\mathsf{in}}, \mathcal{H}_{\mathsf{out}}$ and $\mathcal{H}_{\mathsf{env}}$ serve the same purpose as in the real world.
– $\mathcal{H}_{\mathsf{sim}}$ denotes the subspace in which the simulator operates.
– $\mathcal{H}_{\mathsf{query}}$ is still the register where the adversary constructs his query, but the simulator will be allowed to change this before using it to query the ideal functionality.

– Finally, we have a space for the ideal functionality, $\mathcal{H}_{\mathsf{ideal}}$. This is necessary as we want to purify the random choices made and we need the state of the ideal functionality to be entangled with the input register as the parties would be in the real world.

We will describe the unitaries that differ and refer to the earlier description for those that don't. There will be six unitaries in total. The initial state is,

$$|init\rangle_{\mathsf{t}}^{\mathrm{IW}} = \sum_{x} \alpha_x |0\rangle_{\mathsf{if}} |0\rangle_{\mathsf{i}} |0\rangle_{\mathsf{o}} |0\rangle_{\mathsf{s}} |x\rangle_{\mathsf{e}} |0\rangle_{\mathsf{q}} ,$$

where $|init\rangle_{\mathsf{t}} \in \mathcal{H}_{\mathsf{total}}$, $|0\rangle_{\mathsf{if}} \in \mathcal{H}_{\mathsf{ideal}}$, $|0\rangle_{\mathsf{i}} \in \mathcal{H}_{\mathsf{in}}$, $|0\rangle_{\mathsf{o}} \in \mathcal{H}_{\mathsf{out}}$, $|0\rangle_{\mathsf{s}} \in \mathcal{H}_{\mathsf{sim}}$, $|0\rangle_{\mathsf{e}} \in \mathcal{H}_{\mathsf{env}}$ and $|0\rangle_q \in \mathcal{H}_{\mathsf{query}}$, as should be expected from the notation. The superscript, IW, denotes that it's in the ideal world. The first unitary is exactly as in the real world and hence,

$$U_{\mathrm{IN}}^{\mathrm{ADV}} : \mathcal{H}_{\mathsf{in}} \otimes \mathcal{H}_{\mathsf{env}} \to \mathcal{H}_{\mathsf{in}} \otimes \mathcal{H}_{\mathsf{env}}$$

$$|1\rangle_{\mathsf{t}}^{\mathrm{IW}} = \sum_{x,s} \alpha_{x,s} |0\rangle_{\mathsf{if}} |s\rangle_{\mathsf{i}} |0\rangle_{\mathsf{o}} |0\rangle_{\mathsf{s}} |x\rangle_{\mathsf{e}} |0\rangle_{\mathsf{q}} .$$

The ideal functionality now entangles itself with the input register and produces the correct outputs in the output register. The *correct* outputs are defined via an $n$-ary function $f$. For inputs $(s_1, \ldots, s_n)$ the correct outputs are $(o_1, \ldots, o_n) = f(s_1, \ldots, s_n)$. We allow $f$ to be a randomized function, so that $(o_1, \ldots, o_n) = f(s_1, \ldots, s_n; r)$ for some randomizer $r$. The ideal functionality is the unitary which evaluates the classical function $f$ in superposition using a purified $r$. We will denote this unitary, $U_{\mathrm{EVAL},f}^{\mathrm{IW}}$,

$$U_{\mathrm{EVAL},f}^{\mathrm{IW}} : \mathcal{H}_{\mathsf{ideal}} \otimes \mathcal{H}_{\mathsf{in}} \otimes \mathcal{H}_{\mathsf{out}} \to \mathcal{H}_{\mathsf{ideal}} \otimes \mathcal{H}_{\mathsf{in}} \otimes \mathcal{H}_{\mathsf{out}} .$$

Recall that any random choices are purified onto $\mathcal{H}_{\mathsf{ideal}}$. The resulting state is therefore,

$$|2\rangle_{\mathsf{t}}^{\mathrm{IW}} = \sum_{x,s,r} \alpha_{x,s} \sqrt{p_r} |s, r\rangle_{\mathsf{if}} |s\rangle_{\mathsf{i}} \left|o_{[n]}(s, r)\right\rangle_{\mathsf{o}} |0\rangle_{\mathsf{s}} |x\rangle_{\mathsf{e}} |0\rangle_{\mathsf{q}} .$$

Inspect the $\mathsf{out}$-register to note that we use the notation $o_{[n]}(s, r)$ for the output vector $(o_1, \ldots, o_n) = f(s_1, \ldots, s_n; r)$. This could potentially lead to confusion as we also use $o_{[n]}(s, r)$ to denote the output of the *protocol*, in the real world. We will, however, restrict our attention to perfectly correct protocols, meaning that running the protocol $(o_1, \ldots, o_n) = \pi(x_1, \ldots, x_n; r_1, \ldots, r_n)$, i.e., with uniformly random randomness $r_i$ for party $P_i$, will give a distribution which is identical to evaluating $f(x_1, \ldots, x_n; r)$ for a uniformly random randomizer $r$ for the randomized function $f$. This means that we in the analysis, with loss of generality, can assume that $r = (r_1, \ldots, r_n)$ and that $f(x_1, \ldots, x_n; r) = \pi(x_1, \ldots, x_n; r_1, \ldots, r_n)$. Hence $o_{[n]}(s, r)$ is exactly the same value in the ideal world and in the real world.

The adversary constructs his query exactly as earlier,

$$U_{\mathrm{QUERY}}^{\mathrm{ADV},F} : \mathcal{H}_{\mathsf{env}} \otimes \mathcal{H}_{\mathsf{query}} \to \mathcal{H}_{\mathsf{env}} \otimes \mathcal{H}_{\mathsf{query}}$$

$$|3\rangle_{\mathsf{t}}^{\mathrm{IW}} = \sum_{x,s,r,A,a} \alpha_{x,s,A,a} \sqrt{p_r} |s, r\rangle_{\mathsf{if}} |s\rangle_{\mathsf{i}} \left|o_{[n]}(s, r)\right\rangle_{\mathsf{o}} |0\rangle_{\mathsf{s}} |x\rangle_{\mathsf{e}} |A, a\rangle_{\mathsf{q}} .$$

Now the simulator gets the query and must construct an appropriate response to the adversary, only using the ideal functionality. First the simulator is allowed to change the adversary's query using an auxiliary register. Only requirement is that it is corruption preserving [8]. This is an arbitrary unitary operation. We will denote it, $U_{\mathrm{QUERY}}^{\mathrm{SIM}}$,

$$U_{\mathrm{QUERY}}^{\mathrm{SIM}} : \mathcal{H}_{\mathsf{sim}} \otimes \mathcal{H}_{\mathsf{query}} \to \mathcal{H}_{\mathsf{sim}} \otimes \mathcal{H}_{\mathsf{query}}$$

$$|4\rangle_{\mathsf{t}}^{\mathrm{IW}} = \sum_{x,z,s,r,A,a} \alpha_{x,z,s,A,a} \sqrt{p_r} |s, r\rangle_{\mathsf{if}} |s\rangle_{\mathsf{i}} \left|o_{[n]}(s, r)\right\rangle_{\mathsf{o}} |z\rangle_{\mathsf{s}} |x\rangle_{\mathsf{e}} |A, a\rangle_{\mathsf{q}}$$

[8] That is, the amplitude of each corruption subset $A$ is unchanged

Now the corruption oracle is run on the query from the simulator. This is a classical function that each corrupted subset and each view in the superposition which fills in the input and output into the response register. We will denote this unitary, $U^{\mathrm{IW}}_{\mathrm{RES}}$,

$$U^{\mathrm{IW}}_{\mathrm{RES}} : \mathcal{H}_{\mathsf{in}} \otimes \mathcal{H}_{\mathsf{out}} \otimes \mathcal{H}_{\mathsf{query}} \to \mathcal{H}_{\mathsf{in}} \otimes \mathcal{H}_{\mathsf{out}} \otimes \mathcal{H}_{\mathsf{query}}$$

and the response register now contains input and output from the corrupted parties, but, of course, not their views.

$$|5\rangle^{\mathrm{IW}}_{\mathsf{t}} = \sum_{x,z,s,r,A,a} \alpha_{x,z,s,A,a}\sqrt{p_r}|s,r\rangle_{\mathsf{if}}|s\rangle_{\mathsf{i}}\big|o_{[n]}(s,r)\big\rangle_{\mathsf{o}}|z\rangle_{\mathsf{s}}|x\rangle_{\mathsf{e}}\big|A, a_{\mathsf{qi}} \oplus s_A, a_{\mathsf{qo}} \oplus o_A(s,r), a_{\mathsf{qv}}\big\rangle_{\mathsf{q}} .$$

Next the simulator must try to simulate the response the adversary got in the real world. We will denote this unitary, $U^{\mathrm{SIM}}_{\mathrm{RES}}$,

$$U^{\mathrm{SIM}}_{\mathrm{RES}} : \mathcal{H}_{\mathsf{sim}} \otimes \mathcal{H}_{\mathsf{query}} \to \mathcal{H}_{\mathsf{sim}} \otimes \mathcal{H}_{\mathsf{query}} .$$

The resulting state is

$$|6\rangle^{\mathrm{IW}}_{\mathsf{t}} = \sum_{x,z,s,r,A,a} \alpha'_{x,z,s,A,a}\sqrt{p_r}|s,r\rangle_{\mathsf{if}}|s\rangle_{\mathsf{i}}\big|o_{[n]}(s,r)\big\rangle_{\mathsf{o}}|z\rangle_{\mathsf{s}}|x\rangle_{\mathsf{e}}|A, \psi_{z,s,A,a}\rangle_{\mathsf{q}} .$$

Finally, to see what state the adversary sees we must trace out the ideal functionality and the simulator registers,

$$\begin{aligned}
\rho^{\mathrm{IW}}_{\mathrm{ADV}} &= \mathrm{tr}_{\mathsf{if,sim}}(|6\rangle\langle 6|^{\mathrm{IW}}_{\mathsf{t}}) \\
&= \sum_{r,r',s,s',z,z'} p_r\big|\psi^{\mathrm{ADV}}_{r,s,z}\big\rangle\big\langle\psi^{\mathrm{ADV}}_{r',s',z'}\big| \,\mathrm{tr}_{\mathsf{if,sim}}\big((|s,r\rangle_{\mathsf{if}}\langle s',r'|_{\mathsf{if}} \otimes |z\rangle_{\mathsf{s}}\langle z'|_{\mathsf{s}}\big) \\
&= \sum_{r,r',s,s',z,z'} p_r\big|\psi^{\mathrm{ADV}}_{r,s,z}\big\rangle\big\langle\psi^{\mathrm{ADV}}_{r',s',z'}\big| \,\big(\langle s,r|_{\mathsf{if}}|s',r'\rangle_{\mathsf{if}} \times \langle z|_{\mathsf{s}}|z'\rangle_{\mathsf{s}}\big) \\
&= \sum_{r,s,z} p_r\big|\psi^{\mathrm{ADV}}_{r,s,z}\big\rangle\big\langle\psi^{\mathrm{ADV}}_{r,s,z}\big| ,
\end{aligned}$$

where $|\psi^{\mathrm{ADV}}_{r,s,z}\rangle = \sum_{x,A,a} \alpha_{x,z,s,A,a}|s\rangle_{\mathsf{i}}\big|o_{[n]}(s,r)\big\rangle_{\mathsf{o}}|x\rangle_{\mathsf{e}}|A,\psi_{z,s,A,a}\rangle_{\mathsf{q}}$. We sum up the steps in Figure 2.

A MPC protocol is defined by the operator $U^{\mathrm{PRO}}_{\mathrm{RUN}}$. A specific adversay is defined by the initial state $\sum_x \alpha_a|x\rangle$ and the two unitary operators, $U^{\mathrm{ADV}}_{\mathrm{IN}}, U^{adv,F}_{\mathrm{QUERY}}$. Finally, a simulator is defined by the two unitary operators, $\{U^{\mathrm{SIM}}_{\mathrm{QUERY}}, U^{\mathrm{SIM}}_{\mathrm{RES}}\}$. This allows for the following definition.

**Definition 2.** *We say that the MPC protocol $\pi$ is a perfect passive-secure implementation of the $n$-ary function $f$ against $F$-superposition attacks if there exists a perfect black-box simulator of $\pi$ given $f$ against superposition attacks from $F$. A pair of unitary operators, $(U^{\mathrm{SIM}}_{query}, U^{\mathrm{SIM}}_{res})$, is a perfect black-box simulator of $\pi$ given $f$ against superposition attacks from $F$ if, and only if, for all auxiliary inputs, $\sum_x \alpha_x|x\rangle$, and all pairs of operators $(U^{\mathrm{ADV}}_{\mathrm{IN}}, U^{\mathrm{ADV},F}_{\mathrm{QUERY}})$,*

$$\rho^{\mathrm{RW}}_{\mathrm{ADV}} = \rho^{\mathrm{IW}}_{\mathrm{ADV}} ,$$

*where*

$$\begin{aligned}
\rho^{\mathrm{RW}}_{\mathrm{ADV}} &= \mathrm{tr}_{\mathit{p}}(|4\rangle\langle 4|^{\mathrm{RW}}_{t}) \\
\rho^{\mathrm{IW}}_{\mathrm{ADV}} &= \mathrm{tr}_{\mathit{if,s}}(|6\rangle\langle 6|^{\mathrm{IW}}_{t}) \\
|4\rangle^{\mathrm{RW}}_{t} &= U^{\mathrm{RW}}_{\mathrm{RES}} U^{\mathrm{ADV},F}_{\mathrm{QUERY}} U^{\mathrm{PRO}}_{\mathrm{RUN},\pi} U^{\mathrm{ADV}}_{\mathrm{IN}} |init\rangle^{\mathrm{RW}}_{t} \\
|6\rangle^{\mathrm{IW}}_{t} &= U^{\mathrm{SIM}}_{\mathrm{RES}} U^{\mathrm{IW}}_{\mathrm{RES}} U^{\mathrm{SIM}}_{\mathrm{QUERY}} U^{\mathrm{ADV},F}_{\mathrm{QUERY}} U^{\mathrm{IW}}_{\mathrm{EVAL},f} U^{\mathrm{ADV}}_{\mathrm{IN}} |init\rangle^{\mathrm{IW}}_{t} ,
\end{aligned}$$

*where we for readability assume that the operators are padded with appropriate identities on the subspaces they don't operate.*

Inital state:

$$|init\rangle_{\mathsf{t}}^{\text{IW}} = \sum_{x} \alpha_x |0\rangle_{\mathsf{if}} |0\rangle_{\mathsf{i}} |0\rangle_{\mathsf{o}} |0\rangle_{\mathsf{s}} |x\rangle_{\mathsf{e}} |0\rangle_{\mathsf{q}}$$

1.

$$U_{\text{IN}}^{\text{ADV}} : \mathcal{H}_{\mathsf{in}} \otimes \mathcal{H}_{\mathsf{env}} \to \mathcal{H}_{\mathsf{in}} \otimes \mathcal{H}_{\mathsf{env}}$$

$$|1\rangle_{\mathsf{t}}^{\text{IW}} = \sum_{x,s} \alpha_{x,s} |0\rangle_{\mathsf{if}} |s\rangle_{\mathsf{i}} |0\rangle_{\mathsf{o}} |0\rangle_{\mathsf{s}} |x\rangle_{\mathsf{e}} |0\rangle_{\mathsf{q}}$$

2.

$$U_{\text{EVAL},f}^{\text{IW}} : \mathcal{H}_{\mathsf{ideal}} \otimes \mathcal{H}_{\mathsf{in}} \otimes \mathcal{H}_{\mathsf{out}} \to \mathcal{H}_{\mathsf{ideal}} \otimes \mathcal{H}_{\mathsf{in}} \otimes \mathcal{H}_{\mathsf{out}}$$

$$|2\rangle_{\mathsf{t}}^{\text{IW}} = \sum_{x,s,r} \alpha_{x,s} \sqrt{p_r} |s,r\rangle_{\mathsf{if}} |s\rangle_{\mathsf{i}} \big|o_{[n]}(s,r)\big\rangle_{\mathsf{o}} |0\rangle_{\mathsf{s}} |x\rangle_{\mathsf{e}} |0\rangle_{\mathsf{q}}$$

3.

$$U_{\text{QUERY}}^{adv,F} : \mathcal{H}_{\mathsf{env}} \otimes \mathcal{H}_{\mathsf{query}} \to \mathcal{H}_{\mathsf{env}} \otimes \mathcal{H}_{\mathsf{query}}$$

$$|3\rangle_{\mathsf{t}}^{\text{IW}} = \sum_{x,s,r,A,a} \alpha_{x,s,A,a} \sqrt{p_r} |s,r\rangle_{\mathsf{if}} |s\rangle_{\mathsf{i}} \big|o_{[n]}(s,r)\big\rangle_{\mathsf{o}} |0\rangle_{\mathsf{s}} |x\rangle_{\mathsf{e}} |A,a\rangle_{\mathsf{q}}$$

4.

$$U_{\text{QUERY}}^{\text{SIM}} : \mathcal{H}_{\mathsf{sim}} \otimes \mathcal{H}_{\mathsf{query}} \to \mathcal{H}_{\mathsf{sim}} \otimes \mathcal{H}_{\mathsf{query}}$$

$$|4\rangle_{\mathsf{t}}^{\text{IW}} = \sum_{x,z,s,r,A,a} \alpha_{x,z,s,A,a} \sqrt{p_r} |s,r\rangle_{\mathsf{if}} |s\rangle_{\mathsf{i}} \big|o_{[n]}(s,r)\big\rangle_{\mathsf{o}} |z\rangle_{\mathsf{s}} |x\rangle_{\mathsf{e}} |A,a\rangle_{\mathsf{q}}$$

5.

$$U_{\text{RES}}^{\text{IW}} : \mathcal{H}_{\mathsf{in}} \otimes \mathcal{H}_{\mathsf{out}} \otimes \mathcal{H}_{\mathsf{query}} \to \mathcal{H}_{\mathsf{in}} \otimes \mathcal{H}_{\mathsf{out}} \otimes \mathcal{H}_{\mathsf{query}}$$

$$|5\rangle_{\mathsf{t}}^{\text{IW}} = \sum_{x,z,s,r,A,a} \alpha_{x,z,s,A,a} \sqrt{p_r} |s,r\rangle_{\mathsf{if}} |s\rangle_{\mathsf{i}} \big|o_{[n]}(s,r)\big\rangle_{\mathsf{o}} |z\rangle_{\mathsf{s}} |x\rangle_{\mathsf{e}} |A, a_{\mathsf{qi}} \oplus s_A, a_{\mathsf{qo}} \oplus o_A(s,r), a_{\mathsf{qv}}\rangle_{\mathsf{q}}$$

6.

$$U_{\text{RES}}^{\text{SIM}} : \mathcal{H}_{\mathsf{sim}} \otimes \mathcal{H}_{\mathsf{query}} \to \mathcal{H}_{\mathsf{sim}} \otimes \mathcal{H}_{\mathsf{query}}$$

$$|6\rangle_{\mathsf{t}}^{\text{IW}} = \sum_{x,z,s,r,A,a} \alpha'_{x,z,s,A,a} \sqrt{p_r} |s,r\rangle_{\mathsf{if}} |s\rangle_{\mathsf{i}} \big|o_{[n]}(s,r)\big\rangle_{\mathsf{o}} |z\rangle_{\mathsf{s}} |x\rangle_{\mathsf{e}} |A, \psi_{z,s,A,a}\rangle_{\mathsf{q}}$$

**Fig. 2.** Purified run of multiparty computation in the ideal world with superposition attacks

## B.3 No simulator for general MPC if $U^{\mathrm{SIM}}_{\mathrm{QUERY}}$ is unitary

In this section we argue that in the model with supplied response registers, it seems that simulation is impossible for purely technical reasons.

For this purpose, consider the dummy 4-party function $d(x_1, x_2, x_3, x_4) = (\lambda, \lambda, \lambda, \lambda)$, where $\lambda$ is the empty string, i.e., the function which gives no outputs on any of its inputs. Consider protocol $\delta$, where parties $P_2, P_3, P_4$ runs as follows: On input $x_i$, output $\lambda$ and terminate, and where $P_1$ runs as follows: On input $x_1$, let $s$ denote the first bit of $x_1$ and then create a random secret sharing $(s_1, \ldots, s_4)$ of $s$, send $s_i$ to $P_i$ and then output $\lambda$ and terminate.

The adversary will not need an auxiliary register in the following and will therefore be left out of the equations. Let the adversary make a query, defined by $U^{adv,F}_{\mathrm{QUERY}} : \mathcal{H}_{\mathsf{query}} \to \mathcal{H}_{\mathsf{query}}$, that is only of the dealer and one more party where he puts the response register for the input in perfect superposition. The complete state after applying $U^{adv,F}_{\mathrm{QUERY}}$ on the query register is,

$$|3\rangle^{\mathrm{RW}}_{\mathsf{t}} = \sum_{r,a_{\mathsf{qi}}} \frac{1}{\sqrt{|\mathbb{S}|}} \sqrt{p_r} \big|v_{[n]}(s,r)\big\rangle_{\mathsf{p}} |s\rangle_{\mathsf{i}} \big|o_{[n]}(s,r)\big\rangle_{\mathsf{o}} \big|A, a_{\mathsf{qi}}, 0, 0\big\rangle_{\mathsf{q}}$$

where $A = \{1, 2\}$ The state after applying the oracle is

$$|4\rangle^{\mathrm{RW}}_{\mathsf{t}} = \sum_{r,a_{\mathsf{qi}}} \frac{1}{\sqrt{|\mathbb{S}|}} \sqrt{p_r} \big|v_{[n]}(s,r)\big\rangle_{\mathsf{p}} |s\rangle_{\mathsf{i}} \big|o_{[n]}(s,r)\big\rangle_{\mathsf{o}} \big|A, a_{\mathsf{qi}} \oplus s_A, 0, v_A(s,r)\big\rangle_{\mathsf{q}}$$

If we look at the final part of the query register ($v_A(s,r)$) we can see that it is in a classical state and contains the view of the dealer and one other party, that is, the randomness and one secret share. This uniquely defines the secret and hence the query register must be orthogonal for two different secrets.
In the ideal world assume, for the time being, that $U^{\mathrm{SIM}}_{\mathrm{QUERY}}$ is the identity. The state after the oracle for the ideal functionality is applied is then

$$|5\rangle^{\mathrm{IW}}_{\mathsf{t}} = \sum_{r,a_{\mathsf{qi}}} \frac{1}{\sqrt{|\mathbb{S}|}} \sqrt{p_r} |s,r\rangle_{\mathsf{if}} |s\rangle_{\mathsf{i}} \big|o_{[n]}(s,r)\big\rangle_{\mathsf{o}} |0\rangle_{\mathsf{s}} \big|A, a_{\mathsf{qi}} \oplus s_A, 0, 0\big\rangle_{\mathsf{q}}$$

Since this is in perfect superposition over all values of $a_{\mathsf{qi}}$ we can conclude that,

$$\begin{aligned}|5\rangle^{\mathrm{IW}}_{\mathsf{t}} &= \sum_{r,a_{\mathsf{qi}}} \frac{1}{\sqrt{|\mathbb{S}|}} \sqrt{p_r} |s,r\rangle_{\mathsf{if}} |s\rangle_{\mathsf{i}} \big|o_{[n]}(s,r)\big\rangle_{\mathsf{o}} |0\rangle_{\mathsf{s}} \big|A, a_{\mathsf{qi}} \oplus s_A, 0, 0\big\rangle_{\mathsf{q}} \\ &= \sum_{r,a_{\mathsf{qi}}} \frac{1}{\sqrt{|\mathbb{S}|}} \sqrt{p_r} |s,r\rangle_{\mathsf{if}} |s\rangle_{\mathsf{i}} \big|o_{[n]}(s,r)\big\rangle_{\mathsf{o}} |0\rangle_{\mathsf{s}} \big|A, a_{\mathsf{qi}}, 0, 0\big\rangle_{\mathsf{q}}\end{aligned}$$

and the state the simulator sees is therefore independent of the secret. There's hence no way it can produce two orthogonal states depending on the secret and hence no way to simulate. For the case of a different simulator where $U^{\mathrm{SIM}}_{\mathrm{QUERY}}$ is not the identity, but some unitary operation on $\mathcal{H}_{\mathsf{query}}$, there exists a different adversary that applies the unitary, $\tilde{U}^{\mathrm{ADV}}_{\mathrm{QUERY}} = U^{adv,F}_{\mathrm{QUERY}}(U^{\mathrm{SIM}}_{\mathrm{QUERY}})^{-1}$ instead. The register the simulator sends to the oracle of the ideal functionality is now exactly the same as above, and the same argumentation can now be repeated. It follows that for any such simulator there exists an adversary that cannot be simulated.

## B.4 No simulator for superposition attacks by running classical simulator in superposition

In this section we argue that most protocols cannot be simulated against a superposition attack by running a classical simulator in superposition. This, in some sense, shows that simulators against superposition attacks have to be inherently quantum: they should "globally" manipulate the quantum state they get back from the ideal functionality, not just locally work on the individual classical states in the superposition.

Unfortunately, this makes it unclear how to design a good quantum simulator, even assuming restrictions on $F$ as in Theorem 1 for secret sharing and in the setting with created response registers, as we need fundamentally new techniques. We will later return to how we construct such simulators. For now we want to argue that constructing them will require new techniques.

As discussed above, a natural first attempt to construct a simualtor for a protocol which is classically secure would be to use the classical simulator and produce a superposition of what it produces on those corrupted subsets that occur in the corruption query from the adversary. We can show, however, that this probably does not work in general.

Let us first make clear what we mean by running a classical simulator in superposition: consider a classical machine $S$ which gets as input parties subset $A$, the inputs and outputs of those parties $(s_A, o_A(s))$ and a random string $c$. It then outputs $S(A, s_A, o_A(s), c)$. Running $S$ in superposition now means that on input $|\psi^{\mathrm{SIM}}_{x,s}\rangle = \sum_{A\in F,c} \alpha_{x,s,A}|0\rangle_{\mathsf{s}}|A, s_A, o_A(s), 0\rangle_{\mathsf{q}}$, we output

$$\sum_{A\in F} \alpha_{x,s,A}\sqrt{p_c}|c\rangle_{\mathsf{s}}|A, s_A, o_A(s), S(A, s_A, o_A(s), c)\rangle_{\mathsf{q}} .$$

This means that the state returned to the adversary will be

$$\sum_{A,A'\in F} \alpha_A\alpha^*_{A'}|A\rangle\langle A'| \otimes |s_A, o_A(s)\rangle\langle s_{A'}, o_{A'}(s)| \otimes \sum_c p_c|S(A, s_A, o_A(s), c)\rangle\langle S(A', s_{A'}, o_{A'}(s), c)| .$$

Now consider the following simple example protocol $\delta$: we have 4 parties $P_0, P_1, P_2, P_3$. Player $P_0$ gets as input a bit $s$. He will then secret share it additively among the other parties: he chooses uniformly bits $r_1, r_2$ and sends $r_1$ to $P_1$, $r_2$ to $P_2$ and $s \oplus r_1 \oplus r_2$ to $P_4$. Then all parties outputs the empty string $\lambda$. Let $d$ be the 4-party function $d(\cdot,\cdot,\cdot,\cdot) = (\lambda, \lambda, \lambda, \lambda)$. Clearly, $\delta$ is a perfectly secure evaluation of $d$ against a classical attack where at most 2 parties are corrupted. One might therefore hope that it would be perfectly secure against superposition attacks using superpositions of sets containing only 1 party.

However, we now argue that such security cannot be shown by running any classical simulator $S$ in superposition. To this end, consider the case where we corrupt all parties in equally weighted superposition. This will mean that the simulator has to start from the input state.

$$\frac{1}{2}(|P_0\rangle|s\rangle|0\rangle + \sum_{i=1}^{3} |P_i\rangle|\bot\rangle|0\rangle) .$$

The state has this form since the input and output for $P_0$ is just $s$ and the other parties have no input or output. The state returned will be

$$\rho_{sim,s} = \sum_{A,A'\in\{\{P_0\},\{P_1\},\{P_2\},\{P_3\}\}} \frac{1}{2}|A\rangle\langle A'| \otimes |s_A\rangle\langle s_{A'}| \otimes \sum_c p_c|S(A, s_A, c)\rangle\langle S(A', s_{A'}, c)| .$$

If we define that $s_{\{P_i\}} = s_i$ is some secret $s$ if $i = 0$ and $\bot$ otherwise, and index with $P_i, P_{i'}$ instead of $A, A'$, we can write the state a bit more conveniently as:

$$\rho_{sim,s} = \sum_{i,i'} \frac{1}{2}|P_i\rangle\langle P_{i'}| \otimes |s_i\rangle\langle s_{i'}| \otimes \sum_c p_c|S(i, s_i, c)\rangle\langle S(i', s_{i'}, c)| .$$

On the other hand, we can compute the state $\rho_{real,s}$ that would be returned from a real attack. Define $v_i(s, r_1, r_2)$ to be the view of the protocol for $P_i$, defined as a 2-bit register. Thus

$$v_0(s, r_1, r_2) = (r_1, r_2), v_1(s, r_1, r_2) = (r_1, 0), v_2(s, r_1, r_2) = (r_2, 0), v_3(s, r_1, r_2) = (r_1 \oplus r_2 \oplus s, 0) .$$

This means that the state returned for a particular choice of $s, r_1, r_2$ is

$$|\Psi_{s,r_1,r_2}\rangle = \frac{1}{2}\sum_{i=0}^{3} |P_i\rangle|s_i\rangle|v_i(s, r_1, r_2)\rangle .$$

For fixed $s$, each choice of $r_1, r_2$ occurs with probability $1/4$, so

$$\rho_{real,s} = \sum_{r_1,r_2} \frac{1}{4} |\Psi_{s,r_1,r_2}\rangle\langle\Psi_{s,r_1,r_2}| = \frac{1}{16} \sum_{i,i'} |P_i\rangle\langle P_{i'}||s_i\rangle\langle s_{i'}| \sum_{r_1,r_2} |v_i(s,r_1,r_2)\rangle\langle v_{i'}(s,r_1,r_2)| \,.$$

Consider the part of $\rho_{sim,s}$ that corresponds to $P_i = P_0$ and $P_{i'} = P_1$. Then, since we assume perfect simulation, i.e., $\rho_{sim,s} = \rho_{real,s}$, for any $c$, we must have $S(P_0, s_A, o_A(s), c) = S(0, s, c) = (r_1, r_2)$ and $S(P_{i'}, s_{A'}, o_{A'}(s), c) = S(1, \perp, c) = (r_1, 0)$, where the *same* $r_1$ occurs in both strings, since in a real execution, $P_1$ would of course receive the same bit that $P_0$ sent. We get an exactly similar conclusion for $P_{i'} = P_2$, and for $P_{i'} = P_3$, we can conclude that $S(3, \perp, c) = (s \oplus r_1 \oplus r_2, 0)$.

But now note that, we can simply compute $S(i, \perp, c)$ for $i = 1, 2, 3$ and some fixed $c$. Then the above shows that if running $S$ in superposition was a perfect quantum simulator, we could compute $(s, 0) = S(1, \perp, c) \oplus S(2, \perp, c) \oplus S(3, \perp, c)$, without any information on $s$, which is of course a contradiction.

## B.5 Simulators in the created response registers model

The result in B.3 strongly suggests that it's impossible to construct a simulator for general MPC protocols in the setting with supplied response registers: if not even the above simple secret sharing-based protocol is secure, there is not much hope that more advanced protocols, which actually use the secret sharings for computations, could be proven secure. In this section we will discuss the problem of constructing a simulator for the model with *created* response registers and protocols for deterministic functions. That is, the output does not depend on the chosen randomness. While simulators in general consists of two operators, $\{U^{\mathsf{SIM}}_{\mathsf{QUERY}}, U^{\mathsf{SIM}}_{\mathsf{RES}}\}$, we will restrict $U^{\mathsf{SIM}}_{\mathsf{QUERY}}$ to be the identity. That is, the query from the adversary is sent directly to the oracle. We can do this wlog because there's no values in the response register and $U^{\mathsf{SIM}}_{\mathsf{QUERY}}$ must be corruption preserving.

Assume all randomness is uniformly chosen. This can be done wlog and helps to unclutter the notation. For a specific protocol, define the matrix (or vector of states) $M(A, s)$ as

$$M(A,s) = (|A, v_A(s,0)\rangle, \ldots, |A, v_A(s,r)\rangle, \ldots, |A, v_A(s, |\mathcal{R}|-1)\rangle) \,.$$

We can now express the security of the protocol in the following way

**Lemma 2.** *A multiparty computation protocol for a deterministic function $f$ is perfectly secure against superposition $F$-attacks if, and only if, there exists a set of $|\mathcal{R}| \times |\mathcal{R}|$ unitary matrices $\{U_s\}_{s\in\mathbb{S}}$ such that for all $s, s' \in \mathbb{S}, A \in F_{s,s'}$ where $F_{s,s'} = \{A \in F | s_A = s'_A \wedge o_A(s) = o_A(s')\}$*

$$M(A,s)U_s = M(A,s')U_{s'}$$

*Proof.* Consider the final state the adversary sees (using the output is independent of randomness and only 0 in the response registers),

$$\rho^{\mathsf{RW}}_{\mathsf{ADV}} = \mathrm{tr}_{\mathsf{p}}(|4\rangle\langle 4|^{\mathsf{RW}}_{\mathsf{t}}) = \sum_{r\in\mathcal{R}} \frac{1}{|\mathcal{R}|} |\psi^{\mathsf{ADV}}_r\rangle\langle\psi^{\mathsf{ADV}}_r|$$

where $|\psi^{\mathsf{ADV}}_r\rangle = \sum_{x,s,A\in F} \alpha_{x,s,A} |s\rangle_{\mathsf{i}} |o_{[n]}(s)\rangle_{\mathsf{o}} |x\rangle_{\mathsf{e}} |A, s_A, o_A(s), v_A(s,r)\rangle_{\mathsf{q}}$ and the state the simulator sees after the oracle has been applied in the ideal world,

$$\rho^{\mathsf{IW}}_{\mathsf{SIM}} = \mathrm{tr}_{\mathsf{if,e,i}}(|5\rangle\langle 5|^{\mathsf{RW}}_{\mathsf{t}}) = \sum_{x,s} |\psi^{\mathsf{SIM}}_{x,s}\rangle\langle\psi^{\mathsf{SIM}}_{x,s}|$$

where $|\psi^{\mathsf{SIM}}_{x,s}\rangle = \sum_{A\in F} \alpha_{x,s,A} |0\rangle_{\mathsf{s}} |A, s_A, o_A(s), 0\rangle_{\mathsf{q}}$. The state the simulator must sent back is,

$$\sum_{x,r,s} \frac{1}{|\mathcal{R}|} |\psi_{x,r,s}\rangle\langle\psi_{x,r,s}|$$

where

$$|\psi_{r,s}\rangle = \sum_{A\in F} \alpha_{x,s,A}|A, s_A, o_A(s), v_A(s,r)\rangle_{\mathsf{q}} \,.$$

To continue we need a claim that is almost equivalent to Theorem 4 in [CJW04], but changed slightly for our specific purposes. For this reason we will also provide a separate proof. The reader is encouraged to read [CJW04] for additional information on the existence of transformations between sets of quantum states.

*Claim.* There exists a perfect simulator, i.e., $\rho^{\text{RW}}_{\text{ADV}} = \rho^{\text{IW}}_{\text{ADV}}$, iff there exist unitary operator, $U^{\text{SIM}}_{\text{RES}}$ such that for all $x$, $s \in \mathbb{S}$

$$U^{\text{SIM}}_{\text{RES}}|\psi^{\text{SIM}}_{x,s}\rangle = U^{\text{SIM}}_{\text{RES}} \sum_{A\in F} \alpha_{x,s,A}|0\rangle_{\mathsf{s}}|A, s_A, o_A(s), 0\rangle_{\mathsf{q}} = \frac{1}{\sqrt{|\mathcal{R}|}} \sum_{A\in F} \alpha_{x,s,A} \sum_{i,k\in\mathcal{R}} [U_s]_{i,k}|k\rangle_{\mathsf{s}}|A, s_A, o_A(s), v_A(s,i)\rangle_{\mathsf{q}} \tag{4}$$

where $[U_s]_{i,k}$ is the $i,k$ index of a $|\mathcal{R}| \times |\mathcal{R}|$ unitary matrix in some set of unitary matrices, $\{U_s\}_{s\in\mathbb{S}}$.

*Proof.* For the forward direction, consider the final state the adversary sees

$$\rho^{\text{IW}}_{\text{ADV}} = \text{tr}_{\mathsf{if},\mathsf{s}}(|6\rangle\langle 6|^{\text{IW}}_{\mathsf{t}}) = \sum_{k\in\mathcal{R}, s\in\mathbb{S}} |\psi^{\text{ADV}}_{k,s}\rangle\langle\psi^{\text{ADV}}_{k,s}| \,,$$

where $|\psi^{\text{ADV}}_{k,s}\rangle = \sum_{x,A\in F} \alpha_{x,s,A}|s\rangle_{\mathsf{i}}|o_{[n]}(s)\rangle_{\mathsf{o}}|x\rangle_{\mathsf{e}} \left(\sum_{i\in\mathcal{R}} [U_s]_{i,k}\frac{1}{\sqrt{|\mathcal{R}|}}|A, s_A, o_A(s), v_A(s,i)\rangle_{\mathsf{q}}\right)$. Note that $\forall s, A, A', x, x'$,

$$\begin{aligned} &\frac{1}{|\mathcal{R}|} \sum_{i,j,k'\in\mathcal{R}} [U^*_s]_{i,k}[U_s]_{j,k}|A, s_A, o_A(s), v_A(s,i)\rangle\langle A', s_{A'}, o_{A'}(s), v_{A'}(s,j)| \qquad (5)\\ &= \frac{1}{|\mathcal{R}|} \sum_{i,k\in\mathcal{R}} [U^*_s]_{i,k}[U_s]_{i,k}|A, s_A, o_A(s), v_A(s,i)\rangle\langle A', s_{A'}, o_{A'}(s), v_{A'}(s,i)| \\ &= \frac{1}{|\mathcal{R}|} \sum_{i\in\mathcal{R}} p_i|A, s_A, o_A(s), v_A(s,i)\rangle\langle A', s_{A'}, o_{A'}(s), v_{A'}(s,i)| \end{aligned}$$

where we used that $[U_s]_{i,k}$ is unitary and only depends on $s$. This means the adversary sees the state he expects (except for labels) and we've completed the proof for the forward direction.
Conversely, assume for contradiction that there is a simulator that can constructs the correct mixed state for the adversary, but none that can construct one of the form of equation (4). If $\sum_{x,k,s} p_k|\phi_{x,k,s}\rangle\langle\phi_{x,k,s}|$ is the state the simulator sents back, then it must be that $\forall x, s$ :

$$\sum_k p_k|\phi_{x,k,s}\rangle\langle\phi_{x,k,s}| = \sum_r \frac{1}{|\mathcal{R}|}|\psi_{x,r,s}\rangle\langle\psi_{x,r,s}|$$

where

$$|\psi_{r,s}\rangle = \sum_A \alpha_{x,s,A}|A, s_A, o_A(s), v_A(s,r)\rangle_{\mathsf{q}} \,.$$

This is true if, and only if, there exist unitary matrices, $\{U_{x,s}\}_{x,s}$, such that,

$$\forall k : \sqrt{p_k}|\phi_{x,k,s}\rangle = \frac{1}{\sqrt{|\mathcal{R}|}} \sum_i [U_{x,s}]_{i,k}|\psi_{x,i,s}\rangle \,.$$

By purifying the state we see we get,

$$\sum_k \sqrt{p_k}|k\rangle|\phi_{x,k,s}\rangle = \frac{1}{\sqrt{|\mathcal{R}|}} \sum_{i,k} [U_{x,s}]_{i,k}|k\rangle|\psi_{x,i,s}\rangle = \frac{1}{\sqrt{|\mathcal{R}|}} \sum_{A\in F} \alpha_{x,s,A} \sum_{i,k} [U_{x,s}]_{i,k}|k\rangle_{\mathsf{s}}|A, s_A, o_A(s), v_A(s,i)\rangle_{\mathsf{q}} \,.$$

Note that we can choose the purification because all purifications are unitarily equivalent. And since the value of $x$ only effects the amplitude we can, by linearity, assume that $U_{x,s}$ does not depend on $x$. This shows that the state produced by any perfect simulator must be of that form. This is a contradiction and completes the proof. $\square$

Note that $U_{\mathtt{RES}}^{\mathtt{SIM}}$ exists (and is unitary) if it preserves inner product between all possible states, hence we can conclude that there exists a perfect simulator if there exists a set of unitary matrices, $\{U_s\}_{s\in\mathbb{S}}$ such that for all $x, x', s, s' \in \mathbb{S}$ and all queries $\{\alpha_{x,s,A}\}$, $\{\alpha'_{x',s',A}\}$

$$\Big(\sum_{A\in F}\alpha^*_{x,s,A}\langle 0|_{\mathsf{s}}\langle A, s_A, o_A(s), 0|_{\mathsf{q}}\Big)\Big(\sum_{A'\in F}\alpha'_{x',s',A'}|0\rangle_{\mathsf{s}}|A', s'_{A'}, o_{A'}(s'), 0\rangle_{\mathsf{q}}\Big)$$
$$= \frac{1}{|\mathcal{R}|}\left(\sum_{A\in F}\alpha^*_{x,s,A}\sum_{i,k\in\mathcal{R}}[U^*_s]_{i,k}\langle k|_{\mathsf{s}}\langle A, s_A, o_A(s), v_A(s,i)|_{\mathsf{q}}\right)$$
$$\left(\sum_{A'\in F}\alpha'_{x',s',A'}\sum_{j,k'\in\mathcal{R}}[U_{s'}]_{j,k'}|k'\rangle_{\mathsf{s}}|A', s'_{A'}, o_{A'}(s'), v_{A'}(s',j)\rangle_{\mathsf{q}}\right) .$$

For the LHS we have that,

$$\Big(\sum_{A\in F}\alpha^*_{x,s,A}\langle 0|_{\mathsf{s}}\langle A, s_A, o_A(s), 0|_{\mathsf{q}}\Big)\Big(\sum_{A'\in F}\alpha'_{x',s',A'}|0\rangle_{\mathsf{s}}|A', s'_{A'}, o_{A'}(s'), 0\rangle_{\mathsf{q}}\Big)$$
$$= \sum_{A}\alpha^*_{x,s,A}\alpha'_{x',s',A}\langle s_A, o_A(s)||s'_A, o_A(s')\rangle = \sum_{A\in F_{s,s'}}\alpha^*_{x,s,A}\alpha'_{x',s',A} .$$

For the RHS we get that,

$$\frac{1}{|\mathcal{R}|}\left(\sum_{A\in F}\alpha^*_{x,s,A}\sum_{i,k\in\mathcal{R}}[U^*_s]_{i,k}\langle k|_{\mathsf{s}}\langle A, s_A, o_A(s), v_A(s,i)|_{\mathsf{q}}\right)$$
$$\left(\sum_{A'\in F}\alpha'_{x',s',A'}\sum_{j,k'\in\mathcal{R}}[U_{s'}]_{j,k'}|k'\rangle_{\mathsf{s}}|A', s'_{A'}, o_{A'}(s'), v_{A'}(s',j)\rangle_{\mathsf{q}}\right)$$
$$= \frac{1}{|\mathcal{R}|}\sum_{A\in F,k,i,j\in\mathcal{R}}\alpha^*_{x,s,A}\alpha'_{x',s',A}[U^*_s]_{i,k}[U_{s'}]_{j,k}\langle s_A, o_A(s), v_A(s,i)||s'_A, o_A(s'), v_A(s',j)\rangle$$
$$= \frac{1}{|\mathcal{R}|}\sum_{s,s',A\in F,k\in\mathcal{R}}\alpha^*_{x,s,A}\alpha'_{x',s',A}\langle s_A, o_A(s)||s'_A, o_A(s')\rangle \times \left(\sum_{i\in\mathcal{R}}[U^*_s]_{i,k}\langle A, v_A(s,i)|\right)\left(\sum_{j\in\mathcal{R}}[U_{s'}]_{j,k}|A, v_A(s',j)\rangle\right)$$
$$= \frac{1}{|\mathcal{R}|}\sum_{s,s',A\in F_{s,s'},k\in\mathcal{R}}\alpha^*_{x,s,A}\alpha'_{x',s',A} \times \left(\sum_{i\in\mathcal{R}}[U^*_s]_{i,k}\langle A, v_A(s,i)|\right)\left(\sum_{j\in\mathcal{R}}[U_{s'}]_{j,k}|A, v_A(s',j)\rangle\right) .$$

The maximum value of the inner product of two normalized vectors is 1, so the maximum value of

$$\frac{1}{|\mathcal{R}|}\left(\sum_{i\in\mathcal{R}}[U^*_s]_{i,k}\langle A, v_A(s,i)|\right)\left(\sum_{j\in\mathcal{R}}[U_{s'}]_{j,k}|A, v_A(s',j)\rangle\right)$$

is $\frac{1}{|\mathcal{R}|}$. Hence for the sum is over $\mathcal{R}$ to be 1, it must be that

$$\sum_{i\in\mathcal{R}}[U_s]_{i,k}|A, v_A(s,i)\rangle = \sum_{j\in\mathcal{R}}[U_{s'}]_{j,k}|A, v_A(s',j)\rangle.$$

That is, there exists a simulator iff there exist $|\mathcal{R}| \times |\mathcal{R}|$ unitary matrices $U_s$ such that for all $s, s' \in \mathbb{S}, k \in \mathcal{R}, A \in F_{s,s'}$

$$\sum_{i \in \mathcal{R}} [U_s]_{i,k} |A, v_A(s, i)\rangle = \sum_{j \in \mathcal{R}} [U_{s'}]_{j,k} |A, v_A(s', j)\rangle \ . \tag{6}$$

Using the definition of $M(A, s)$ we can now write the requirement in the more convenient way: There exists a simulator for the MPC protocol if, and only if, there exist a set of unitary matrices, $\{U_s\}_{s \in \mathbb{S}}$, such that for all $s, s' \in \mathbb{S}, A \in F_{s,s'}$:

$$M(A, s)U_s = M(A, s')U_{s'}$$

which completes the proof of Lemma 2. □

We'd like to warn the reader that it is important not to confuse the unitary matrices, $\{U_s\}_{s \in \mathbb{S}}$, with the simulator itself. They are merely a tool to show the existence of one. Also, at this point the reader would be excused for lacking any intuition on why the lemma is reasonable. We therefore find it instructive to consider two examples. First we'll reconsider secret sharing in this framework and secondly a simple MPC protocol.

## B.6 Secret sharing example

In this section we show that a simple secret sharing based protocol is a secure implementation of a dummy function in the MPC model with created response registers. The section serves as a sanity check of the model, as one should expect the given protocol to be a secure implementation of the given function. The section also shows an application of Lemma 2.

Consider a secret sharing scheme for $n$ parties $P_1, \ldots, P_n$ and an MPC setting with $n' = n + 1$ parties $P_0, P_1, \ldots, P_n$. Consider the dummy function $d(x_0, x_1, \ldots, x_n) = (\lambda, \ldots, \lambda)$, where $\lambda$ is the empty string. Consider the protocol $\pi$, where $P_0$ parses $x_0$ as a secret $s$ for the secret sharing scheme, secret shares $s$ into $(s_1, \ldots, s_n)$ and sends $s_i$ to $P_i$. Then all parties output $\lambda$. Let $t$ be the threshold of the secret sharing scheme. By Theorem 1 we would expect $\delta$ to be a secure implementation of $d$ against a superpostion attack of $t/2$ of the parties $P_1, \ldots, P_n$. We show that this is indeed the case.

**Theorem 5.** *Let $F = \{A \subseteq \{P_1, \ldots, P_n\} : |A| \leq t/2\}$. The protocol $d$ is a secure implementation of $\delta$ under superposition $F$- attacks.*

In the proof, we will allow each choice of randomness to be non-uniform as it adds only very little clutter. We note that since none of the parties $P_1, \ldots, P_n$ have any consequential input or output to $\delta$ we have that for all $s, s' \in \mathbb{S} : F_{s,s'} = F$. Note, in particular, that $P_0 \notin A$ for all $A \in F$, so the secret $s$ is never an input of a corrupted party. The result then follows as a Corollary of Theorem 1.

**Corollary 1.** *If a secret sharing scheme $\mathcal{S}$ is perfectly secure against superposition $F$-attacks then there exists a set of $|\mathcal{R}| \times |\mathcal{R}|$ unitary matrices $\{U_s\}_{s \in \mathbb{S}}$ such that for all $s, s' \in \mathbb{S}, r \in \mathcal{R}, A \in F$*

$$\sum_{i \in \mathcal{R}} \sqrt{p_i} [U_s]_{i,r} |v_A(i, s)\rangle = \sum_{j \in \mathcal{R}} \sqrt{p_j} [U_{s'}]_{j,r} |v_A(j, s')\rangle \ .$$

*Proof.* Since the secret sharing scheme is secure we know from Theorem 1 that the joint distribution of the view for any $A, A' \in F$ is independent of s and hence, for all $s, s' \in \mathbb{S}$:

$$\sigma_s = \sum_{r \in \mathcal{R}} p_r |\psi_{s,r}\rangle\langle\psi_{s,r}| = \sigma_{s'} = \sum_{r \in \mathcal{R}} p_r |\psi_{s',r}\rangle\langle\psi_{s',r}|$$

where

$$|\psi_{s,r}\rangle = \sum_{A \in F} \alpha_A |A\rangle |v_A(r, s)\rangle \ .$$

This is equivalent to saying that the fidelity of two such states is 1. According to Uhlmann's Theorem this implies that there exist purifications of $\sigma_s$ and $\sigma_{s'}$ such that their inner product is 1. Unitary equivalence of purifications implies you can write any such purification in $\mathcal{H} \otimes \mathcal{R}$, where $\dim(\mathcal{R}) = |\mathcal{R}|$, as

$$\sum_{k\in\mathcal{R}} \sqrt{p'_k}|\psi'_{s,k}\rangle|k\rangle_{\mathcal{R}} = \sum_{i,k\in\mathcal{R}} \sqrt{p_i}[U_s]_{i,k}|\psi_{s,i}\rangle|k\rangle_{\mathcal{R}}$$

where $\{U_s\}_{s\in\mathbb{S}}$ is a set of $|\mathcal{R}| \times |\mathcal{R}|$ unitary matrices. As the fidelity must be 1, it must be that for all $s, s' \in \mathbb{S}$

$$\left(\sum_{i,k\in\mathcal{R}} \sqrt{p_i}[U_s^*]_{i,k}\langle\psi_{s,i}|\langle k|_{\mathcal{R}}\right)\left(\sum_{j,k'\in\mathcal{R}} \sqrt{p_j}[U_{s'}]_{j,k'}|\psi_{s',j}\rangle|k'\rangle_{\mathcal{R}}\right) = \sum_{i,j,k\in\mathcal{R}} \sqrt{p_i}\sqrt{p_j}[U_s^*]_{i,k}[U_{s'}]_{j,k}\langle\psi_{s,i}||\psi_{s',j}\rangle$$

$$= \sum_{A\in F,k\in\mathcal{R}} |\alpha_A|^2\left(\sum_{i\in\mathcal{R}} \sqrt{p_i}[U_s^*]_{i,k}\langle v_A(i,s)|\right)\left(\sum_{j\in\mathcal{R}} \sqrt{p_j}[U_{s'}]_{j,k}|v_A(j,s')\rangle\right) = 1\,.$$

Because $\sum_{A\in F} |\alpha_A|^2 = 1$ and noting the states are normalized, we can conclude that

$$\forall s, s' \in \mathbb{S}, k \in \mathcal{R}, A \in F : \left(\sum_{i\in\mathcal{R}} \sqrt{p_i}[U_s^*]_{i,k}\langle v_A(i,s)|\right)\left(\sum_{j\in\mathcal{R}} \sqrt{p_j}[U_{s'}]_{j,k}|v_A(j,s')\rangle\right) = 1\,,$$

from which the result follows. □

## B.7 Simple MPC simulator example

For this example we return to the simple four-party $(P_0, P_1, P_2, P_3)$ bit-sharing scheme considered in section B.4. There we showed that we could not simulate an attack by simply running a classical simulator for the protocol in superposition. We are now in position to show that a simulator nonetheless exists. Recall that $|v_{P_0}(s,(r_0,r_1))\rangle = |r_0, r_1\rangle, |v_{P_1}(s,(r_0,r_1))\rangle = |r_0, 0\rangle, |v_{P_2}(s,(r_0,r_1))\rangle = |r_1, 0\rangle, |v_{P_3}(s,(r_0,r_1))\rangle = |r_0 \oplus r_1 \oplus s, 0\rangle$. We have two possible inputs $s \in \{0,1\}$ so we need to find two unitary matrices, $U_0, U_1$ in order to apply Lemma 2. These have been found manually.

$$U_0 = \begin{pmatrix} 1&0&0&0 \\ 0&1&0&0 \\ 0&0&1&0 \\ 0&0&0&1 \end{pmatrix}$$

$$U_1 = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{pmatrix}.$$

It is a tedious, but straightforward, calculation to show that[9],

$$M(P_1, 0) = M(P_1, 1)U_1$$
$$M(P_2, 0) = M(P_2, 1)U_1$$
$$M(P_3, 0) = M(P_3, 1)U_1$$

[9] Note that we encode the randomness as $(0,0) = 0, (0,1) = 1, (1,0) = 2, (1,1) = 3$

and since $P_0 \notin F_{0,1}$ we do *not* require that $M(P_0, 0) = M(P_0, 1)U_1$. To get a better feeling for what is going on we will consider equation (6) for two specific choices for $A$ and $r$. First let $A = P_2$ and $r = (1, 0)$. The following must be true for the unitary matrices to be correct choices,

$$|P_2, v_{P_2}(0, (1, 0))\rangle = \sum_{(i_0,i_1)\in\mathcal{R}} [U_1]_{(i_0,i_1),(1,0)} |P_2, v_{P_2}(1, (i_0, i_1))\rangle$$

Since $U_0$ is just the identity, the LHS is easy to calculate, $|P_2, v_{P_2}(0, (1, 0))\rangle = |0, 0\rangle$ For the RHS we see that

$$\begin{aligned}&\sum_{(i_0,i_1)\in\mathcal{R}} [U_1]_{(i_0,i_1),(1,0)} |P_2, v_{P_2}(1, (i_0, i_1))\rangle \\ &= \frac{1}{2}|P_2, v_{P_2}(1, (0, 0))\rangle - \frac{1}{2}|P_2, v_{P_2}(1, (0, 1))\rangle + \frac{1}{2}|P_2, v_{P_2}(1, (1, 0))\rangle + \frac{1}{2}|P_2, v_{P_2}(1, (1, 1))\rangle \\ &= \frac{1}{2}|0, 0\rangle - \frac{1}{2}|1, 0\rangle + \frac{1}{2}|0, 0\rangle + \frac{1}{2}|1, 0\rangle = |0, 0\rangle\ .\end{aligned}$$

Of particular interest is $A = P_3$ as the view depends on the secret (for fixed randomness). Choose $r = (0, 0)$:

$$|P_3, v_{P_3}(0, (0, 0))\rangle = \sum_{(i_0,i_1)\in\mathcal{R}} [U_1]_{(i_0,i_1),(0,0)} |P_3, v_{P_3}(1, (i_0, i_1))\rangle\ .$$

For LHS, $|P_3, v_{P_3}(0, (0, 0))\rangle = |0, 0\rangle$. For RHS,

$$\begin{aligned}&\sum_{(i_0,i_1)\in\mathcal{R}} [U_1]_{(i_0,i_1),(0,0)} |P_3, v_{P_3}(1, (i_0, i_1))\rangle \\ &= \frac{1}{2}|P_3, v_{P_3}(1, (0, 0))\rangle + \frac{1}{2}|P_3, v_{P_3}(1, (0, 1))\rangle + \frac{1}{2}|P_3, v_{P_3}(1, (1, 0))\rangle - \frac{1}{2}|P_3, v_{P_3}(1, (1, 1))\rangle \\ &= \frac{1}{2}|1, 0\rangle + \frac{1}{2}|0, 0\rangle + \frac{1}{2}|0, 0\rangle - \frac{1}{2}|1, 0\rangle = |0, 0\rangle\ .\end{aligned}$$

### B.8 General MPC

In this section we will give a restatement of Lemma 2, expressing the requirement for the existence of a simulator as an explicit property of the multiparty computation protocol. This will allow for a straightforward, albeit extremely inefficient, method for checking the security of any MPC protocol for a deterministic function in our model. For all ordered pairs of inputs, $s, s' \in \mathbb{S}$, and all sets $A \in F_{s,s'}$ we will associate a permutation of the randomness, $\{\pi_{s,s',A}\}_{s,s',A\in F_{s,s'}} \in S(\mathcal{R})$. By ordered pairs we mean that $\pi_{s,s',A}$ may differ from $\pi_{s',s,A}$.

**Theorem 6.** *A multiparty computation protocol for a deterministic function $f$ is perfectly secure against superposition $F$-attacks with created response registers if, and only if, it is correct and there exist permutations, $\{\pi_{s,s',A}\}_{s,s',A\in F_{s,s'}}$ with the following two properties,*

*1.*

$$\forall s, s' \in \mathbb{S}, \forall A \in F_{s,s'}, \forall r \in \mathcal{R} : \\ \big|v_A(s, \pi_{s,s',A}(r))\big\rangle = \big|v_A(s', \pi_{s',s,A}(r))\big\rangle$$

*2.*

$$\forall s, s', s'' \in \mathbb{S}, \forall A \in F_{s,s'}, A' \in F_{s,s''} : \\ \sum_{r\in\mathcal{R}} |v_A(s, r)\rangle\langle v_{A'}(s, r)| = \sum_{r\in\mathcal{R}} \big|v_A(s, \pi_{s,s',A}(r))\big\rangle\big\langle v_{A'}(s, \pi_{s,s'',A'}(r))\big|$$

*Note that property (1) is exactly the statement that a (not necessarily efficient) simulator exists in the classical model.*

*Proof.* Before we begin we will need the following claim,

*Claim.* We can without loss of generality assume that all the rows (and columns) of $U_s$ sum to 1.

*Proof.* First recall that according to Lemma 2 we have that there is a simulator iff

$$\exists U_0, \cdots, U_{|\mathbb{S}|-1} \text{ st.}$$
$$\forall s, s' \in \mathbb{S}, \forall A \in F_{s,s'} : M(A,s)U_s = M(A,s')U_{s'} \ .$$

For any solution we can always multiply with $U_0^{-1}$ on the right side of all equations and we can hence wlog assume that $U_0 = I$. Now consider $\forall s \in \mathbb{S}$ the equation,

$$\forall A \in F_{0,s} : M(A,0) = M(A,s)U_s \ .$$

That is,

$$\forall r \in \mathcal{R} : \sum_{k \in \mathcal{R}} [U_s]_{r,k} |A v_A(s,k)\rangle = |v_A(0,r)\rangle \ .$$

Hence all rows (and columns) of $U_s$ must sum to 1. □

We can, in other words, view $M(A_{0,s}, s)U_s$ simply as $M(A_{0,s}, 0)$ under some permutation of the columns. In fact, since $U_s$ must always preserve the length of the columns, any $M(A,s)U_s$ is always just a permutation of the columns in $M(A,s)$. Although the permutation is defined by the same unitary it is not necessarily the same permutation. But they are clearly related. We will make this relationship explicit by separating the requirement for a simulator into two parts. Let $M^{\pi_{s,s',A}}(A,s)$ denote the permutation of the columns in $M(A,s)$ that corresponds to applying the permutation function to the randomness for the view in each column, that is,

$$M^{\pi_{s,s',A}}(A,s) = \left(\big|A, v_A(s, \pi_{s,s',A}(0))\big\rangle, \ldots, \big|A, v_A(s, \pi_{s,s',A}(r))\big\rangle, \ldots\right) \ .$$

The first requirement is simply the statement that for $A \in F_{s,s'}$ some permutation of the randomness exist to allow their views to be equal for $s$ and $s'$. The second is the statement that, when the input to the parties are the same, these permutations must be performed by the same unitary matrix. That is, there exists a simulator if, and only if, there exist permutations, $\{\pi_{s,s',A}\}_{s,s',A \in F_{s,s'}}$ with the following two properties,

1.
$$\forall s, s' \in \mathbb{S}, \forall A \in F_{s,s'} :$$
$$M^{\pi_{s,s',A}}(A,s) = M^{\pi_{s',A}}(A,s')$$

2. There exist unitary matrices, $\{U_s\}_{s \in \mathbb{S}}$, such that $\forall s, s', A \in F_{s,s'} : M(A,s)U_s = M^{\pi_{s,s',A}}(A,s)$.

For a specific choice of permutations such unitary matrices exist iff they preserve the inner product. That is, iff

$$\forall s, s', s'' \in \mathbb{S}, \forall A \in F_{s,s'}, A' \in F_{s,s''} :$$
$$M(A,s)M(A',s)^\dagger = M^{\pi_{s,s',A}}(A,s)M^{\pi_{s,s'',A'}}(A',s)^\dagger \ .$$

Writing out the equations using the definition of $M(A,s)$ we conclude the proof. □

Had the choice of corrupted parties been classical we could have let the unitary matrices depend on both the input *and which party was corrupted.* Specifically, in equation (5), the reason the unitaries cannot depend on A is that $\sum_{i,j,k \in \mathcal{R}} [U^*_{s,A}]_{i,k}[U_{s,A'}]_{j,k}$ does not cancel out correctly if they differ and the adversary would not see the correct state. If the choice of $A$ was classical we would see no such cross-terms [10] and no such relationship would be required.

[10] Recall that there are no cross-terms for $s$ because the input register is perfectly entangled with the parties/ideal functionality