Fully Simulatable Quantum-Secure Coin-Flipping and Applications

Carolin Lunemann and Jesper Buus Nielsen

Department of Computer Science, Aarhus University, Denmark
{carolin|jbn}@cs.au.dk

Abstract We propose a coin-flip protocol which yields a string of strong, random coins and is fully simulatable against poly-sized quantum adversaries on both sides. It can be implemented with quantum-computational security without any set-up assumptions, since our construction only assumes mixed commitment schemes which we show how to construct in the given setting. We then show that the interactive generation of random coins at the beginning or during outer protocols allows for quantum-secure realizations of classical schemes, again without any set-up assumptions. As example applications we discuss quantum zero-knowledge proofs of knowledge and quantum-secure two-party function evaluation. Both applications assume only fully simulatable coin-flipping and mixed commitments. Since our framework allows to construct fully simulatable coin-flipping from mixed commitments, this in particular shows that mixed commitments are *complete* for quantum-secure two-party function evaluation. This seems to be the first completeness result for quantum-secure two-party function evaluation from a generic assumption.

1 Introduction

True randomness is a crucial ingredient in many cryptographic applications. Therefore, secure coin-flipping is an essential primitive, which allows two parties to agree on a uniformly random bit in a fair way, such that neither party can influence the value of the coin to his advantage. We investigate coin-flip protocols with classical messages exchange but where the adversary is assumed to be capable of quantum computing. Security of cryptographic protocols in the quantum world means, of course, that quantum computation does not jeopardize the assumption, underlying the protocol construction. However, we encounter additional setbacks in the security proofs, which are mostly due to the fact that some well-known classical proof techniques cannot be applied in a quantum environment.

Our Contribution. We aim at establishing coin-flipping as a stand-alone tool in a model without any setup assumptions. As such, our protocol can be used in several contexts and different generic constructions. One notable application is as subroutine for realizing the theoretical assumption of the common-random-string-model (CRS-model).[1] Since the generation of a CRS often significantly

[1] In the CRS-model the parties are provided with a public common random string CRS before communication, taken from the uniform distribution.

simplifies the design of (quantum-secure) protocols, this then implies that various interesting applications can be implemented quantum-securely in a simple manner from scratch.

In more detail, we first investigate different degrees of security that a coin-flip protocol can acquire. Then, we propose and prove constructions that allow us to amplify the respective degrees of security such that weaker coins are converted into very strong ones.[2] The amplification only requires mixed commitment schemes, which we know how to construct with quantum security under reasonable assumptions—for instance, based on the quantum hardness of the learning with error problem. Combining our amplification protocols allows to take a very weak notion of coin-flipping and amplify it to a coin-flip protocol which is *fully simulatable against poly-sized quantum adversaries*. By fully simulatable we mean that both sides can be simulated in quantum polynomial time.

Our amplification framework should also be understood as a step towards fully simulatable *constant-round* coin-flipping. To the best of our knowledge, to date there does not exist any fully simulatable protocol which is constant-round and which allows to generate a long random bit-string. In particular, no fully simulatable constant-round coin-flip protocol is known to securely compose in parallel. Since all our amplification protocols work in constant-round, we show that if there exists a constant-round coin-flip protocol of long strings with weak security, then there also exists a constant-round coin-flip protocol of long strings which is fully simulatable. Even though our work leaves fully simulatable constant-round coin-flipping of long strings as a fascinating open problem, we consider it a contribution in itself to define a reasonably weak but sufficient security notion to realize fully simulatable constant-round coin-flipping of long strings.

Related Work. The standard coin-flip protocol of [2] was proven secure in a quantum environment in previous work [7]. In its basic form this protocol yields *one* coin as output. Of greater importance, however, is flipping a string of coins instead of a bit, in particular, when generating a CRS. The basic construction composes in sequence with security classified as medium in our framework here. Parallel composition is possible using an extended construction providing efficient simulations on both sides. This extension, however, requires a CRS as initial assumption, i.e. the CRS-model, and hence, violates our strong requirement of applications, implementable quantum-securely without any set-up assumptions.

As an example application, we discussed in [7] the generation of a CRS in the context of e.g. a quantum zero-knowledge proof. For an overview and more details, see also [14]. To further show the implications of coin-flipping as an implementation of the CRS-model in the quantum setting, we here add the functionalities of a quantum zero-knowledge proof of knowledge and quantum-secure function evaluation. We want to mention the following related work. First, an alternative approach in the context of zero-knowledge was independently inves-

[2] For clarity, we note that we use the intuitive interpretation of "weak" and "strong" coins related to their security degrees, which differs from the definitions in the quantum literature.

tigated by Smith [17]. There, coin-flipping is implemented by a string commitment with special openings and validated in subsequent zero-knowledge proofs in sequence, and which therefore has round complexity depending on the security parameter, i.e. how many proofs must be completed to achieve a negligible soundness error. The coin-string is used as key to encode the witness and more zero-knowledge proofs are given to prove that. As encryption scheme, they suggest a scheme with similar properties as in the standard construction for mixed commitments [4, 5, 8]. To the best of our knowledge, the question of its actual secure implementation was left open, and a formal description and analysis was never published. Second, we want to mention the concurrent and independent work of Hallgren, Smith, and Song, as sketched in [12]. They also prove, among other things, classical protocols for zero-knowledge proofs of knowledge and function evaluation secure in the quantum setting by proposing a composition theorem that allows to use the basic coin-flipping protocol in [7] to generate a CRS. In addition, they give a UC-secure protocol for said tasks in the CRS-model.

Furthermore, the techniques used in our reductions are inspired by techniques used by works in the UC framework (cf. [8]), where rewinding is also a problem. But to the best of our knowledge, all our reductions are novel, and might be also of classical interest.

Security in the Quantum World. It is well known that bit commitments imply a single coin-flip—in the classical as in the quantum world [2, 7]—in a straightforward way: Alice chooses a random bit a and commits to it, Bob then sends his bit b in plain, then the commitment is opened, and the resulting coin is $a \oplus b$. However, even when basing the embedded commitment scheme on a computational assumption that withstands quantum attacks (for the hiding property), the security proof of the outer coin-flipping (and its integration into other applications) cannot easily be translated from the classical to the quantum world. Typically, security against a classical adversary is argued in this context by rewinding the adversary in a simulation. In brief, it is shown that a run of a protocol between a dishonest Bob and honest Alice can be efficiently simulated without interacting with Alice but with a simulator instead. A simulator basically prepares a valid conversation and tries it on dishonest Bob. Now, in case Bob does not send the expected reply, we need the possibility to rewind him. Then to conclude the proof, we have to show that the expected running time of the simulation is polynomial.

Unfortunately, rewinding as a proof technique can generally not be directly applied in the quantum world, i.e., if the dishonest machine is a quantum computer. First, we cannot trivially copy and store an intermediate state of a quantum system, and second, quantum measurements are in general irreversible. In order to produce a classical transcript, the simulator would have to partially measure the quantum system without copying it beforehand, but then it would become impossible to reconstruct all information necessary for correct rewinding [11]. It is worth mentioning though that rewinding in the quantum world is possible in a limited setting, as shown by Watrous [18]. This technique was also used for proving the quantum security of single coin-flipping based on bit

commitments [7]. However, the generation of a string of coin must be based on string commitments. In this setting, the simulator cannot rewind in poly-time. A possible solutions for simulating against a classical Bob is then to let him commit to his message in a way which allows to extract the message in the simulation. Therewith, the message is known to the simulator in any following iteration of rewinding. This technique seems to be doomed to fail in the quantum realm, since it is neither known how to rewind quantumly for string commitments nor can any intermediate status (such as Bob's commitment) be preserved. Moreover, commitment constructions providing flavors of extractability without rewinding require some stronger set-up assumptions. Thus, other techniques such as our method based on mixed commitments, are needed for solutions in this context.
Applications. Even though we establish coin-flipping as a stand-alone tool, we highlight again that the generation of a CRS leads to a simple and quantum-secure implementation of various interesting applications without any set-up assumptions. We show two different example applications, in addition to the functionalities already discussed in [7]. First, we propose a *quantum-secure zero-knowledge proof of knowledge* based on a witness encoding scheme, which we define such that it provides a certain degree of extractability and simulatability in the quantum world. Our zero-knowledge construction only requires mixed commitments, which can be implemented with quantum security. This is of particular interest, as the problems of rewinding in the quantum realm complicate implementing proofs of knowledge from scratch. And second, we show that mixed commitment schemes are sufficient for *quantum-secure function evaluation* of *any* classical poly-time function f with security against active quantum adversaries. In more detail, we first show that mixed commitments imply an oblivious transfer protocol with passive security. From that it is straightforward to construct a protocol for any classical poly-time function with security against passive quantum adversaries [13]. As our main result in that context, we then propose a quantum-secure implementation for evaluating any such function with security against active quantum adversaries.

2 Preliminaries

Notation. We use $negl\,(n)$ to denote the set of *negligible* functions (in n). For a bit-string $x \in \{0,1\}^n$ and a subset $S \subseteq \{1,\ldots,n\}$ of size s, we define $x|_S \in \{0,1\}^s$ to be the restriction $(x_i)_{i\in S}$. The *probability* of event E is denoted by $\Pr[E]$. For a random variable X we use P_X to denote the *distribution* of X, and for an additional random variable Y we use $P_{X|Y}$ to denote the *conditional distribution* of X given Y. *Statistical indistinguishability* of families of classical random variables is denoted by $\overset{s}{\approx}$, and $\overset{q}{\approx}$ indicates *quantum poly-time indistinguishability* of families of random variables, i.e., the families cannot be distinguished by poly-sized families of quantum circuits.

Definition of Security. We are interested in classical two-party protocols secure in a quantum world. We work in the security framework, introduced in [9]

and extended in [4]. The definitions are proposed for quantum protocols that implement *classical non-reactive two-party functionalities*, meaning that in- and output must be classical. The framework allows functionalities which behave differently in case of a dishonest player, and it is further shown that any protocol in the framework *composes sequentially* in a classical environment, i.e. within an outer classical protocol. For the sake of simplicity, the framework does not assume additional entities such as e.g. an environment. The original security definitions for unconditional security [9] are phrased in simple information-theoretic conditions, depending on the functionality, which implies strong simulation-based security. In [4], it is then shown that computational security (in the CRS-model) can be defined similarly. In the following, we state the formalism essential for this work.[3] For more details on the framework and notation, we refer to [4, 6, 9], and to [14] for an overview.

Our protocols run between players Alice (A) and Bob (B) and all definitions are given in the *two-world paradigm* of simulation-based proofs. The *real world* captures the actual protocol Π, consisting of message exchange between the parties and local computations. Real-world players are denoted by honest A, B and are restricted to poly-time classical strategies. Dishonest players A', B' are allowed any *quantum* poly-time strategy. Formally, let $\mathfrak{P}$ denote the set of poly-size quantum circuits, so we assume that $\mathsf{A}', \mathsf{B}' \in \mathfrak{P}$. The ideal functionality $\mathcal{F}$ models the intended behavior of the protocol in the *ideal world*, where the players interact using $\mathcal{F}$. Honest and dishonest players in the ideal world (a.k.a. simulators) are denoted by $\hat{\mathsf{A}}, \hat{\mathsf{B}}$ and $\hat{\mathsf{A}}', \hat{\mathsf{B}}'$, respectively. An honest player simply forwards messages to and from $\mathcal{F}$, dishonest players are allowed to change their messages. Again $\hat{\mathsf{A}}', \hat{\mathsf{B}}' \in \mathfrak{P}$. Now, the input-output behavior of $\mathcal{F}$ defines the required input-output behavior of Π. Intuitively, if the executions are indistinguishable, security of the protocol in real life follows. In other words, a dishonest real-world player that attacks protocol Π cannot achieve (significantly) more than an ideal-world adversary that attacks the corresponding functionality $\mathcal{F}$.

The common input state $\rho_{UV} = \sum_{u,v} P_{UV}(u,v)|u\rangle\langle u| \otimes |v\rangle\langle v|$ for some probability distribution P_{UV} is classical, and we understand U, V as random input variables (for Alice and Bob, respectively). The same holds for the classical output state ρ_{XY} with output X, Y for Alice respectively Bob. The input-output behavior of the protocol is uniquely determined by $P_{XY|UV}$, and we write $\Pi(U,V) = (X,Y)$. Then, a general classical ideal functionality $\mathcal{F}$ is given by a conditional probability distribution $P_{\mathcal{F}(U,V)|UV}$ with $\mathcal{F}(U,V)$ denoting the ideal-world execution, where the players forward their inputs U, V to $\mathcal{F}$ and output whatever they obtain from $\mathcal{F}$.

Definition 1 (Correctness). *A protocol $\Pi(U,V) = (X,Y)$ correctly implements an ideal classical functionality $\mathcal{F}$, if for every distribution of the input values U and V, the resulting common output (X,Y) satisfies $(U,V,X,Y) \overset{s}{\approx} (U,V,\mathcal{F}(U,V))$.*

[3] Note that we use a simplified joint output representation in comparison to [9].

We now define computational security against dishonest Alice, the definitions for dishonest Bob are analogue. Let Z and U' denote dishonest Alice's classical and quantum information. We consider a poly-size quantum circuit, called *input sampler*, which takes as input the security parameter and produces the input state $\rho_{U'ZV}$. We require from the input sampler that any $\rho_{U'ZV}$ is restricted to be of form $\rho_{U'\leftrightarrow Z\leftrightarrow V} = \sum_{z,v} P_{ZV}(z,v)|z\rangle\langle z| \otimes |v\rangle\langle v| \otimes \rho^z_{U'}$ (see [6] for notational details), where it holds that[4] $\rho^z_{U'} = \rho^{z,v}_{U'}$. This expresses *conditional independence*, namely that Bob's classical V is independent of Alice's quantum part U' when given Z. In other words, Alice's quantum part U' is correlated with Bob's part only via her classical Z.

Definition 2 (Computational security against dishonest Alice). *A protocol Π implements an ideal classical functionality $\mathcal{F}$ computationally securely against dishonest Alice, if for any real-world adversary $\mathsf{A}' \in \mathfrak{P}$, there exists an ideal-world adversary $\hat{\mathsf{A}}' \in \mathfrak{P}$ such that, for any efficient input sampler with $\rho_{U'ZV} = \rho_{U'\leftrightarrow Z\leftrightarrow V}$, it holds that the outputs are quantum-computationally indistinguishable, i.e., $out^{\Pi}_{\mathsf{A}',\mathsf{B}} \overset{q}{\approx} out^{\mathcal{F}}_{\hat{\mathsf{A}}',\hat{\mathsf{B}}}$.*

We state these output states explicitly as $out^{\Pi}_{\mathsf{A}',\mathsf{B}} = \rho_{UX'ZY}$ and $out^{\mathcal{F}}_{\hat{\mathsf{A}}',\hat{\mathsf{B}}} = \rho_{UX'\leftrightarrow Z\leftrightarrow Y}$, which shows that Alice's possibilities in the ideal world are limited: She can produce some classical input U for $\mathcal{F}$ from her quantum input state U', and then she can obtain a quantum state X' by locally processing U and possibly $\mathcal{F}$'s classical reply X.

3 Security Notions for Coin-Flipping

We denote a generic protocol with a λ-bit coin-string as output by $\Pi^{\lambda-\mathtt{COIN}}_{\mathsf{A},\mathsf{B}}$, corresponding to an ideal functionality $\mathcal{F}_{\lambda-\mathtt{COIN}}$. The outcome of such a protocol is $c \in \{0,1\}^\lambda \cup \{\perp\}$, i.e., either an λ-bit-string or an error message. We use several security parameters, indicating the length of coin-strings for different purposes; the length of a coin-flip yielding a key or a challenge are denoted by κ or σ, respectively. The ideal functionality for coin-flipping is defined symmetric such that always the respective dishonest party has an option to abort. We state the ideal functionalities in the case of both players being honest and in the case of dishonest Alice and honest Bob (Fig. 1). Note that the latter then also applies to honest Alice and dishonest Bob by simply switching sides and names.

Recall that the *joint output representation* of a protocol execution is denoted by $out^{\Pi}_{\mathsf{A},\mathsf{B}}$ (with $\Pi = \Pi^{\lambda-\mathtt{COIN}}_{\mathsf{A},\mathsf{B}}$) and given here for the case of honest players. The same notation with $\mathcal{F} = \mathcal{F}_{\lambda-\mathtt{COIN}}$ and $\hat{\mathsf{A}}, \hat{\mathsf{B}}$ applies in the ideal world as $out^{\mathcal{F}}_{\hat{\mathsf{A}},\hat{\mathsf{B}}}$, where the players invoke the ideal functionality $\mathcal{F}_{\lambda-\mathtt{COIN}}$ and output whatever

[4] ρ^x_E denotes a state in register E, depending on value $x \in \mathcal{X}$ of random variable X over $\mathcal{X}$ with distribution P_X. Then, from the view of an observer, who holds register E but does not know X, the system is in state $\rho_E = \sum_{x\in\mathcal{X}} P_X(x)\rho^x_E$, where ρ_E depends on X in the sense that E is in state ρ^x_E exactly if $X = x$.

FUNCTIONALITY $\mathcal{F}_{\lambda-\texttt{COIN}}$ WITH HONEST PLAYERS:
Upon receiving requests `start` from both Alice and Bob, $\mathcal{F}_{\lambda-\texttt{COIN}}$ outputs uniformly random $h \in_R \{0,1\}^\lambda$ to Alice and Bob.

FUNCTIONALITY $\mathcal{F}_{\lambda-\texttt{COIN}}$ WITH DISHONEST ALICE:

1. Upon receiving requests `start` from both Alice and Bob, $\mathcal{F}_{\lambda-\texttt{COIN}}$ outputs uniformly random $h \in_R \{0,1\}^\lambda$ to Alice.
2. It then waits to receive her second input $\top$ or $\bot$ and outputs h or $\bot$ to Bob, respectively.

Figure 1. The Ideal Functionality for λ-bit Coin-Flipping.

they obtain from it. We need an additional notation here, describing the *outcome* of a protocol run between e.g. honest A and B, namely $c \leftarrow \Pi_{\mathsf{A},\mathsf{B}}^{\lambda-\texttt{COIN}}$.

We will define three flavors of security for coin-flip protocols, namely *uncontrollable (uncont)*, *random* and *enforceable (force)*. The two sides can have different flavors. Then, if a protocol $\Pi_{\mathsf{A},\mathsf{B}}^{\lambda-\texttt{COIN}}$ is, for instance, enforceable against Alice and random against Bob, we write $\pi^{(\texttt{force},\texttt{random})}$, and similarly for the eight other combinations of security. Note that for simplicity of notation, we will then omit the indexed name as well as the length of the coin, as they are clear from the context. Again, we define all three flavors for Alice's side only, as the definitions for Bob are analogue. Recall that U' and Z resp. V denote dishonest Alice's quantum and classical input resp. honest Bob's classical input. As before, we assume a poly-size input sampler, which takes as input the security parameter, and produces a valid input state $\rho_{U'ZV} = \rho_{U' \leftrightarrow Z \leftrightarrow V}$. Note that an honest player's input is empty but models the invocation `start`. We stress that we require for all three security flavors and for all $c \in \{0,1\}^\lambda$ that

$$\Pr\left[c \leftarrow \Pi_{\mathsf{A},\mathsf{B}}^{\lambda-\texttt{COIN}}\right] = 2^{-\lambda}\,,$$

which implies that when both parties are honest, then the coin is unbiased. Below we only define the extra properties required for each of the three flavors.

We call a coin-flip *uncontrollable* against Alice, if she cannot force the coin to hit some negligible subset, except with negligible probability.

Definition 3 (Uncontrollability against dishonest Alice). *We say that protocol $\Pi_{\mathsf{A},\mathsf{B}}^{\lambda-\texttt{COIN}}$ implements an* uncontrollable *coin-flip against dishonest Alice, if it holds for any poly-sized adversary $\mathsf{A}' \in \mathfrak{P}$ with inputs as specified above and all negligible subsets $Q \subset \{0,1\}^\lambda$ that*

$$\Pr\left[c \leftarrow \Pi_{\mathsf{A}',\mathsf{B}}^{\lambda-\texttt{COIN}} \,:\, c \in Q\right] \in \mathit{negl}\left(\kappa\right).$$

Note that we denote by $Q \subset \{0,1\}^\lambda$ a family of subsets $\{Q(\kappa) \subset \{0,1\}^{\lambda(\kappa)}\}_{\kappa\in\mathbb{N}}$ for security parameter κ. Then we call Q negligible, if $|Q(\kappa)|2^{-\lambda(\kappa)}$ is negligible in κ. In other words, we call a subset negligible, if it contains a negligible fraction of the elements in the set in which it lives.

We call a coin-flip *random* against Alice, if she cannot enforce a non-uniformly random output string in $\{0,1\}^\lambda$, except by making the protocol fail on some chosen runs. That means she can at most lower the probability of certain output strings compared to the uniform case.

Definition 4 (Randomness against dishonest Alice). *We say that protocol* $\Pi_{\mathsf{A},\mathsf{B}}^{\lambda-\mathtt{COIN}}$ *implements a* random *coin-flip against dishonest Alice, if it holds for any poly-sized adversary* $\mathsf{A}' \in \mathfrak{P}$ *with inputs as specified above that there exists an event* E *such that* $\Pr[E] \in negl(\kappa)$ *and for all* $x \in \{0,1\}^\lambda$ *it holds that*

$$\Pr\left[c \leftarrow \Pi_{\mathsf{A}',\mathsf{B}}^{\lambda-\mathtt{COIN}} : c = x \,|\, \bar{E}\right] \leq 2^{-\lambda}.$$

It is obvious that if a coin-flip is random against Alice, then it is also an uncontrollable coin-flip against her. We will later discuss a generic transformation going in the other direction from uncontrollable to random coin-flipping.

We call a coin-flip *enforceable* against Alice, if it is possible, given a uniformly random c, to simulate a run of the protocol hitting exactly the outcome c, though we still allow that the corrupted party forces abort on some outcomes.[5]

Definition 5 (Enforceability against dishonest Alice). *We call protocol* $\Pi_{\mathsf{A},\mathsf{B}}^{\lambda-\mathtt{COIN}}$ enforceable *against dishonest Alice, if it implements the ideal functionality* $\mathcal{F}_{\lambda-\mathtt{COIN}}$ *against her.*

That means that for any poly-sized adversary $\mathsf{A}' \in \mathfrak{P}$, there exists an ideal-world adversary $\hat{\mathsf{A}}' \in \mathfrak{P}$ that simulates the protocol with A' as follows. $\hat{\mathsf{A}}'$ requests output $h \in \{0,1\}^\lambda$ from $\mathcal{F}_{\lambda-\mathtt{COIN}}$. Then it simulates a run of the coin-flip protocol with A' and tries to enforce output h. If $\hat{\mathsf{A}}'$ succeeds, it inputs $\top$ as A''s second input to $\mathcal{F}_{\lambda-\mathtt{COIN}}$. In that case, $\mathcal{F}_{\lambda-\mathtt{COIN}}$ outputs h. Otherwise, $\hat{\mathsf{A}}'$ inputs $\bot$ to $\mathcal{F}_{\lambda-\mathtt{COIN}}$ as second input and $\mathcal{F}_{\lambda-\mathtt{COIN}}$ outputs $\bot$. In addition, the simulation is such that the ideal output is quantum-computationally indistinguishable from the output of an actual run of the protocol, i.e., $out_{\mathsf{A}',\mathsf{B}}^{\Pi} \overset{q}{\approx} out_{\hat{\mathsf{A}}',\hat{\mathsf{B}}}^{\mathcal{F}}$, where $\Pi = \Pi_{\mathsf{A}',\mathsf{B}}^{\lambda-\mathtt{COIN}}$ and $\mathcal{F} = \mathcal{F}_{\lambda-\mathtt{COIN}}$. Enforceability against dishonest Bob is analogously defined. Corollary 1 follows.

Corollary 1. *If* $\Pi_{\mathsf{A},\mathsf{B}}^{\lambda-\mathtt{COIN}} \in \pi^{(force,force)}$*, i.e., it is enforceable against both dishonest Alice and dishonest Bob, then* $\Pi_{\mathsf{A},\mathsf{B}}^{\lambda-\mathtt{COIN}}$ *is a secure implementation of* $\mathcal{F}_{\lambda-\mathtt{COIN}}$*, according to Definition 2.*

4 Mixed Commitments

We use mixed commitment schemes throughout our constructions—they will indeed be our only computational assumption. Mixed commitment are unconditionally hiding for some public keys and unconditionally binding for others.

[5] Note that an enforceable coin-flip is not necessarily a random coin-flip, as it is allowed that the outcome of an enforceable coin-flip is only quantum-computationally indistinguishable from uniformly random, whereas a random coin-flip is required to produce truly random outcomes on the non-aborting runs.

In the following, we introduce mixed commitments, denoted by $\texttt{commit}_{pk}$, more formally. We also describe a construction of an interactive commitment protocol $\texttt{COMMIT}_{pk}$ with mixed-commitment-scheme-like properties. The reason for presenting the protocol here is to simplify the description of the later protocol in which it is used as a subprotocol.

4.1 Mixed Commitment Schemes

Mixed commitment schemes consists of four poly-time algorithms $\mathcal{G}_{\mathtt{H}}$, $\mathcal{G}_{\mathtt{B}}$, $\texttt{commit}$, and $\texttt{xtr}$. The *unconditionally hiding key generator* $\mathcal{G}_{\mathtt{H}}$ outputs public keys $pk \in \{0,1\}^{\kappa}$.[6] The *unconditionally binding key generator* $\mathcal{G}_{\mathtt{B}}$ outputs key pairs (pk, sk), where $pk \in \{0,1\}^{\kappa}$ and where sk is the secret key. The commitment algorithm takes as input a message m, a randomizer r and a public key pk and outputs a commitment $C = \texttt{commit}_{pk}(m, r)$. The extraction algorithm $\texttt{xtr}$ takes as input a commitment C and a secret key sk and outputs a message m', meant to be the message committed by C. We require the following properties:

Unconditionally hiding: For keys pk generated by $\mathcal{G}_{\mathtt{H}}$ it holds that $\texttt{commit}_{pk}$ is statistically hiding, i.e. $(pk, \texttt{commit}_{pk}(m_1, r_1)) \stackrel{s}{\approx} (pk, \texttt{commit}_{pk}(m_2, r_2))$ for all m_1, m_2 when r_1 and r_2 are uniformly random and independent.
Extractability: It holds for all pairs (pk, sk) generated by $\mathcal{G}_{\mathtt{B}}$ and for all values m, r that $\texttt{xtr}_{sk}(\texttt{commit}_{pk}(m, r)) = m$.
Key indistinguishability: A random public key pk_1 generated by $\mathcal{G}_{\mathtt{B}}$ and a random public key pk_2 generated by $\mathcal{G}_{\mathtt{H}}$ are indistinguishable by poly-sized quantum circuits, i.e., $pk_1 \stackrel{q}{\approx} pk_2$.

We additionally require that random public keys generated by $\mathcal{G}_{\mathtt{H}}$ are statistically close to uniform in $\{0,1\}^{\kappa}$, i.e., almost all keys are unconditionally hiding.[7]

As a candidate for instantiating our definition we can, for instance, take the lattice-based public-key encryption scheme of Regev [16] in its multi-bit variant as given in the full version of [15]. Regev's cryptosystem is based on the hardness of the learning with error problem, which can be reduced from worst-case (quantum) hardness of the shortest vector problem (in its decision version). Thus, breaking the scheme implies an efficient algorithm for approximating the lattice problem in the worst-case, which is assumed to be hard even with quantum computing power. A regular public key for Regev's scheme is proven to be quantum-computationally indistinguishable from the case where a public key is chosen from the uniform distribution. In this case, the ciphertext carries essentially no information about the message [16, Lemma 5.4]. This proof of semantic

[6] For notational simplicity, the length of public keys is assumed to equal security parameter κ.

[7] The definition is a weakening of the original notion of mixed commitments from [8], in that we do not require that unconditionally hiding keys are equipped with an equivocation trapdoor. It is also a strengthening in that we require quantum indistinguishability of the two key flavors.

security for Regev's cryptosystem is in fact the property we require for our commitment.

4.2 The protocol $\texttt{COMMIT}_{pk}$

In one of our security amplifications of coin-flip protocols we will need a mixed commitment scheme which also provides *equivocability*, i.e., a simulator can open unconditionally hiding commitments to different values. We add equivocability using an interactive protocol $\texttt{COMMIT}_{pk}$. Instead of equipping unconditionally hiding keys with equivocation trapdoors, we will do it by letting the equivocation trapdoor be the ability of the simulator to force the outcome of a coin-flip protocol in the simulation. The reason for this change, as compared to [8], is that the notion of a mixed commitment scheme in [8] was developed for the CRS-model, where the simulator is free to pick the CRS and hence could pick it to be a unconditionally hiding public key with known equivocation trapdoor. Here we are interested in the bare (CRS devoid) model and hence have to add equivocation in a different manner. This is one of the essential steps in bootstrapping fully simulatable strong coin-flipping from weak coin-flipping.

The protocol $\texttt{COMMIT}_{pk}$ uses a secret sharing scheme $\texttt{sss}$, described now. Let σ be a secondary security parameter. Given message $m = (m_1, \ldots, m_\sigma) \in \mathbb{F}^\sigma$ and randomizer $s = (s_1, \ldots, s_\sigma) \in \mathbb{F}^\sigma$, let $f_{m,s}(\mathtt{X})$ denote the unique polynomial of degree $2\sigma - 1$, for which $f_{m,s}(-i+1) = m_i$ for $i = 1, \ldots, \sigma$ and $f_{m,s}(i) = s_i$ for $i = 1, \ldots, \sigma$. Furthermore, we "fill up" positions $i = \sigma + 1, \ldots, \Sigma$, where $\Sigma = 4\sigma$, by letting $s_i = f_{m,s}(i)$. The shares are now $s = (s_1, \ldots, s_\Sigma)$.

We stress two simple facts about $\texttt{sss}$. First, for any message $m \in \mathbb{F}^\sigma$ and any subset $S \subset \{1, \ldots, \Sigma\}$ of size $|S| = \sigma$, the shares $s|_S$ are uniformly random in $\mathbb{F}^\sigma$, when S is chosen uniformly at random in $\mathbb{F}^\sigma$ and independent of m. This aspect is trivial for $S = \{1, \ldots, \sigma\}$, as we defined it that way, and it extends to the other subsets using Lagrange interpolation. And second, if $m^1, m^2 \in \mathbb{F}^\sigma$ are two distinct messages, then $\texttt{sss}(m^1; s^1)$ and $\texttt{sss}(m^2; s^2)$ have Hamming distance at least $\Sigma - 2\sigma$. Again, this follows by Lagrange interpolation, since the polynomial $f_{m^1,s^1}(\mathtt{X})$ has degree at most $2\sigma - 1$, and hence, can be computed from any 2σ shares s_i using Lagrange interpolation. The same holds for $f_{m^2,s^2}(\mathtt{X})$. Thus, if 2σ shares are the same, then $f_{m^1,s^1}(\mathtt{X})$ and $f_{m^2,s^2}(\mathtt{X})$ are the same, which implies that the messages $m^1 = f_{m^1,s^1}(-\sigma+1), \ldots, f_{m^1,s^1}(0)$ and $m^2 = f_{m^2,s^2}(-\sigma+1), \ldots, f_{m^2,s^2}(0)$ are the same.

In addition to $\texttt{sss}$, the protocol $\texttt{COMMIT}_{pk}$ uses a mixed commitment scheme $\texttt{commit}_{pk}$. The key generators for $\texttt{COMMIT}_{pk}$ are the same as for $\texttt{commit}_{pk}$. Finally, $\texttt{COMMIT}_{pk}$ uses a coin-flip protocol $\pi^{(\texttt{random},\texttt{force})}$ which is random for the committer and which is enforceable against the receiver of the commitment. The details of $\texttt{COMMIT}_{pk}$ are given in Fig. 2.

We first show that when (pk, sk) is generated using $\mathcal{G}_{\mathtt{B}}$, then $\texttt{COMMIT}_{pk}$ is extractable. Given any commitment $M = \big(M_1, \ldots, M_\Sigma\big)$, we extract $\texttt{xtr}_{sk}(M) = \big(\texttt{xtr}_{sk}(M_1), \ldots, \texttt{xtr}_{sk}(M_\Sigma)\big) = (s_1, \ldots, s_\Sigma) = s$. Assume $s' = (s'_1, \ldots, s'_\Sigma)$ is the consistent sharing closest to s. That means that s' is the vector which is consistent with a polynomial $f_{m',s'}(\mathtt{X})$ of degree at most $2\sigma - 1$ and which at

Commitment Scheme $\texttt{COMMIT}_{\texttt{pk}}$:

Commitment Phase:

1. Let message $m \in \mathbb{F}^\sigma$ be the message. The committer samples uniformly random $s \in \mathbb{F}^\sigma$ and computes the shares $\texttt{sss}(m; s) = (s_1, \ldots, s_\Sigma)$, where $s_i \in \mathbb{F}$.
2. He computes $\texttt{COMMIT}_{pk}\big(m, (s, r)\big) = \big(M_1, \ldots, M_\Sigma\big)$, where $M_i = \texttt{commit}_{pk}(s_i, r_i)$ for randomness $r = (r_1, \ldots, r_\Sigma)$.
3. The committer sends $(M_1, \ldots, M_\Sigma)$.

Opening Phase:

1. The committer sends the shares $s = (s_1, \ldots, s_\Sigma)$ to the receiver.
2. If the shares are not consistent with a polynomial of degree at most $2\sigma - 1$, the receiver aborts.
3. The parties run $\pi^{(\texttt{random},\texttt{force})}$ to generate a uniformly random subset $S \subset \{1, \ldots, \Sigma\}$ of size $|S| = \sigma$.
4. The committer sends $r|_S$.
5. The receiver verifies that $M_i = \texttt{commit}_{pk}(s_i, r_i)$ for all $i \in S$. If the test fails, he aborts. Otherwise, he computes the message $m \in \mathbb{F}^\sigma$ consistent with s.

Figure 2. The Commitment Scheme $\texttt{COMMIT}_{pk}$.

the same time differs from s in the fewest positions. Note that we can find s' in poly-time when using a Reed Solomon code, which has efficient minimal distance decoding. We then interpolate the polynomial $f_{m',s'}(\texttt{X})$, let $m' = f_{m',s'}(-\sigma + 1), \ldots, f_{m',s'}(0)$, and let $\texttt{xtr}_{sk}(M) = m'$. Any other sharing $s'' = (s''_1, \ldots, s''_\Sigma)$ must have Hamming distance at least 2σ to s'. Now, since s is closer to s' than to any other consistent sharing, it must, in particular, be closer to s' then to s''. This implies that s is at distance at least σ to s''.

We will use this observation for proving soundness of the opening phase. To determine the soundness error, assume that $\texttt{COMMIT}_{pk}$ does not open to the shares s' consistent with s. As observed, this implies that $\big(\texttt{xtr}_{sk}(M_1), \ldots, \texttt{xtr}_{sk}(M_\Sigma)\big)$ has Hamming distance at least σ to s'. However, when $\texttt{commit}_{pk}$ is unconditionally binding, all M_i can only be opened to $\texttt{xtr}_{sk}(M_i)$. From the above two facts, we have that there are at least σ values $i \in \{1, \ldots, \Sigma\}$ such that the receiver cannot open M_i to s_i for $i \in S$. Since $\Sigma = 4\sigma$, these σ bad indices (bad for a dishonest sender) account for a fraction of $\frac{1}{4}$ of all points in $\{1, \ldots, \Sigma\}$. Thus, the probability that none of the σ points in S is a bad index is at most $(\frac{3}{4})^\sigma$, which is negligible. Setting $\sigma = \log_{\frac{4}{3}} 2$ gives a negligible error of $(\frac{1}{2})^\kappa$, where κ is the security parameter.

We then analyze the equivocability of $\texttt{COMMIT}_{pk}$. We will use the ability of the simulator for the committer to force the challenge S as the simulator's trapdoor. It will simply pick S uniformly at random before the simulation and prepare for this particular challenge. The details are given in Fig. 3. We omit an analysis here but refer to Section 5.2, where the construction will be further discussed.

SIMULATING $\texttt{COMMIT}_{pk}$ WITH TRAPDOOR S:

1. $\hat{\mathcal{S}}$ gets as input a uniformly random subset $S \subset \{1, \ldots, \Sigma\}$ of size σ and an initial message $m \in \mathbb{F}^\sigma$.
2. $\hat{\mathcal{S}}$ commits honestly to $m \in \mathbb{F}^\sigma$ by $M = \texttt{COMMIT}_{sk}\left(m, (s, r)\right)$, as specified in the commitment phase.
3. $\hat{\mathcal{S}}$ is given an alternative message $\tilde{m} \in \mathbb{F}^\sigma$, i.e., the aim is opening M to $\tilde{m}$.
4. $\hat{\mathcal{S}}$ lets $s|_S$ be the σ messages committed to by $M|_S$. Then it interpolates the unique polynomial $f_{\tilde{m},s}$ of degree at most $2\sigma - 1$ for which $f_{\tilde{m},s}(i) = s_i$ for $i \in S$ and for which $f_{\tilde{m},s}(-i+1) = \tilde{m}_i$ for $i = 1, \ldots, \sigma$. Note that this is possible, as we have exactly 2σ points which restrict our choice of $f_{\tilde{m},s}$. $\hat{\mathcal{S}}$ sends $s = \left(f_{\tilde{m},s}(1), \ldots, f_{\tilde{m},s}(\Sigma)\right)$ to the receiver.
5. The parties run $\pi^{(\texttt{random},\texttt{force})}$ and $\hat{\mathcal{S}}$ forces the outcome S.
6. For all $i \in S$, the sender opens M_i to $f_{\tilde{m},s}(i)$. This is possible, since $f_{\tilde{m},s}(i) = s_i$ is exactly the message committed to by M_i when $i \in S$.

Figure 3. The Ideal-World Simulation of $\texttt{COMMIT}_{pk}$.

5 Amplification Theorems for Strong Coin-Flipping

We now propose and prove theorems, which allow us to amplify the security strength of coins. Ultimately, we aim at constructing a strong coin-flip protocol $\pi^{(\texttt{force},\texttt{force})}$ with outcomes of any polynomial length ℓ in λ from a weaker coin-flip protocol $\pi^{(\texttt{force},\texttt{uncont})}$ of κ-bit-strings, where κ is the key length of the mixed commitment scheme. We do this in two steps. We first show how to implement $\pi^{(\texttt{force},\texttt{random})}$ for ℓ-bit-strings (for any polynomial ℓ) given $\pi^{(\texttt{force},\texttt{uncont})}$ for κ-bit-strings, and we then show how to implement $\pi^{(\texttt{force},\texttt{force})}$ for poly-long bit-strings given $\pi^{(\texttt{force},\texttt{random})}$ for poly-long bit-strings.

The ability to amplify $\pi^{(\texttt{force},\texttt{uncont})}$ for κ-bit-strings to $\pi^{(\texttt{force},\texttt{force})}$ for poly-bit-string is of course only interesting, if there exists such a candidate. We do not know of any protocol with flavor $(\texttt{force}, \texttt{uncont})$ but not $(\texttt{force}, \texttt{random})$. However, we consider it as a contribution in itself to find the weakest security notion for coin-flipping that allows to amplify to the final strong $(\texttt{force}, \texttt{force})$ notion using a constant-round reduction.

A candidate for $\pi^{(\texttt{force},\texttt{random})}$ with one-bit outcomes is the protocol in [7], which is—in terms of this context—enforceable against one side in poly-time and random on the other side, with empty event E according to Definition 4, and the randomness guarantee even withstanding an unbounded adversary.[8] The protocol was shown to be sequentially composable [7,14]. Repeating the protocol κ times in sequence gives a protocol $\pi^{(\texttt{force},\texttt{random})}$ for κ-bit-strings. Note that this, in particular, gives a protocol $\pi^{(\texttt{force},\texttt{uncont})}$ for κ-bit-strings.

[8] The protocol was described and proven as $\pi^{(\texttt{random},\texttt{force})}$, but due to the symmetric coin-flip definitions here, we can easily switch sides between A and B.

5.1 From (force, uncont) to (force, random)

Assume that we are given a protocol $\pi^{(\texttt{force},\texttt{uncont})}$, that only guarantees that Bob cannot force the coin to hit a negligible subset (except with negligible probability). We now amplify the security on Bob's side from *uncontrollable* to *random* and therewith obtain a protocol $\pi^{(\texttt{force},\texttt{random})}$, in which Bob cannot enforce a non-uniformly random output string, except by letting the protocol fail on some occasions. The stronger protocol $\pi^{(\texttt{force},\texttt{random})}$ is given in Fig. 4, where $\texttt{commit}_{pk}$ is the basic mixed commitment scheme as described in Section 4.1. Correctness of $\pi^{(\texttt{force},\texttt{random})}$ is obvious by inspection of the protocol.

PROTOCOL $\pi^{(\texttt{force},\texttt{random})}$:

1. A and B run $\pi^{(\texttt{force},\texttt{uncont})}$ to produce a public key $pk \in \{0,1\}^\kappa$.
2. A samples $a \in_R \{0,1\}^\ell$, commits to it with $A = \texttt{commit}_{pk}(a, r)$ and randomizer $r \in_R \{0,1\}^\ell$, and sends A to B.
3. B samples $b \in_R \{0,1\}^\ell$ and sends b to A.
4. A opens A towards B.
5. The outcome is $c = a \oplus b$.

Figure 4. Amplification from (force, uncont) to (force, random).

Theorem 1. *If $\pi^{(force,uncont)}$ is enforceable against Alice and uncontrollable against Bob, then protocol $\pi^{(force,random)}$ is enforceable against Alice and random for Bob.*

We sketch the basic ideas behind the proof, which can be found in greater detail in Appendix A. Enforceability against A follows by forcing pk to be a pk generated as $(pk, sk) \leftarrow \mathcal{G}_{\mathsf{B}}$. The simulator then uses sk to extract a from A and then sends the b which makes $a \oplus b$ hit the desired outcome. Randomness against B follows from the fact that only a negligible fraction of the keys $pk \in \{0,1\}^\kappa$ are not unconditionally hiding keys and the outcome of $\pi^{(\texttt{force},\texttt{uncont})}$ is uncontrollable for B.

5.2 From (force, random) to (force, force)

We now show how to obtain a coin-flip protocol, which is enforceable against both parties. Then, we can also claim by Corollary 1 that this protocol is a strong coin-flip protocol, poly-time simulatable on both sides for the natural ideal functionality $\mathcal{F}_{\ell-\texttt{COIN}}$. The protocol $\pi^{(\texttt{force},\texttt{force})}$ is described in Fig. 5 and uses the extended commitment construction $\texttt{COMMIT}_{pk}$ from Section 4.2. The protocol makes two calls to a subprotocol with random flavor on one side and enforceability on the other side, but where the sides are interchanged, i.e. $\pi^{(\texttt{force},\texttt{random})}$ and $\pi^{(\texttt{random},\texttt{force})}$, so we simply switch the players' roles. Again, correctness of the protocol can be trivially checked.

PROTOCOL $\pi^{(\texttt{force},\texttt{force})}$:

1. A and B run $\pi^{(\texttt{force},\texttt{random})}$ to produce a random public key $pk \in \{0,1\}^\kappa$.
2. A computes and sends commitments $\texttt{COMMIT}_{pk}\big(a,(s,r)\big) = (A_1,\ldots,A_\Sigma)$ to B. In more detail, A samples uniformly random $a, s \in \mathbb{F}^\sigma$. She then computes $\texttt{sss}(a;s) = (a_1,\ldots,a_\Sigma)$ and $A_i = \texttt{commit}_{pk}(a_i,r_i)$ for $i = 1,\ldots,\Sigma$.
3. B samples uniformly random $b \in \{0,1\}^\ell$ and sends b to A.
4. A sends secret shares $(a_1,\ldots,a_\Sigma)$ to B. If $(a_1,\ldots,a_\Sigma)$ is not consistent with a polynomial of degree at most $(2\sigma-1)$, B aborts.
5. A and B run $\pi^{(\texttt{random},\texttt{force})}$ to produce a challenge $S \subset \{1,\ldots,\Sigma\}$ of length $|S| = \sigma$.
6. A sends $r|_S$ to B.
7. B checks if $A_i = \texttt{commit}_{pk}(a_i,r_i)$ for all $i \in S$. If that is the case, B computes message $a \in \mathbb{F}^\sigma$ consistent with $(a_1,\ldots,a_\Sigma)$ and the outcome of the protocol is $c = a \oplus b$. Otherwise, B aborts and the outcome is $c = \perp$.

Figure 5. Amplification from $(\texttt{force},\texttt{random})$ to $(\texttt{force},\texttt{force})$.

Theorem 2. *If $\pi^{(force,random)}$ is enforceable against Alice and random against Bob, then protocol $\pi^{(force,force)}$ is enforceable against both Alice and Bob.*

We sketch the main ideas behind the proof, which can be found in greater detail in Appendix B. Enforceability against A follows by forcing pk to be a key pk generated as $(pk,sk) \leftarrow \mathcal{G}_{\mathsf{B}}$. The simulator then uses sk to extract a from $(A_1,\ldots,A_\Sigma)$. Then it sends the b that makes $a \oplus b$ hit the desired outcome. Enforceability against B follows by letting the simulator sample a uniformly random S and running $\texttt{COMMIT}_{pk}\big(a,(s,r)\big) = (A_1,\ldots,A_\Sigma)$ in the equivocal model with trapdoor S. Then the simulator waits for b and forces the outcome of $\pi^{(\texttt{random},\texttt{force})}$ to be S, which allows it to open $(A_1,\ldots,A_\Sigma)$ to the a that makes $a \oplus b$ hit the desired outcome.

6 Application: Zero-Knowledge Proof of Knowledge

The purpose of a zero-knowledge proof of knowledge [1,10] is to verify in classical poly-time in the length of the instance, whether the prover's private input w is a valid witness for the common instance x in relation $\mathcal{R}$, i.e. $(x,w) \in \mathcal{R}$. Here, we propose a quantum-secure construction of a zero-knowledge proof of knowledge based on witness encoding, which we define in the context of a simulation in the quantum world. The protocol is constant-round if the coin-flip protocol is constant-round.

6.1 Simulatable Witness Encodings of $\mathcal{NP}$

We first specify a simulatable encoding scheme for binary relation $\mathcal{R} \subset \{0,1\}^* \times \{0,1\}^*$, which consists of five classical poly-time algorithms $(E,D,S,J,\hat{E})$. Then, we define completeness, extractability and simulatability for such a scheme in terms of the requirements of our zero-knowledge proof of knowledge.

Let $E : \mathcal{R} \times \{0,1\}^m \to \{0,1\}^n$ denote an *encoder*, such that for each $(x, w) \in \mathcal{R}$, the n-bit output $e \leftarrow E(x, w, r')$ is a random encoding of w, with randomness $r' \in \{0,1\}^m$ and polynomials $m(|x|)$ and $n(|x|)$. The corresponding *decoder* $D : \{0,1\}^* \times \{0,1\}^n \to \{0,1\}^*$ takes as input an instance $x \in \{0,1\}^*$ and an encoding $e \in \{0,1\}^n$ and outputs $w \leftarrow D(x, e)$ with $w \in \{0,1\}^*$. Next, let S denote a *selector* with input $s \in \{0,1\}^\sigma$ (with polynomial $\sigma(|x|)$) specifying a challenge, and output $S(s)$ defining a poly-sized subset of $\{1, \ldots, n\}$ corresponding to challenge s. We will use $S(s)$ to select which bits of an encoding e to reveal to the verifier. For simplicity, we use e_s to denote the collection of bits $e|_{S(s)}$. We denote with J the *judgment* that checks a potential encoding e by inspecting only bits e_s. In more detail, J takes as input instance $x \in \{0,1\}^*$, challenge $s \in \{0,1\}^\sigma$ and the $|S(s)|$ bits e_s, and outputs a judgment $j \leftarrow J(x, s, e_s)$ with $j \in \{\texttt{abort}, \texttt{success}\}$. Finally, the *simulator* is called $\hat{E}$. It takes as input instance $x \in \{0,1\}^*$ and challenge $s \in \{0,1\}^\sigma$ and outputs a random collection of bits $t|_{S(s)} \leftarrow \hat{E}(x, s)$. Again for simplicity, we let $t_s = t|_{S(s)}$. Then, if this set has the same distribution as bits of an encoding e in positions $S(s)$, the bits needed for the judgment to check an encoding e can be simulated given just instance x (see Definition 8).

Definition 6 (Completeness). *If an encoding $e \leftarrow E(x, w, r)$ is generated correctly, then $\texttt{success} \leftarrow J(x, s, e_s)$ for all $s \in \{0,1\}^\sigma$.*

We will call an encoding e *admissible* for x, if there *exist* two distinct challenges $s, s' \in \{0,1\}^\sigma$ for which $\texttt{success} \leftarrow J(x, s, e_s)$ and $\texttt{success} \leftarrow J(x, s', e_{s'})$.

Definition 7 (Extractability). *If an encoding e is admissible for x, then $\big(x, D(x, e)\big) \in \mathcal{R}$.*

We stress that extractability is similarly defined to the special soundness property of a classical Σ-protocol, which allows to extract w from two accepting conversations with distinct challenges. Such a requirement would generally be inapplicable in the quantum setting, as the usual rewinding technique is problematic and in particular in the context here, we cannot measure two accepting conversations during rewinding in the quantum world. Therefore, we define the stronger requirement that if there *exist* two distinct answerable challenges for one encoding e, then w can be extracted given only e. This condition works nicely in the quantum world, since we can obtain e without rewinding, as we demonstrate below.

Definition 8 (Simulatability). *For all $(x, w) \in \mathcal{R}$ and all $s \in_R \{0,1\}^\sigma$, the distribution of $e \leftarrow E(x, w, r')$ restricted to positions $S(s)$ is identical to the distribution of $t_s \leftarrow \hat{E}(x, s)$.*

To construct a simulatable witness encoding one can, for instance, start from the commit-and-open protocol for circuit satisfiability in [3], where the bits of the randomized circuit committed to by the sender is easy to see as a simulatable encoding of a witness being a consistent evaluation of the circuit to output 1. The challenge in the protocol is one bit e and the prover replies by showing either the

bits corresponding to some positions $S'(0)$ or positions $S'(1)$. The details can be found in [3]. This gives us a simulatable witness encoding for any $\mathcal{NP}$-relation $\mathcal{R}$ with $\sigma = 1$, using a Karp reduction from $\mathcal{NP}$ to circuit simulatability. By repeating it σ times in parallel we get a simulatable witness encoding for any σ. For $i = 1, \ldots, \sigma$, compute an encoding e^i of w and let $e = (e^1, \ldots, e^\sigma)$. Then for $s \in \{0,1\}^\sigma$, let $S(s)$ specify that the bits $S'(s_i)$ should be shown in e^i and check these bits. Note, in particular, that if two distinct s and s' passes this judgment, then there exists i such that $s_i \neq s'_i$, so e^i passes the judgment for both $s_i = 0$ and $s_i = 1$, which by the properties of the protocol for circuit satisfiability allows to compute a witness w for x from e^i. One can find w from e simply by trying to decode each e^j for $j = 1, \ldots, \sigma$ and check if $(x, w_j) \in \mathcal{R}$.

6.2 The Protocol

We now construct a quantum-secure zero-knowledge proof of knowledge from prover A to verifier B. We are interested in the $\mathcal{NP}$-language $\mathcal{L}(\mathcal{R}) = \{x \in \{0,1\}^* \mid \exists\, w \text{ s.t. } (x,w) \in \mathcal{R}\}$, where A has input x and w, and both A and B receive positive or negative judgment of the validity of the proof as output. We assume in the following that on input $(x,w) \notin \mathcal{R}$, honest A aborts. Unlike zero-knowledge proofs, proofs of knowledge can be modeled by an ideal functionality, given as $\mathcal{F}_{\mathtt{ZKPK}(\mathcal{R})}$ in Fig. 6. $\mathcal{F}_{\mathtt{ZKPK}(\mathcal{R})}$ can be thought of as a channel which only allows to send messages in the language $\mathcal{L}(\mathcal{R})$. It models *zero-knowledge*, as it only leaks instance x and judgment j but not witness w. Furthermore, it models a *proof of knowledge*, since Alice has to know and input a valid witness w to obtain output $j = \mathtt{success}$.

FUNCTIONALITY $\mathcal{F}_{\mathtt{ZKPK}(\mathcal{R})}$:

1. On input (x,w) from Alice, $\mathcal{F}_{\mathtt{ZKPK}(\mathcal{R})}$ sets $j = \mathtt{success}$ if $(x,w) \in \mathcal{R}$. Otherwise, it sets $j = \mathtt{abort}$.
2. $\mathcal{F}_{\mathtt{ZKPK}(\mathcal{R})}$ outputs (x,j) to Bob.

Figure 6. The Ideal Functionality for a Zero-Knowledge Proof of Knowledge.

Protocol $\mathtt{ZKPK}(\mathcal{R})$ is describe in Fig. 7. It is based on our fully simulatable coin-flip protocol $\pi^{(\mathtt{force},\mathtt{force})}$, which we analyze here in the hybrid model by invoking the ideal functionality of sequential coin-flipping twice (but with different output lengths).[9] One call to the ideal functionality $\mathcal{F}_{\kappa-\mathtt{COIN}}$ with output length κ is required to instantiate a mixed bit commitment scheme $\mathtt{COMMIT}_{pk}$. The second call to the functionality $\mathcal{F}_{\sigma-\mathtt{COIN}}$ produces σ-bit challenges for a simulatable witness encoding scheme with $(E, D, S, J, \hat{E})$ as specified in the pre-

[9] Note that in the hybrid model, a simulator can enforce a particular outcome to hit also when invoking the ideal coin-flip functionality. We then use Definition 5 to replace the ideal functionality by the actual protocol $\pi^{(\mathtt{force},\mathtt{force})}$.

vious Section 6.1. The formal proof of Theorem 3 can be found in Appendix C. Corollary 2 follows immediately.

Theorem 3. *For any simulatable witness encoding scheme $(E, D, S, J, \hat{E})$, satisfying completeness, extractability, and simulatability according to Definitions 6 - 8, and for negligible knowledge error $2^{-\sigma}$, protocol $\texttt{ZKPK}(\mathcal{R})$ securely implements $\mathcal{F}_{\texttt{ZKPK}(\mathcal{R})}$.*

Corollary 2. *If there exist mixed commitment schemes, then we can construct a classical zero-knowledge proof of knowledge against any quantum adversary $\mathsf{P}' \in \mathfrak{P}$ without any set-up assumptions.*

PROTOCOL $\texttt{ZKPK}(\mathcal{R})$:

1. A and B invoke $\mathcal{F}_{\kappa-\texttt{COIN}}$ to get a commitment key $pk \in \{0,1\}^\kappa$.
2. A samples $e \leftarrow E(x, w, r')$ with randomness $r' \in \{0,1\}^m$ and commits position-wise to all e_i for $i = 1, \ldots, n$, by computing $E_i = \texttt{commit}_{pk}(e_i, r_i)$ with randomness $r \in \{0,1\}^n$. She sends x and all E_i to B.
3. A and B invoke $\mathcal{F}_{\sigma-\texttt{COIN}}$ to flip a challenge $s \in_R \{0,1\}^\sigma$.
4. A opens her commitments to all e_s.
5. If any opening is incorrect, B outputs `abort`. Otherwise, he outputs $j \leftarrow J(x, s, e_s)$.

Figure 7. Zero-Knowledge Proof of Knowledge.

7 Application: Two-Party Function Evaluation

Here, we first show that mixed commitments imply a passively secure oblivious transfer protocol. From such a protocol it is straightforward to construct a protocol for *any* classical poly-time function with security against passive quantum adversaries [13]. We then propose a quantum-secure implementation for evaluating any such function with security against active quantum adversaries.

7.1 Oblivious Transfer

In an oblivious transfer protocol (OT), the sender A sends two messages m_0 and m_1 to the selector B. B can choose which message to receive, i.e. m_c according to his choice bit c. B does not learn anything about the other message m_{1-c}, and A does not learn B's choice bit c (see Fig. 8). The protocol is correct, as B knows sk_c and $\texttt{xtr}_{sk_c}(C_c) = \texttt{xtr}_{sk_c}(\texttt{commit}_{pk_c}(m_c, r_c)) = m_c$. Furthermore, it hides the other message m_{1-c} as $\texttt{commit}_{pk_{1-c}}$ is unconditionally hiding for random pk_{1-c}, except with negligible probability. Last, the choice bit is hidden in the sense of quantum-computational indistinguishability between keys for the outer commitments, namely a key produced by $\mathcal{G}_\mathsf{B}$ and a random key by $\mathcal{G}_\mathsf{H}$.

PROTOCOL $\mathtt{OT}$:

1. B samples two keys pk_0 and pk_1 according to his choice bit c, i.e. he samples pk_c as $(pk_c, sk_c) \leftarrow \mathcal{G}_{\mathtt{B}}$ and pk_{1-c} as $p_{1-c} \leftarrow \mathcal{G}_{\mathtt{H}}$. He sends (pk_0, pk_1) to A.
2. A commits to her messages (m_0, m_1) by computing $C_0 = \mathtt{commit}_{pk_0}(m_0, r_0)$ and $C_1 = \mathtt{commit}_{pk_1}(m_1, r_1)$. She sends (C_0, C_1) to B.
3. B computes $\mathtt{xtr}_{sk_c}(C_c)$.

Figure 8. Oblivious Transfer based on Mixed Commitments.

7.2 The Protocol

Based on protocol $\mathtt{OT}$, we can construct a passively secure protocol for any classical poly-time function f. Let $\Pi^f_{\mathsf{A},\mathsf{B}}(x_1, r_1, x_2, r_2)$ denote such a protocol between parties A and B with inputs x_1 and x_2 and random strings r_1 and r_2, respectively. We show an implementation of the ideal functionality $\mathcal{F}^f_{\mathtt{SFE}}$ evaluating—with security against active quantum adversaries—any classical poly-time function f for which there exists a classical passively secure protocol as described above. Functionality $\mathcal{F}^f_{\mathtt{SFE}}$ is shown in Fig. 9.[10] The implementation $\Pi^{\mathtt{SFE}(f)}_{\mathsf{A},\mathsf{B}}$ of $\mathcal{F}^f_{\mathtt{SFE}}$ is shown in Fig. 10. Corollary 3 is proven in Appendix D.

Corollary 3. *If there exist mixed commitment schemes, then there exists a classical implementation of $\mathcal{F}^f_{\mathtt{SFE}}$ for all classical poly-time functions f secure, according to Definitions 1 and 2.*

FUNCTIONALITY $\mathcal{F}^f_{\mathtt{SFE}}$ WITH HONEST PLAYERS:
On input x_1 from Alice and x_2 from Bob, $\mathcal{F}^f_{\mathtt{SFE}}$ outputs $y = f(x_1, x_2)$ to Alice and Bob.

FUNCTIONALITY $\mathcal{F}^f_{\mathtt{SFE}}$ WITH DISHONEST ALICE:

1. On input x_1 from Alice and x_2 from Bob, $\mathcal{F}^f_{\mathtt{SFE}}$ outputs $y = f(x_1, x_2)$ to Alice.
2. It then waits to receive her second input $\top$ or $\bot$ and outputs y or $\bot$ to Bob, respectively.

Figure 9. The Ideal Functionality for Secure Function Evaluation.

[10] Note that y does not need to be kept secure against external observers and also allows the adversary to abort depending on the value of y. We stress that it is no restriction that we consider common outputs nor that we leak y to observers. If we want to compute function $g(x_1, x_2) = (y_1, y_2)$ where *only* A (B) learns y_1 (y_2), we evaluate the common output function $y = f((x_1, p_1), (x_2, p_2))$ as follows. Public y contains $y_1 \oplus p_1$ and $y_2 \oplus p_2$, where p_1 and p_2 are A's and B's uniformly random additional input of the same length as y_1 and y_2. Thus, the common outputs are one-time pad encrypted using pads known only to the party who is to learn the result.

PROTOCOL $\Pi^{\mathtt{SFE}(f)}_{\mathsf{A},\mathsf{B}}$:

1. A and B invoke $\mathcal{F}_{\kappa-\mathtt{COIN}}$ to get a commitment key $pk \in \{0,1\}^\kappa$.
2. A sends a random commitment $X_1 = \mathtt{commit}_{pk}(x_1, \hat{r}_1)$ and B sends a random commitment $X_2 = \mathtt{commit}_{pk}(x_2, \hat{r}_2)$. Both parties use $\mathcal{F}_{\mathtt{ZKPK}(\mathcal{R})}$ to give a zero-knowledge proof of knowledge that they know the plaintext x_i inside commitments X_i for $i = 1, 2$.
3. A sends random commitment $S_1 = \mathtt{commit}_{pk}(s_1, \hat{r}_1)$ for uniformly random s_1 of length $|s_1| = |r_1|$, where r_1 is the randomness she intends to use in $\Pi^f_{\mathsf{A},\mathsf{B}}$. Similarly, B sends random commitment $S_2 = \mathtt{commit}_{pk}(s_2, \hat{r}_2)$ for uniformly random s_2 of length $|s_2| = |r_2|$. Again, they use $\mathcal{F}_{\mathtt{ZKPK}(\mathcal{R})}$ to give a zero-knowledge proof of knowledge of s_i in S_i for $i = 1, 2$.
4. A and B invoke $\mathcal{F}_{\sigma-\mathtt{COIN}}$ twice to get uniformly random s'_1 and s'_2 with $|s'_i| = |s_i|$ for $i = 1, 2$.
5. A lets $r_1 = s_1 \oplus s'_1$ and B lets $r_2 = s_2 \oplus s'_2$.
6. A and B run $\Pi^f_{\mathsf{A},\mathsf{B}}(x_1, r_1, x_2, r_2)$, i.e. they run the passively secure protocol on inputs and randomness as defined in the previous steps.
7. Whenever A sends a message m in the execution of $\Pi^f_{\mathsf{A},\mathsf{B}}(x_1, r_1, x_2, r_2)$, she gives a zero-knowledge proof of knowledge of s_1 in S_1 and x_1 in X_1, such that if $\Pi^f_{\mathsf{A},\mathsf{B}}(x_1, r_1, x_2, r_2)$ is run on x_1, $r_1 = s_1 \oplus s'_1$, and B's messages sent to A so far, then A would indeed send m. This is an $\mathcal{NP}$-statement, so we can use $\mathcal{F}_{\mathtt{ZKPK}(\mathcal{R})}$ for this proof.
8. If $\Pi^f_{\mathsf{A},\mathsf{B}}(x_1, r_1, x_2, r_2)$ terminates with output y, both parties output y.

Figure 10. Procedure for Secure Function Evaluation

Acknowledgement

Lunemann acknowledges financial support for part of this work by Institut Mittag-Leffler, The Royal Swedish Academy of Sciences. Nielsen acknowledges support from the Danish National Research Foundation and the National Science Foundation of China (under the grant 61061130540) for the Sino-Danish Center for the Theory of Interactive Computation, within which part of this work was performed.

References

1. Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In *Advances in Cryptology—CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*, pages 390–420. Springer, 1992.
2. Manuel Blum. Coin flipping by telephone. In *Advances in Cryptology: A Report on CRYPTO '81*, pages 11–15. U.C. Santa Barbara, Dept. of Elec. and Computer Eng., ECE Report No 82-04, 1981.
3. Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *Journal of Compututer and System Sciences*, 37(2):156–189, 1988.
4. Ivan B. Damgård, Serge Fehr, Carolin Lunemann, Louis Salvail, and Christian Schaffner. Improving the security of quantum protocols via commit-and-open. In

Advances in Cryptology—CRYPTO '09, volume 5677 of *Lecture Notes in Computer Science*, pages 408–427. Springer, 2009. Full version available at `arXiv:0902.3918v4[quant-ph]`.

5. Ivan B. Damgård, Serge Fehr, and Louis Salvail. Zero-knowledge proofs and string commitments withstanding quantum attacks. In *Advances in Cryptology—CRYPTO '04*, volume 3152 of *Lecture Notes in Computer Science*, pages 254–272. Springer, 2004.
6. Ivan B. Damgård, Serge Fehr, Louis Salvail, and Christian Schaffner. Secure identification and QKD in the bounded-quantum-storage model. In *Advances in Cryptology—CRYPTO '07*, volume 4622 of *Lecture Notes in Computer Science*, pages 342–359. Springer, 2007.
7. Ivan B. Damgård and Carolin Lunemann. Quantum-secure coin-flipping and applications. In *Advances in Cryptology—ASIACRYPT '09*, volume 5912 of *Lecture Notes in Computer Science*, pages 52–69. Springer, 2009.
8. Ivan B. Damgård and Jesper B. Nielsen. Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor. In *Advances in Cryptology—CRYPTO '02*, volume 2442 of *Lecture Notes in Computer Science*, pages 581–596. Springer, 2002.
9. 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.
10. Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *17th Annual ACM Symposium on Theory of Computing (STOC)*, pages 291–304, 1985.
11. Jeroen van de Graaf. *Towards a formal definition of security for quantum protocols*. PhD thesis, Université de Montréal (Canada), 1997.
12. Sean Hallgren, Adam Smith, and Fang Song. Classical cryptographic protocols in a quantum world. Extended abstract available at `qip2011.quantumlah.org/scientificprogramme/abstract/183.pdf`, 2011.
13. Joe Kilian. Founding cryptography on oblivious transfer. *20th Annual ACM Symposium on Theory of Computing (STOC)*, pages 20–31, 1988.
14. Carolin Lunemann. *Cryptographic Protocols under Quantum Attacks*. PhD thesis, Aarhus University (Denmark), November 2010. `arXiv:1102.0885[quant-ph]`.
15. 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. Full version available at `eprint.iacr.org/2007/348.pdf`.
16. 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.
17. Adam Smith. Personal communication, 2009.
18. John Watrous. Zero-knowledge against quantum attacks. In *SIAM Journal on Computing*, volume 39.1, pages 25–58, 2009. Preliminary version in *38th Annual ACM Symposium on Theory of Computing (STOC)*, pages 296–305, 2006.

A Proof of Theorem 1 (Enforceability and Randomness)

Proof (*Enforceability against Alice*). In case of corrupted A', $\hat{\mathsf{A}}'$ samples $(pk, sk) \leftarrow \mathcal{G}_{\mathsf{B}}$ as input. It then requests a uniformly random value h from $\mathcal{F}_{\ell-\mathtt{COIN}}$. It runs $\pi^{(\mathtt{force},\mathtt{uncont})}$ with A', in which $\hat{\mathsf{A}}'$ enforces the outcome pk in the first step. When A' sends commitment A, $\hat{\mathsf{A}}'$ uses sk to decrypt A to learn the unique string a that A can be opened to. $\hat{\mathsf{A}}'$ computes $b = h \oplus a$ and sends b to A'. If A' opens commitment A correctly, then the result is $c = a \oplus b = a \oplus (h \oplus a) = h$ as desired. In case she does not open correctly, $\hat{\mathsf{A}}'$ aborts with result $\perp$. Otherwise, $\hat{\mathsf{A}}'$ outputs whatever A' outputs.

Since h is uniformly random and independent of A and a, it follows that $b = h \oplus a$ is uniformly random and independent of A, exactly as in the protocol. Therefore, the transcript of the simulation has the same distribution as the real protocol, except that pk is uniform in $\mathcal{X}$ and not in $\{0,1\}^\kappa$. This is, however, quantum-computationally indistinguishable, as otherwise, A' could distinguish random access to samples from $\mathcal{X}$ from random access to samples from $\{0,1\}^\kappa$. The formal proof proceeds through a series of hybrids as described in full detail in the proof for Theorem 2 in Appendix B.

The above two facts, that first we hit h when we do not abort, and second that the transcript of the simulation is quantum-computationally indistinguishable from the real protocol, show that the resulting protocol is enforceable against Alice and simulatable on Alice's side for functionality $\mathcal{F}_{\ell-\mathtt{COIN}}$, according to Definition 5 combined with Theorem 5. ■

Proof (*Randomness against Bob*). For any B', pk is uncontrollable, i.e. $pk \in \{0,1\}^\kappa \setminus \mathcal{X}$, except with negligible probability, as $\mathcal{X}$ is negligible in $\{0,1\}^\kappa$. This, in particular, means that the commitment A is perfectly hiding the value a. Therefore, a is uniformly random and independent of b, and thus, $h = a \oplus b$ is uniformly random. This proves that the resulting coin-flip is random against Bob, according to Definition 4. ■

B Proof of Theorem 2 (Enforceability)

Proof (*Enforceability against Alice*). If A' is corrupted, $\hat{\mathsf{A}}'$ samples $(pk, sk) \leftarrow \mathcal{G}_{\mathsf{B}}$ as input and enforces $\pi^{(\mathtt{force},\mathtt{random})}$ in the first step to hit the outcome pk. It then requests value h from $\mathcal{F}_{\ell-\mathtt{COIN}}$. When A' sends commitments $(A_1, \ldots, A_\Sigma)$, $\hat{\mathsf{A}}'$ uses sk to extract a' with $\left(a'_1, \ldots, a'_\Sigma\right) = \left(\mathtt{xtr}_{sk}(A_1), \ldots, \mathtt{xtr}_{sk}(A_\Sigma)\right)$. $\hat{\mathsf{A}}'$ then sets $b = h \oplus a'$, and sends b to A'. Then $\hat{\mathsf{A}}'$ finishes the protocol honestly. In the following, we will prove that the transcript is quantum-computationally indistinguishable from the real protocol and that if $c \neq \perp$, then $c = h$, except with negligible probability.

First, we show indistinguishability. The proof proceeds via a hybrid argument.[11] Let $\mathcal{D}^0$ denote the distribution of the output of the simulation as described. We now change the simulation such that, instead of sending $b = h \oplus a'$, we simply choose a uniformly random $b \in \{0,1\}^\ell$ and then output the corresponding $h = a' \oplus b$. Let $\mathcal{D}^1$ denote the distribution of the output of the simulation after this change. Since h is uniformly random and independent of a' in the first case, it follows that then $b = h \oplus a'$ is uniformly random. Therefore, the change to choose a uniformly random b in the second case actually does not change the distribution at all, and it follows that $\mathcal{D}^0 = \mathcal{D}^1$.

By sending a uniformly random b, we are in a situation where we do not need the decryption key sk to produce $\mathcal{D}^1$, as we no longer need to know a'. So we can now make the further change that, instead of forcing $\pi^{(\texttt{force},\texttt{random})}$ to produce a random public key $pk \in \mathcal{X}$, we force it to hit a random public key $pk \in \{0,1\}^\kappa$. This produces a distribution $\mathcal{D}^2$ of the output of the simulation. Since $\mathcal{D}^1$ and $\mathcal{D}^2$ only differ in the key we enforce $\pi^{(\texttt{force},\texttt{random})}$ to hit and the simulation is quantum poly-time, there exists a poly-sized circuit Q, such that $Q(\mathcal{U}(\mathcal{X})) = \mathcal{D}^1$ and $Q(\mathcal{U}(\{0,1\}^\kappa)) = \mathcal{D}^2$, where $\mathcal{U}(\mathcal{X})$ and $\mathcal{U}(\{0,1\}^\kappa)$ denote the uniform distribution on $\mathcal{X}$ and the uniform distribution on $\{0,1\}^\kappa$, respectively. As $\mathcal{U}(\mathcal{X})$ and $\mathcal{U}(\{0,1\}^\kappa)$ are quantum-computationally indistinguishable, and Q is poly-sized, it follows that $Q(\mathcal{U}(\mathcal{X}))$ and $Q(\mathcal{U}(\{0,1\}^\kappa))$ are quantum-computationally indistinguishable, and therewith, $\mathcal{D}^1 \overset{q}{\approx} \mathcal{D}^2$.

A last change to the simulation is applied by running $\pi^{(\texttt{force},\texttt{random})}$ honestly instead of enforcing a uniformly random $pk \in \{0,1\}^\kappa$. Let $\mathcal{D}^3$ denote the distribution obtained after this change. As given in Definition 5, real runs of $\pi^{(\texttt{force},\texttt{random})}$ and runs enforcing a uniformly random value are quantum-computationally indistinguishable. Using a similar argument as above, where Q is the part of the protocol following the run of $\pi^{(\texttt{force},\texttt{random})}$, we get that $\mathcal{D}^2 \overset{q}{\approx} \mathcal{D}^3$. Finally by transitivity, it follows that $\mathcal{D}^0 \overset{q}{\approx} \mathcal{D}^3$. The observation that $\mathcal{D}^0$ is the distribution of the simulation and $\mathcal{D}^3$ is the actual distribution of the real protocol concludes the first part of the proof.

We now argue the second part, i.e., if $c \neq \perp$, then $c = h$, except with negligible probability. This follows from extractability of the commitment scheme $\texttt{COMMIT}_{pk}$. Recall that, if $pk \in \mathcal{X}$, then the probability that A' can open any A to a plaintext different from $\texttt{xtr}_{sk}(A)$ is at most $(\frac{3}{4})^\sigma$ when S is picked uniformly at random and independent of A. The requirement on S is however guaranteed (except with negligible probability) by the `random` flavor of the underlying protocol $\pi^{(\texttt{random},\texttt{force})}$ producing S. This concludes the proof of enforceability against Alice, as given in Definition 5. ■

[11] Briefly, a hybrid argument is a proof technique to show that two (extreme) distributions are computationally indistinguishable via proceeding through several (adjacent) hybrid distributions. If all adjacent distributions are pairwise computationally indistinguishability, it follows by transitivity that the two end points are so as well. We want to point out that we are not subject to any restrictions in how to obtain the hybrid distributions as long as we maintain indistinguishability.

Proof (*Enforceability against Bob*). To prove enforceability against corrupted B', we construct a simulator $\hat{\mathsf{B}}'$ as shown in Fig. 11. It is straightforward to verify that the simulation always ensures that $c = h$, if B' does not abort. However, we must explicitly argue that the simulation is quantum-computationally indistinguishable from the real protocol.

SIMULATION $\hat{\mathsf{B}}'$ for $\pi^{(\texttt{force},\texttt{force})}$:

1. $\hat{\mathsf{B}}'$ requests h from $\mathcal{F}_{\ell-\texttt{COIN}}$ and runs $\pi^{(\texttt{force},\texttt{random})}$ honestly with B' to produce a uniformly random public key $pk \in \{0,1\}^\kappa$.
2. $\hat{\mathsf{B}}'$ computes $\texttt{COMMIT}_{pk}\left(a', (s,r)\right) = (A_1, \ldots, \mathsf{A}_\Sigma)$ for uniformly random $a', s \in \mathbb{F}^\sigma$ and sends $(A_1, \ldots, A_\Sigma)$ to B'.
3. $\hat{\mathsf{B}}'$ receives b from B'.
4. $\hat{\mathsf{B}}'$ computes $a = b \oplus h$. It then picks a uniformly random subset $S \subset \{1, \ldots, \Sigma\}$ with $|S| = \sigma$, and lets $a'|_S$ be the σ messages committed to by $A|_S$. Then, it interpolates the unique polynomial f of degree at most $(2\sigma - 1)$ for which $f(i) = a'_i$ for $i \in S$ and for which $f(-i+1) = a_i$ for $i \in \{1, \ldots, \Sigma\} \setminus S$. Finally, it sends $(f(1), \ldots, f(\Sigma))$ to B'.
5. During the run of $\pi^{(\texttt{random},\texttt{force})}$, $\hat{\mathsf{B}}'$ enforces the challenge S.
6. $\hat{\mathsf{B}}'$ sends $r|_S$ to B'.
7. $\hat{\mathsf{B}}'$ outputs whatever B' outputs.

Figure 11. Simulation for Bob's `force` in $\pi^{(\texttt{force},\texttt{force})}$.

Indistinguishability follows by first arguing that the probability for $pk \notin \{0,1\}^\kappa \setminus \mathcal{X}$ is negligible. This follows from $\mathcal{X}$ being negligible in $\{0,1\}^\kappa$ and pk produced with flavor `random` against B' by $\pi^{(\texttt{force},\texttt{random})}$ being uniformly random in $\{0,1\}^\kappa$, except with negligible probability.

Second, we have to show that if $pk \in \{0,1\}^\kappa \setminus \mathcal{X}$, then the simulation is quantum-computationally close to the real protocol. This can be shown via the following hybrid argument. Let $\mathcal{D}^0$ be the distribution of the output of the simulation and let $\mathcal{D}^1$ be the distribution of the output of the simulation where we send all a'_i for all $i = \{1, \ldots, \Sigma\}$ at the end of Step (4.). Since commitments by $\texttt{commit}_{pk}$ are unconditionally hiding in case of $pk \in \{0,1\}^\kappa \setminus \mathcal{X}$, commitments by $\texttt{COMMIT}_{pk}$ are unconditionally hiding as well. Furthermore, both a' and a are uniformly random, so we obtain statistical closeness between $(a', \texttt{COMMIT}_{pk}\left(a', (s,r)\right))$ and $(a, \texttt{COMMIT}_{pk}\left(a', (s,r)\right))$. Note further that distributions $\mathcal{D}^0$ and $\mathcal{D}^1$ can be produced by a poly-sized circuit applied to either $(a', \texttt{COMMIT}_{pk}\left(a', (s,r)\right))$ or $(a, \texttt{COMMIT}_{pk}\left(a', (s,r)\right)$, it holds that $\mathcal{D}^0 \overset{q}{\approx} \mathcal{D}^1$.

Now, let $\mathcal{D}^2$ be the distribution obtained by not simulating the opening via the trapdoor, but instead doing it honestly to the value committed to, i.e. (a', r). We still use the challenge S from the forced run of $\pi^{(\texttt{random},\texttt{force})}$ though. However, for uniformly random challenges, real runs are quantum-computationally indistinguishable from simulated runs, and we get $\mathcal{D}^1 \overset{q}{\approx} \mathcal{D}^2$.

Next, let $\mathcal{D}^3$ be the distribution of the output of the simulation where we run $\pi^{(\texttt{random},\texttt{force})}$ honestly instead of enforcing outcome S. We then use the honestly

produced S' in the proof in Step (6.) instead of the enforced S. We can do this, as we modified the process leading to $\mathcal{D}^2$ towards an honest opening without any trapdoor, so we no longer need to enforce a particular challenge. Under the assumption that $\pi^{(\texttt{random},\texttt{force})}$ is enforceable against B', and observing that real runs are quantum-computationally indistinguishable from runs enforcing uniformly random outcomes, we obtain $\mathcal{D}^2 \overset{q}{\approx} \mathcal{D}^3$.

It follows by transitivity $\mathcal{D}^0 \overset{q}{\approx} \mathcal{D}^3$, and we conclude the proof by observing that after our changes, the process producing $\mathcal{D}^3$ is the real protocol. This concludes the proof of enforceability against Bob, according to Definition 5 with switched sides. ■

C Proof of Theorem 3 (Zero-Knowledge Proof of Knowledge)

Completeness is obvious. A honest party A, following the protocol with $(x, w) \in \mathcal{R}$ and any valid encoding e, will be able to open all commitments in the positions specified by any challenge s. Honest Bob then outputs $J(x, s, e_s) = \texttt{success}$.

Proof (*Security against dishonest Alice*). To prove security in case of corrupted A', we construct a simulator $\hat{\mathsf{A}}'$ that simulates a run of the actual protocol with A' and $\mathcal{F}_{\texttt{ZKPK}(\mathcal{R})}$. The proof is then twofold. First, we show indistinguishability between the distributions of simulation and protocol. And second, we verify that the extractability property of the underlying witness encoding scheme (see Definition 7) implies a negligible knowledge error. Note that if A' sends `abort` at any point during the protocol, $\hat{\mathsf{A}}'$ sends some input $(x', w') \notin \mathcal{R}$ to $\mathcal{F}_{\texttt{ZKPK}(\mathcal{R})}$ to obtain output (x, j) with $j = \texttt{abort}$, and the simulation halts. Otherwise, the simulation proceeds as shown in Fig. 12.

SIMULATION $\hat{\mathsf{A}}'$ FOR $\texttt{ZKPK}(\mathcal{R})$:

1. $\hat{\mathsf{A}}'$ samples a random key pk along with the extraction key sk. Then it enforces pk as output from $\mathcal{F}_{\kappa-\texttt{COIN}}$
2. When $\hat{\mathsf{A}}'$ receives x and $(E_1, \ldots, E_n)$ from A', it extracts $e = (\texttt{xtr}_{sk}(E_1), \ldots, \texttt{xtr}_{sk}(E_n))$.
3. $\hat{\mathsf{A}}'$ completes the simulation by following the protocol honestly. If any opening of A' is incorrect, $\hat{\mathsf{A}}'$ aborts. Otherwise, $\hat{\mathsf{A}}'$ inputs $\big(x, D(x, e)\big)$ to $\mathcal{F}_{\texttt{ZKPK}(\mathcal{R})}$ and receives (x, j) back. $\hat{\mathsf{A}}'$ outputs the final state of A' as output in the simulation.

Figure 12. Simulation against dishonest Alice.

Note that the only difference between the real protocol and the simulation is that $\hat{\mathsf{A}}'$ uses a random public key pk sampled along with an extraction key sk, instead of a uniformly random $pk \in \{0,1\}^\kappa$. It then enforces $\mathcal{F}_{\kappa-\texttt{COIN}}$ to hit pk.

However, by assumption on the commitment keys and by the properties of the ideal coin-flip functionality, the transcripts of simulation and protocol remain quantum-computationally indistinguishable under these changes.

Next, we analyze the output in more detail. It is clear that whenever honest B would output `abort` in the actual protocol, also $\hat{\mathsf{A}}'$ aborts, namely, if A' does deviate in the last steps of protocol and simulation, respectively. Furthermore, $\hat{\mathsf{A}}'$ accepts if and only if $(x, D(x, e)) \in \mathcal{R}$ or in other words, the judgment of the functionality is positive, denoted by $j_{\mathcal{F}} = \texttt{success}$.

It is therefore only left to prove that the case of $j_{\mathcal{F}} = \texttt{abort}$ but $j_J = \texttt{success}$ is negligible, where the later denotes the judgment of algorithm $J(x, s, e_s)$ as in the protocol. In that case, we have $(x, D(x, e)) \notin \mathcal{R}$. This means that w is not extractable from $D(x, e)$, which in turn implies that $(\texttt{xtr}_{sk}(E_1), \ldots, \texttt{xtr}_{sk}(E_n)) = e$ is not admissible. Thus, there are no two distinct challenges s and s', in which A' could correctly open her commitment to e. It follows by contradiction that there exists at most one challenge s which A' can answer. We produce $s \in \{0,1\}^\sigma$ uniformly at random, from which we obtain an acceptance probability of at most $2^{-\sigma}$. Thus, we conclude the proof with negligible knowledge error, as desired. ■

Proof (*Security against dishonest Bob*). To prove security in case of corrupted B', we construct simulator $\hat{\mathsf{B}}'$ as shown in Fig. 13. Our aim is to verify that this simulation is quantum-computationally indistinguishable from the real protocol. The key aspect will be the simulatability guarantee of the underlying witness encoding scheme, according to Definition 8.

SIMULATION $\hat{\mathsf{B}}'$ FOR $\texttt{ZKPK}(\mathcal{R})$:

1. $\hat{\mathsf{B}}'$ invokes $\mathcal{F}_{\kappa-\texttt{COIN}}$ to receive a uniformly random pk.
2. $\hat{\mathsf{B}}'$ samples a uniformly random challenge $s \in \{0,1\}^\sigma$ and computes $t_s \leftarrow \hat{E}(x, s)$. $\hat{\mathsf{B}}'$ then computes commitments E_i as follows: For all $i \in S(s)$, it commits to the previously sampled t_s via $E_i = \texttt{COMMIT}_{pk}\left(t_i, r_i\right)$. For all other positions $i \in \bar{S}$ (where $\bar{S} = \{1, \ldots, n\} \setminus S(s)$), it commits to randomly chosen values $t'_i \in_R \{0,1\}$, i.e. $E_i = \texttt{COMMIT}_{pk}\left(t'_i, r_i\right)$. It sends x and all E_i to B'.
3. $\hat{\mathsf{B}}'$ forces $\mathcal{F}_{\sigma-\texttt{COIN}}$ to hit s.
4. $\hat{\mathsf{B}}'$ opens E_i to t_i for all $i \in S(s)$, i.e. to all t_s.
5. $\hat{\mathsf{B}}'$ outputs whatever B' outputs.

Figure 13. Simulation against dishonest Bob.

The proof proceeds via a hybrid argument. Let $\mathcal{D}^0$ be the distribution of the simulation as described in Fig. 13. Let $\mathcal{D}^1$ be the distribution obtained from the simulation but with the following change: We inspect $\mathcal{F}_{\texttt{ZKPK}(\mathcal{R})}$ to get a valid witness w for instance x, and let $e \leftarrow E(x, w, r')$ be the corresponding encoding. Note that this is possible as a thought experiment for any adjacent distribution in a hybrid argument. From e we then use bits e_s for the same $S(s)$ as previously, instead of bits t_s sampled by $\hat{E}(x, s)$. All other steps are simulated as before. By the simulatability of the encoding scheme (Definition 8), it holds that the

bits t_s in $\mathcal{D}^0$ and the bits e_s in $\mathcal{D}^1$ have the same distribution. Thus, we obtain $\mathcal{D}^0 = \mathcal{D}^1$.

We further change the simulation in that we compute the bits in all positions $i \in \bar{S}$ by e_i of the encoding e defined in the previous step. Again, all other steps of the simulation remain unchanged. Let $\mathcal{D}^2$ denote the new distribution. The only difference now is that for $i \in \bar{S}$, the commitments E_i are to the bits e_i of a valid e and not to uniformly random bits t'_i. This, however, is quantum-computationally indistinguishable to B' for $pk \in_R \{0,1\}^\kappa$, as $\mathtt{COMMIT}$ is quantum-computationally hiding towards B'. Note that pk is guaranteed to be random by an honest call to $\mathcal{F}_{\kappa-\mathtt{COIN}}$ and recall that we do not have to open the commitments in these positions. Hence, we get that $\mathcal{D}^1 \overset{q}{\approx} \mathcal{D}^2$.

Note that after the two changes, leading to distributions $\mathcal{D}^1$ and $\mathcal{D}^2$, the commitment step and its opening now proceed as in the actual protocol, namely, we commit to the bits of $e \leftarrow E(x, e, r')$ and open the subset corresponding to $S(s)$. The remaining difference to the real protocol is the enforcement of challenge s, whereas s is chosen randomly in the protocol. Now, let $\mathcal{D}^3$ be the distribution of the modified simulation, in which we implement this additional change of invoking $\mathcal{F}_{\sigma-\mathtt{COIN}}$ honestly and then open honestly to the resulting s. Note that both processes, i.e., first choosing a random s and then enforcing it from $\mathcal{F}_{\sigma-\mathtt{COIN}}$, or invoking $\mathcal{F}_{\sigma-\mathtt{COIN}}$ honestly and receiving a random s, result in a uniformly random distribution on the output of $\mathcal{F}_{\sigma-\mathtt{COIN}}$. Thus, we obtain $\mathcal{D}^2 = \mathcal{D}^3$.

By transitivity, we conclude that $\mathcal{D}^0 \overset{q}{\approx} \mathcal{D}^3$, and therewith, that the simulation is quantum-computationally indistinguishable from the actual protocol. ■

D Proof of Corollary 3 (Two-Party Function Evaluation)

Proof (*Security against dishonest Alice*). If A' is corrupted, $\hat{\mathsf{A}}'$ uses the proof of knowledge to learn her x_1 inside commitment X_1. Then $\hat{\mathsf{A}}'$ inputs x_1 to $\mathcal{F}^f_{\mathtt{SFE}}$ as A''s input and receives $y = f(x_1, x_2)$. Now, $\hat{\mathsf{A}}'$ invokes $\mathcal{S}^f_{\hat{\mathsf{A}}',\hat{\mathsf{B}}}$ with input x_1 and y. This, in particular, yields randomness r_1 and is quantum-computationally indistinguishable from a real run of protocol $\Pi^f_{\mathsf{A}',\mathsf{B}}$. Furthermore, the simulated transcript contains all messages sent by $\hat{\mathsf{B}}$. Next, $\hat{\mathsf{A}}'$ uses the proof of knowledge to learn A''s s_1 inside commitment S_1. Then $\hat{\mathsf{A}}'$ enforces challenge s'_1 such that $s'_1 = s_1 \oplus r_1$, and thereby forces A' to use r_1 in the following.

$\hat{\mathsf{A}}'$ now runs $\Pi^f_{\mathsf{A}',\mathsf{B}}$ with A'. Whenever it is the turn of $\hat{\mathsf{B}}$ to send a message, $\hat{\mathsf{A}}'$ sends the next message obtained already by $\mathcal{S}^f_{\hat{\mathsf{A}}',\hat{\mathsf{B}}}$. Whenever it is the turn of A' to send a message m, $\hat{\mathsf{A}}'$ checks whether it coincides with the message obtained already by $\mathcal{S}^f_{\hat{\mathsf{A}}',\hat{\mathsf{B}}}$. Note that by construction her only consistent message really is the message obtained previously. In case of inconsistency, A' will fail in her following proof of knowledge, where she must prove that m is consistent with x_1 in X_1, s_1 in S_1, and where $r_1 = s_1 \oplus s'_1$ with r_1 obtained from $\mathcal{S}^f_{\hat{\mathsf{A}}',\hat{\mathsf{B}}}$. Hence, if

A' does not send an inconsistent m and thereby make the protocol fail, then the transcript of this simulation is consistent with the previous invocation of $\mathcal{S}^f_{\hat{\mathsf{A}}',\hat{\mathsf{B}}}$. In that case, $\hat{\mathsf{A}}'$ inputs $\top$ as second input to $\mathcal{F}^f_{\mathtt{SFE}}$, which outputs y as final result. Otherwise, the input is $\perp$, yielding output $\perp$ from $\mathcal{F}^f_{\mathtt{SFE}}$ and modeling the case where a wrong m makes A' fail in the proof of knowledge.

Therefore, the only difference between the simulation with $\mathcal{F}^f_{\mathtt{SFE}}$ and the real procedure $\mathit{\Pi}^{\mathtt{SFE}(f)}_{\mathsf{A}',\mathsf{B}}$ is A''s views, simulated by $\mathcal{S}^f_{\hat{\mathsf{A}}',\hat{\mathsf{B}}}$ and actually produced by $\mathit{\Pi}^f_{\mathsf{A}',\mathsf{B}}$, respectively. These views, however, are by assumption quantum-computationally indistinguishable. ■