Non-interactive zero-knowledge proofs in the quantum random oracle model

Dominique Unruh
University of Tartu

July 29, 2014

Abstract

We present a construction for non-interactive zero-knowledge proofs of knowledge in the random oracle model from general sigma-protocols. Our construction is secure against quantum adversaries. Prior constructions (by Fiat-Shamir and by Fischlin) are only known to be secure against classical adversaries, and Ambainis, Rosmanis, Unruh (FOCS 2014) gave evidence that those constructions might not be secure against quantum adversaries in general.

To prove security of our constructions, we additionally develop new techniques for adaptively programming the quantum random oracle.

Contents

1 Introduction

Classical NIZK proofs. Zero-knowledge proofs are an vital tool in modern cryptography. Traditional zero-knowledge proofs (e.g., [GMW91]) are interactive protocols, this makes them cumbersome to use in many situations. To circumvent this problem, non-interactive zero-knowledge (NIZK) proofs where introduced [BFM88]. NIZK proofs circumvent the necessity for interaction by introducing a CRS, which is a publicly known value that needs to be chosen by a trusted third party. The ease of use of NIZK proofs comes at a cost, though: generally, NIZK proofs will be less efficient and based on stronger assumptions than their interactive counterparts. So-called sigma protocols (a certain class of three move interactive proofs, see below) exist for a wide variety of problems and admit very generic operations for efficiently constructing more complex ones [CDS94, Dam10] (e.g., the "or" of two sigma protocols). In contrast, efficient NIZK proofs using a CRS exist only for specific languages (most notably related to bilinear groups, using Groth-Sahai proofs [GS08]). To alleviate this, [FS87] introduced so-called Fiat-Shamir proofs that are NIZK proofs in the random oracle model.[1] Those can transform any sigma protocol into a NIZK proof. (In fact the construction is even a proof *of knowledge*, but we will ignore this distinction for the moment.) The Fiat-Shamir construction (or variations of it) has been used in a number of notable protocols, e.g., Direct Anonymous Attestation [BCC04] and the Helios voting system [Adi08]. A second construction of NIZK proofs in the random oracle model was proposed by Fischlin [Fis05]. Fischlin's construction is less efficient than Fiat-Shamir (and imposes an additional condition on the sigma protocol, called "unique responses"), but it avoids certain technical difficulties that Fiat-Shamir has (Fischlin's construction does not need rewinding).

Quantum NIZK proofs. However, if we want security against quantum adversaries, the situation becomes worse. Groth-Sahai proofs are not secure because they are based on hardness assumptions in bilinear groups that can be broken by Shor's algorithm [Sho94]. And [ARU14b] shows that the Fiat-Shamir construction is not secure in general, at least relative to a specific oracle. Although this does not exclude that Fiat-Shamir is still secure without oracle, it at least makes a proof of security less likely – at the least, such a security proof would be non-relativizing, while all known proof techniques that deal with rewinding in the quantum case [Wat09, Unr12] are relativizing. Similarly, [ARU14b] also shows Fischlin's scheme to be insecure in general (relative to an oracle). Of course, even if Fiat-Shamir and Fischlin's construction are insecure in general, for certain specific sigma-protocols, Fiat-Shamir or Fischlin could still be secure. (Recall that both constructions take an *arbitrary* sigma-protocol and convert it into a NIZK proof.) In fact, [DFG13] shows that for a specific class of sigma-protocols (with so-called "oblivious commitments"), a *variant* of Fiat-Shamir is secure[2]. However, sigma-protocols with oblivious commitments are themselves already NIZK proofs in the CRS model.[3] (This is not immediately obvious from the definition presented in [DFG13], but we show this fact in Appendix A.) Also, sigma-protocols with oblivious commitments are not closed under disjunction and similar operations (at least not using the constructions from [CDS94]), thus losing one of the main advantages of sigma-protocols for efficient protocol design. Hence sigma-protocols with oblivious commitments are a much stronger assumption than just normal sigma-protocols, we loose one of the main advantages of the classical Fiat-Shamir construction: the ability to transform *arbitrary* sigma-protocols into NIZK proofs. Summarizing, prior to this paper, no generic quantum-secure construction was known to transform sigma-protocols into NIZK proofs or NIZK proofs of knowledge in the random oracle model. ([DFG13] left this explicitly as an open problem.)

Our contribution. We present a NIZK proof system in the random oracle model, secure against quantum adversaries. Our construction takes any sigma protocol (that has the standard properties "honest verifier zero-knowledge" (HVZK) and "special soundness") and transforms it into a non-interactive proof. The resulting proof is a zero-knowledge proof of knowledge (secure against polynomial-time quantum adversaries) with the extra property of "online extractability". This property guarantees that the witness from a proof can be extracted without rewinding. (Fischlin's scheme also has this property in the classical setting, but not Fiat-Shamir.) Furthermore the scheme is non-malleable, more precisely simulation-sound.

[1] [FS87] originally introduced them as a heuristic construction for signatures schemes (with a security proof in the random oracle model by [PS96]). However, the construction can be seen as a NIZK proof of knowledge in the random oracle model.

[2] Security is shown for Fiat-Shamir as a signature scheme, but the proof technique most likely also works for Fiat-Shamir as a NIZK proof of knowledge.

[3] This observation does not trivialize the construction from [DFG13] because a sigma-protocol with oblivious commitments is a *non-adaptive single-theorem* NIZK proof in the CRS model while the construction from [DFG13] yields an *adaptive multi-theorem* NIZK proof in the random oracle model. See Appendix A.

That is, given a proof for one statement, it is not possible to create a proof for a related statement. This property is, e.g., important if we wish to construct a signature-scheme from the NIZK proof.

As an application we show how to use our proof system to get strongly unforgeable signatures in the quantum random oracle model from any sigma protocol (assuming a generator for hard instances).

In order to prove the security, we additionally develop a result on random oracle programming in the quantum setting (Theorem 10 below) which is a strengthening of a lemma from [Unr14b, Unr14a] to the adaptive case. It allows us to reduce the probability that the adversary notices that a random oracle has been reprogrammed to the probability of said adversary querying the oracle at the programmed location. (This would be relatively trivial in a classical setting but becomes non-trivial if the adversary can query in superposition.)

Further related work. [DFG13] shows the impossibility of proving the quantum security of Fiat-Shamir using a reduction that does not perform quantum rewinding.[4] [ARU14b] shows the quantum insecurity of Fiat-Shamir and Fischlin's scheme relative to an oracle (and therefore the impossibility of a relativizing proof, even with quantum rewinding). [FKMV12] shows that Fiat-Shamir is zero-knowledge and simulation-sound extractable (not simulation-sound online-extractable) in the classical setting under the additional assumption of "unique responses" (a.k.a. computational strict soundness). [Fis05] shows that Fischlin's construction is zero-knowledge and online-extractable (not simulation-sound online-extractable) in the classical setting assuming unique responses.

Difficulties with Fiat-Shamir and Fischlin. In order to understand our protocol construction, we first explain why Fiat-Shamir and Fischlin's scheme are difficult to prove secure in the quantum setting. A sigma-protocol consists of three messages $\mathit{com}, \mathit{ch}, \mathit{resp}$ where the "commitment" com is chosen by the prover, the "challenge" ch is chosen uniformly at random by the verifier, and the "response" resp is computed by the prover depending on ch. Given a sigma-protocol, and a random oracle H, the Fiat-Shamir construction produces the commitment com, computes the challenge $\mathit{ch} := H(\mathit{com})$, and computes a response resp for that challenge. The proof is then $\pi := (\mathit{com}, \mathit{ch}, \mathit{resp})$, and the verifier checks whether it is a valid execution of the sigma-protocol, and whether $\mathit{ch} = H(\mathit{com})$. How do we prove that Fiat-Shamir is a proof (or a proof of knowledge)? (The zero-knowledge property is less interesting for the present discussion, so we skip it.) Very roughly, given a malicious prover P, we first execute P to get $(\mathit{com}, \mathit{ch}, \mathit{resp})$. Then we rewind P to the oracle query $H(\mathit{com})$ that returned ch. We then change ("program") the random oracle such that $H(\mathit{com}) := \mathit{ch}'$ for some random $\mathit{ch}' \neq \mathit{ch}$. And then we then continue the execution of P with the modified oracle H. Then P will output a new triple $(\mathit{com}', \mathit{ch}', \mathit{resp}')$. And since com was determined before the point of rewinding, we have $\mathit{com} = \mathit{com}'$. (This is a vague intuition. But the "forking lemma" [PS96] guarantees that this actually works with sufficiently large probability.) Then we can use a property of sigma-protocols called "special soundness". It states: given valid sigma-protocol interactions $(\mathit{com}, \mathit{ch}, \mathit{resp}), (\mathit{com}, \mathit{ch}', \mathit{resp}')$, one can efficiently compute a witness for the statement being proven. Thus we have constructed an extractor that, given a (successful) malicious prover P, finds a witness. This implies that Fiat-Shamir is a proof of knowledge.

What happens if we try and translate this proof idea into the quantum setting? First of all, rewinding is difficult in the quantum setting. We can rewind P by applying the inverse unitary transformation $P^\dagger$ to reconstruct an earlier state of P. However, if we measure the output of P before rewinding, this disturbs the state, and the rewinding will return to an undefined earlier state. In some situations this can be avoided by showing that the output that is measured contains little information about the state and thus does not disturb the state too much [Unr12], but it is not clear how to do that in the case of Fiat-Shamir. (The output $(\mathit{com}, \mathit{ch}, \mathit{resp})$ may contain a lot of entropy due to $\mathit{com}, \mathit{ch}$, even if we require resp to be unique.)

Even if we have solved the problem of rewinding, we face a second problem. We wish to reprogram the random oracle at the input where it is being queried. Classically, the input of a random oracle query is a well-defined notion. In the quantum setting, though, the query input may be in superposition, and we cannot measure the input because this would disturb the state.

So when trying to prove Fiat-Shamir secure, we face two problems to which we do not have a solution: rewinding, and determining the input to an oracle query.

We now turn to Fischlin's scheme. Fischlin's scheme was introduced in the classical case to avoid the rewinding used in Fiat-Shamir. (There are certain reasons why even classically, rewinding leads to problems, see [Fis05].) Here the prover is supposed to send a valid triple $(\mathit{com}, \mathit{ch}, \mathit{resp})$ such that

[4] I.e., a reduction that cannot apply the inverse of the unitary describing the adversary.

$\boldsymbol{P_{OE}}$:

Input: (x, w) with $(x, w) \in R$

`// Create` $t \cdot m$ `proofs` $(com_i, ch_{i,j}, resp_{i,j})$
for $i = 1$ **to** t **do**
  $com_i \leftarrow P^1_\Sigma(x, w)$
  for $j = 1$ **to** m **do**
    $ch_{i,j} \stackrel{\$}{\leftarrow} N_{ch} \setminus \{ch_{i,1}, \dots, ch_{i,j-1}\}$
    $resp_{i,j} \leftarrow P^2_\Sigma(ch_{i,j})$

`// Commit to responses`
for $i = 1$ **to** t **do**
  for $j = 1$ **to** m **do**
    $h_{i,j} := G(resp_{i,j})$

`// Get challenge by hashing`
$J_1 \| \dots \| J_t := H(x, (com_i)_i, (ch_{i,j})_{i,j}, (h_{i,j})_{i,j})$

`// Return proof (only some responses)`
return $\pi := \big((com_i)_i, (ch_{i,j})_{i,j}, (h_{i,j})_{i,j}, (resp_{i,J_i})_i\big)$ [5]

$\boldsymbol{V_{OE}}$:

Input: (x, π) with
$\pi = \big((com_i)_i, (ch_{i,j})_{i,j}, (h_{i,j})_{i,j}, (resp_i)_i\big)$

$J_1 \| \dots \| J_t := H(x, (com_i)_i, (ch_{i,j})_{i,j}, (h_{i,j})_{i,j})$

for $i = 1$ **to** t **do**
  check $ch_{i,1}, \dots, ch_{i,m}$ pairwise distinct
for $i = 1$ **to** t **do**
  check $V_\Sigma(x, com_i, ch_{i,J_i}, resp_i) = 1$
for $i = 1$ **to** t **do**
  check $h_{i,J_i} = G(resp_i)$.
if *all checks succeed* **then**
  return 1

Figure 1: Prover $P^{G,H}_{OE}(x, w)$ (left) and verifier $V^{G,H}_{OE}(x, \pi)$ (right) from Definition 13. The missing notation will be introduced in Section 2.2.

$H(com, ch, resp) \bmod 2^b = 0$ for a certain parameter b. (This is an oversimplification but good enough for explaining the difficulties.) By choosing b large enough, a prover can only find triples $(com, ch, resp)$ with $H(com, ch, resp) \bmod 2^b = 0$ by trying out a several such triples. Thus, if we inspect the list of all query inputs to H, we will find several different valid triples $(com, ch, resp)$. In particular, there will be two triples $(com, ch, resp)$ and $(com', ch', resp')$ with $com = com'$. (Due to the oversimplified presentation here, the reader will have to take on trust that we can achieve $com = com'$, see [Fis05] for a full analysis.) Again using special soundness, we can extract a witness from these two triples. So Fischlin's scheme is a proof of knowledge with the extra benefit that the extractor can extract without rewinding, just by looking at the oracle queries ("online-extraction").

What happens if we try to show the security of Fischlin's scheme in the quantum setting? Then we again face the problem that there is no well-defined notion of "the list of query inputs". If we measure the query inputs, this disturbs the malicious prover. If we do not measure the query inputs, they are not well-defined.

The problems with Fiat-Shamir and Fischlin seem not to be just limitations of our proof techniques, [ARU14b] shows that relative to some oracle, Fiat-Shamir and Fischlin actually become insecure.

Our protocol. So both in Fiat-Shamir and in Fischlin's scheme we face the challenge that it is difficult to get the query inputs made by the malicious prover. Nevertheless, in our construction we will still try to extract the query inputs, but with a twist: Assume for a moment that the random oracle G is a permutation. Then, given $G(x)$ it is, at least in principle, possible to extract x. Can we use this idea to save Fischlin's scheme? No, because in Fischlin's scheme we need the inputs to queries whose outputs we never learn; inverting G will not help. So in our scheme, for any query input x we want to learn, we need to include $G(x)$ in the output. Basically, we sent $(com, G(resp_1), \dots, G(resp_n))$ where the $resp_j$ are the responses for com given different challenges ch_j. Then, by inverting two of the G, we can get two triples $(com, ch, resp)$ and $(com, ch', resp')$ which allows us to extract the witness. However, so far we have not made sure that the malicious prover indeed puts valid responses into the queries. He could simply send random values instead of $G(resp_j)$. To avoid this, we use a cut-and-choose technique similar to what is done in Fiat-Shamir: We first produce a number of proofs $(com_i, G(resp_{i,1}), \dots, G(resp_{i,n}))$. Then we hash all of them with a second random oracle H (not a permutation). The result of the hashing indicates for each com_i which of the $resp_{i,j}$ should be revealed. A malicious prover who succeeds in this will have to include valid responses in at least a large fraction of the $G(resp_{i,j})$. Thus by inverting G, we can find two valid triples $(com, ch, resp)$ and $(com, ch', resp')$ if the malicious prover's proof passes verification. The full protocol is described in Figure 1.

We have not discussed yet: What if G is not a permutation (a random function will usually not be a permutation)? And how to efficiently invert G? The answer to the first is: as long as domain and range of G are the same, G is indistinguishable from a random permutation [Zha13]. So although the real protocol execution uses G that is a random function, in an execution with the extractor, we simply feed a random permutation to the prover. To answer the second, we need to slightly change our approach (but not the protocol): [Zha12] shows that a random function is indistinguishable from a $2q$-wise independent function (where q is the number of oracle queries performed). Random polynomials of degree $2q-1$ are $2q$-wise independent. So if, during extraction, we replace G not by a random permutation, but by a random polynomial, we can efficiently invert G. (The preimage will not be unique, but the number of possible preimage will be small enough so that we can scan through all of them.) This shows that our protocol is online-extractable: the extractor simply replaces G by a random polynomial, inverts all $G(resp_{i,j})$, searches for two valid triples $(com, ch, resp)$ and $(com, ch', resp')$, and computes the witness. The formal description of the extractor is given in Section 4.3.

Organization. In Section 2 we introduce the main security notions used in this paper: those of non-interactive proof systems in the random oracle model (Section 2.1) and those of sigma-protocols (Section 2.2). In Section 3 we review some simple results on quantum random oracles and then prove our results on adaptive random oracle programming. In Section 4 we introduce and prove secure our NIZK proof system. In Section 5 we illustrate the use of our results and construct a signature scheme in the random oracle model from sigma-protocols.

1.1 Preliminaries

By $x \leftarrow A(y)$ we denote the (quantum or classical) algorithm A executed with (classical) input y, and its (classical) output assigned to x. We write $x \leftarrow A^H(y)$ if A has access to an oracle H. We stress that A may query the random oracle H in superposition. By $x \xleftarrow{\$} M$ we denote that x is uniformly randomly chosen from the set M. $\Pr[P : G]$ refers to the probability that the predicate P holds true when the free variables in P are assigned according to the program (game) in G. All algorithms implicitly depend on a security parameter η that we never write. If we say a quantity is *negligible* or *overwhelming*, we mean that it is in $o(\eta^c)$ or $1 - o(\eta^c)$ for all $c > 0$ where η denote the security parameter. A *polynomial-time* algorithm is a *classical* one that runs in polynomial-time in its input length and the security parameter, and a *quantum-polynomial-time* algorithm is a *quantum* algorithm that runs in polynomial-time in input and security parameter.

$\{0,1\}^n$ are the bitstrings of length n, $\{0,1\}^{\leq n}$ the bitstrings of length at most n, and $\{0,1\}^*$ those of any length. $(M \to N)$ refers to the set of all functions from M to N. $a\|b$ is the concatenation of bitstrings a and b. $\mathrm{GF}(2^n)$ is a finite field of size 2^n, and $\mathrm{GF}(2^n)[X]$ is the set of polynomials over that field. ∂p refers to the degree of the polynomial p. The *collision entropy* of a random variable X is $-\log \Pr[X = X']$ where X' is independent of X and has the same distribution. The *min-entropy* is $\min_x(-\log \Pr[X = x])$. A family of functions F is called *q-wise-independent* if for any distinct $x_1, \dots, x_q$ and for $f \xleftarrow{\$} F$, $f(x_1), \dots, f(x_q)$ are independently uniformly distributed. $\mathrm{E}[X]$ is the expected value of the random variable X.

$\mathrm{TD}(\rho, \rho')$ denotes the trace distance between two density operators.

2 Security notions

In the following we present the security notions used in this work. All security notions capture security against quantum adversaries. To make the notions strongest possible, we formulate them with respect to quantum adversaries, but classical honest parties (and classical simulators and extractors).

2.1 Non-interactive proof systems

In the following, we assume a fixed efficiently decidable relation R on bitstrings, defining the language of our proof systems. That is, a *statement* x is in the language iff there exists a *witness* w with $(x, w) \in R$. We also assume a distribution ROdist on functions, modeling the distributions of our random oracle. (E.g., for a random oracle $H : \{0,1\}^* \to \{0,1\}^n$, ROdist would be the uniform distribution on $\{0,1\}^* \to \{0,1\}^n$.)

[5] The values h_{i,J_i} could be omitted since they can be recomputed as $h_{i,J_i} = G(resp_{i,J_i})$. We include them to keep the notation simple.

A *non-interactive proof system* consists of two polynomial-time oracle algorithms $P(x,w), V(x,\pi)$. (The argument π of V represents the proof produced by P.) We require that $P^H(x,w)=\bot$ whenever $(x,w)\notin R$ and that $V^H(x,\pi)\in\{0,1\}$. Inputs and outputs of P and V are classical.

Definition 1 (Completeness) *(P,V) is* complete *iff for any quantum-polynomial-time oracle algorithm A, we have that*

$$\Pr[(x,w)\in R\wedge ok=0 : H\leftarrow \mathsf{ROdist}, (x,w)\leftarrow A^H(), \pi\leftarrow P^H(x,w), ok\leftarrow V^H(x,\pi)]$$

is negligible.

Zero-knowledge. We now turn to the zero-knowledge property. Zero-knowledge means that an adversary cannot distinguish between real proofs and proofs produced by a simulator (that has no access to the witness). In the random oracle model, we furthermore allow the simulator to control the random oracle. Classically, this means in particular that the simulator learns the input for each query, and can decide on the response adaptively. In the quantum setting, this is not possible: since the random oracle can be queried in superposition, measuring its input would disturb the state of the adversary. We chose an alternative route here: the simulator is allowed to output a circuit that represents the function computed by the random oracle. And he is allowed to update that circuit whenever he is invoked. However, the simulator is *not* invoked upon a random oracle query. (This makes the definition only stronger.) We now proceed to the formal definition:

A *simulator* is a pair of classical algorithms (S_{init}, S_P). S_{init} outputs a circuit H describing a classical function which represents the initial (simulated) random oracle. The stateful algorithm $S_P(x)$ returns a proof π. Additionally S_P is given access to the description H and may replace it with a different description (i.e., it can program the random oracle).

Definition 2 (Zero-knowledge) *Given a simulator (S_{init}, S_P), the oracle $S'_P(x,w)$ does: If $(x,w)\notin R$, return $\bot$. Else return $S_P(x)$. (The purpose of S'_P is merely to serve as an interface for the adversary who expects a prover taking two arguments x,w.)*

A non-interactive proof system (P,V) is zero-knowledge *iff there is a polynomial-time simulator (S_{init}, S_P) such that for every quantum-polynomial-time oracle algorithm A, we have that*

$$\left|\Pr[b=1 : H\leftarrow\mathsf{ROdist}, b\leftarrow A^{H,P}()] - \Pr[b=1 : H\leftarrow S_{init}(), b\leftarrow A^{H,S'_P}()]\right| \text{ is negligibe.} \tag{1}$$

We assume that both S_{init} and S_P have access to and may depend on a polynomial upper bound on the runtime of A.

The reason why we allow the simulator to know an upper bound of the runtime of the adversary is that we use the technique of [Zha12] of using q-wise independent hash functions to mimic random functions. This approach requires that we know upper bounds on the number and size of A's queries; the runtime of A provides such bounds.

Online-extractability. We will now define online-extractability. Online-extractable proofs are a specific form of proofs of knowledge where extraction is supposed to work by only looking at the proofs generated by the adversary and at the oracle queries performed by him. Unfortunately, in the quantum setting, it is not possible to generate (or even define) the list of oracle queries because doing so would imply measuring the oracle input, which would disturb the adversary's state. So, different from the classical definition in [Fis05], we do not give the extractor the power to see the oracle queries. Is it then possible at all for the extractor to extract? Yes, because we allow the extractor to see the description of the random oracle H that was produced by the simulator S_{init}. If the simulator produces suitable circuit descriptions, those descriptions may help the extractor to extract in a way that would not be possible with oracle access alone. We now proceed to the formal definition:

An *extractor* is an algorithm $E(H,x,\pi)$ where H is assumed to be a description of the random oracle, x a statement and π a proof of x. E is supposed to output a witness. Inputs and outputs of E are classical.

Definition 3 (Online extractability) *A non-interactive proof system (P,V) is* online extractable *with respect to S_{init} iff there is a polynomial-time extractor E such that for any quantum-polynomial-time oracle algorithm A, we have that*

$$\Pr[ok=1\wedge(x,w)\notin R : H\leftarrow S_{init}(), (x,\pi)\leftarrow A^H(), ok\leftarrow V^H(x,\pi), w\leftarrow E(H,x,\pi)] \text{ is negligible.}$$

We assume that both S_{init} and E have access to and may depend on a polynomial upper bound on the runtime of A.

Online-extractability intuitively implies that it is not possible for an adversary to produce a proof for a statement for which he does not know a witness (because the extractor can extract a witness from what the adversary produces). However, it does not exclude that the adversary can take one proof π_1 for one statement x_1 and transform it into a valid proof for another statement x_2 (even without knowing a witness for x_2), as long as a witness for x_2 could efficiently be computed from a witness for x_1. This problem is usually referred to as malleability.

To avoid malleability, one definitional approach is simulation-soundness [Sah99, Gro06]. The idea is that extraction of a witness from the adversary-generated proof should be successful even if the adversary has access to simulated proofs (as long as the adversary generated proof does not equal one of the simulated proofs). Adapting this idea to online-extractability, we get:

Definition 4 (Simulation-sound online-extractability) *A non-interactive proof system (P,V) is* simulation-sound online-extractable *with respect to simulator (S_{init}, S_P) iff there is a polynomial-time extractor E such that for any quantum-polynomial-time oracle algorithm A, we have that*

$$\Pr[ok = 1 \land (x,\pi) \notin simproofs \land (x,w) \notin R : \\ H \leftarrow S_{init}(), (x,\pi) \leftarrow A^{H,S_P}(), ok \leftarrow V^H(x,\pi), w \leftarrow E(H,x,\pi)]$$

is negligibe. Here simproofs is the set of all proofs returned by S_P (together with the corresponding statements).

We assume that S_{init}, S_P, and E have access to and may depend on a polynomial upper bound on the runtime of A.

Notice that A^{H,S_P} gets access to S_P, not to S'_P. That is, A can even create simulated proofs of statements where he does not know the witness.

2.2 Sigma protocols

We now introduce sigma protocols. The notions in this section are standard, all we do to adopt them to the quantum setting is to make the adversary quantum-polynomial-time. Note that the definitions are formulated without the random oracle, we only use the random oracle for constructing a NIZK proof out of the sigma protocol.

A *sigma protocol* for a relation R is a three message proof system. It is described by the domains $N_{com}, N_{ch}, N_{resp}$ of the messages (where $|N_{ch}| \geq 2$), a polynomial-time prover (P_1, P_2) and a deterministic polynomial-time verifier V. The first message from the prover is $com \leftarrow P_1(x,w)$ and is called the *commitment*, the uniformly random reply from the verifier is $ch \xleftarrow{\$} N_{ch}$ (called *challenge*), and the prover answers with $resp \leftarrow P_2(ch)$ (the *response*). We assume P_1, P_2 to share state. Finally $V(x, com, ch, resp)$ outputs whether the verifier accepts.

Definition 5 (Properties of sigma protocols) *Let $(N_{com}, N_{ch}, N_{resp}, P_1, P_2, V)$ be a sigma protocol. We define:*

- ***Completeness:*** *For any quantum-polynomial-time algorithm A, $\Pr[(x,w) \in R \land ok = 0 : (x,w) \leftarrow A, com \leftarrow P_1(x,w), ch \xleftarrow{\$} N_{ch}, resp \leftarrow P_2(ch), ok \leftarrow V(x, com, ch, resp)]$ is negligible.*
- ***Computational special soundness:*** *There is a polynomial-time algorithm E_Σ such that for any quantum-polynomial-time A, we have that*

$$\Pr[(x,w) \notin R \land ch \neq ch' \land ok = ok' = 1 : (x, com, ch, resp, ch', resp') \leftarrow A(), \\ ok \leftarrow V(x, com, ch, resp), ok' \leftarrow V(x, com, ch', resp'), w \leftarrow E_\Sigma(x, com, ch, resp, ch)]$$

  is negligible.
- ***Honest-verifier zero-knowledge (HVZK):*** *There is a polynomial-time algorithm S_Σ (the simulator) such that for any stateful quantum-polynomial-time algorithm A the following is negligible for all $(x,w) \in R$:*

$$\Big| \Pr[b = 1 : (x,w) \leftarrow A(), com \leftarrow P_1(x,w), ch \xleftarrow{\$} N_{ch}, resp \leftarrow P_2(ch), b \leftarrow A(com, ch, resp)] \\ - \Pr[b = 1 : (x,w) \leftarrow A(), (com, ch, resp) \leftarrow S(x), b \leftarrow A(com, ch, resp)] \Big|$$

Note that the above are the standard conditions expected from sigma-protocols in the classical setting. In contrast, for a sigma-protocol to be a *quantum* proof of knowledge, a much more restrictive condition is required, strict soundness [Unr12, ARU14b]. Interestingly, this condition is not needed for our protocol to be quantum secure.

3 Quantum random oracles

In this section, we state and prove various lemmas about quantum random oracles that we need in our security proofs. Some of them are known or straightforward extensions of known results. However, the results on adaptive programming below need considerable extensions of prior proofs.

Simple facts. The following is a slight extension of [ARU14a, Lemma 38]. That lemma says that a (sufficiently sparse) random function F cannot be distinguished from the constant zero function N. Here, we show that this even holds if the adversary gets access to the full description of F *after the last query*.

Lemma 6 (Preimage search in a random function) *Let $\gamma \in [0,1]$. Let Z be a finite set. Let $q \geq 0$ be an integer. Let $F : Z \to \{0,1\}$ have the following distribution: Each $F(c)$ is independently Bernoulli-distributed with $\Pr[F(c) = 1] = \gamma$. Let N be the function with $\forall z : N(z) = 0$.*

For any oracle algorithm A that makes at most q queries, and any algorithm A' that can access the final state of A, we have

$$\left|\Pr[b = 1 : A^F(), b \leftarrow A'(F)] - \Pr[b = 1 : A^N(), b \leftarrow A'(F)]\right| \leq 2q\sqrt{\gamma}.$$

Here $A'(F)$ means that A' gets a description of F (e.g., a value table), not just oracle access to F.

Proof. The proof is identical to that of [ARU14a, Lemma 38], except that the last calculation is replaced with:

$$\begin{aligned}
&\left|\Pr[b = 1 : A^F(), b \leftarrow A'(F)] - \Pr[b = 1 : A^N(), b \leftarrow A'(F)]\right| \\
&\leq \sum_f \alpha_f \left|\Pr[b = 1 : A^f(), b \leftarrow A'(f)] - \Pr[b = 1 : A^N(), b \leftarrow A'(f)]\right| \\
&\leq \sum_f \alpha_f \operatorname{TD}(|\Psi_f^q\rangle, |\Psi^q\rangle) \leq 2q\sqrt{\lambda}.
\end{aligned}$$

□

Lemma 7 *Fix $\gamma \in [0,1]$. Let $F : Z \to \{0,1\}$ have the following distribution: Each $F(c)$ is independently Bernoulli-distributed with $\Pr[F(c) = 1] = \gamma$. Let S be an algorithm making at most q queries to F. Then $\mu := \Pr[F(c) = 1 : c \leftarrow S^F()] \leq 2(q+1)\sqrt{\gamma}$.*

Proof. S immediately gives rise to an oracle algorithm doing $q+1$ queries that distinguishes F from the constant zero function with probability μ. Lemma 6 shows that this distinguishing probability is at most $2(q+1)\sqrt{\gamma}$. □

Theorem 8 (Finding collisions) *Let $G : \{0,1\}^m \to \{0,1\}^n$ be uniformly distributed. Let A be an oracle algorithm making at most q queries to G. Then $\Pr[G(x) = G(x') \wedge x \neq x' : (x,x') \leftarrow A^G()] \leq C(q+1)^3 2^{-n}$ for some C (that is independent of A, q, n, m).*

Proof. Shown in [Zha13, Theorem 3.1]. □

Adaptive programming. The following lemma is a generalization of [Unr14a, Lemma 14]. The difference is that in [Unr14a], the position where the random oracle is queried is of the form $x\|m$ where x is random and m is adversarially chosen. In contrast, here the oracle is queried at an adversarially chosen x which is only required to have high min-entropy. The proof from [Unr14a] does no apply in that case because it relies on the fact that part of $x\|m$ (namely x) is chosen independently of the adversary's state, and that $x\|m$ uniquely determines x. The lemma from [Unr14a] can be recovered (with worse bounds) from the present lemma by letting A_C picks x at random and return $x\|m$.

Lemma 9 (One-way to hiding, adaptive) *Let $H : M \to N$ be a random oracle for finite sets M, N. (Infinite $M \subseteq \{0,1\}^*$ is also permissible.) Consider the following algorithms:*

- *The oracle algorithm A_0 that makes at most q_0 queries to H.*
- *The classical algorithm A_C that may access the classical part of the final state of A_0. Assume that for every initial state, the output of A_C has collision entropy at least k.*
- *The oracle algorithm A_1 that may access the final states of A_0 and A_C and makes at most $q_1 \geq 1$ queries to H.*
- *Let C_1 be an oracle algorithm that on input (j, B, x) does the following: run $A_1^H(x, B)$ until (just before) the j-th query, measure the argument of the query in the computational basis, output the measurement outcome. (When A_1 makes less than j queries, C_1 outputs $\bot \notin \{0,1\}^\ell$.)*

Let

$$P_A^1 := \Pr[b' = 1 : H \xleftarrow{\$} (M \to N), A_0^H(), x \leftarrow A_C(), b' \leftarrow A_1^H(x, H(x))]$$

$$P_A^2 := \Pr[b' = 1 : H \xleftarrow{\$} (M \to N), A_0^H(), x \leftarrow A_C(), B \xleftarrow{\$} N, b' \leftarrow A_1^H(x, B)]$$

$$P_C := \Pr[x = x' : H \xleftarrow{\$} (M \to N), A_0^H(), x \leftarrow A_C(), B \xleftarrow{\$} N, j \xleftarrow{\$} \{1, \dots, q_1\}, x' \leftarrow C_1^H(j, B, x)]$$

Then $|P_A^1 - P_A^2| \leq (4 + \sqrt{2})\sqrt{q_0}\, 2^{-k/4} + 2q_1\sqrt{P_C}$.[6]

Note that we do not allow A_C to have access to H, and that A_C is required to be classical. Both conditions are necessary, see the examples after the special case Corollary 11.

Proof. In the following we assume that M is finite. The case of infinite $M =: M' \subseteq \{0,1\}^*$ follows directly by considering $M := M' \cap \{0,1\}^{\leq n}$ for $n \to \infty$. Such a restricted M' leads to an error term in the values of P_A^1, P_A^2, P_C that converges (for fixed A_0, A_C, A_1) towards 0; we then recover the final bound on $|P_A^1 - P_A^2|$ for infinite $M = M'$ as the limit of the bounds for the finite $M := M' \cap \{0,1\}^{\leq n}$.

Without loss of generality, we can assume that A_1^H does not access the final state of A_C. This is because A_1 gets the output x of A_C as input and can just sample the (classical) final state of A_C conditioned on its (classical) input and its (classical) output. (Recall that we do not assume A_1 to be computationally limited.)

Furthermore, we make the classical part of the final state of A_0 explicit and call it m. Thus instead of writing "$A_0^H(), x \leftarrow A_C()$" we write . "$m \leftarrow A_0^H(), x \leftarrow A_C(m)$". Then $A_C(m)$ can be simply considered as a probability distribution, parametric in m.

We define two auxiliary algorithms:

- Let $d := \lceil k/2 + \log q_0 \rceil$. For a function $F : M \to \{0,1\}^d$ and a bitstring $z \in \{0,1\}^d$, and a bitstring m, the algorithm $I(z, m, F)$ samples x according to the distribution $A_C(m)$, conditioned on $F(x) = z$. (Or $\bot$ if no such x exists.)
- Given a set $X \subset M$ and a bitstring m, the algorithm $J(m, X)$ picks a uniformly random $F \in (M \to \{0,1\}^d)$ and $z \in \{0,1\}^d$ conditioned on $X = F^{-1}(\{z\})$. Then J invokes $x \leftarrow I(z, m, F)$ and returns x.

We then have:

$$\begin{aligned}
P_A^1 &= \Pr[b' = 1 : H \xleftarrow{\$} (M \leftarrow N),\ m \leftarrow A_0^H(),\ x \leftarrow A_C(m),\ b' \leftarrow A_1^H(x, H(x))] \\
&= \Pr[b' = 1 : H \xleftarrow{\$} (M \leftarrow N),\ m \leftarrow A_0^H(),\ F \leftarrow (M \to \{0,1\}^d), z \leftarrow \mathcal{D}_{F,m}, \\
&\qquad x \leftarrow I(z, m, F),\ b' \leftarrow A_1^H(x, H(x))]
\end{aligned}$$

where $\mathcal{D}_{F,m}$ is the distribution resulting from picking $x \leftarrow A_C(m), z := F(x)$ and returning z. Then the equality follows by definition of I.

$$\begin{aligned}
\cdots &\overset{\varepsilon_1}{\approx} \Pr[b' = 1 : H \xleftarrow{\$} (M \leftarrow N),\ m \leftarrow A_0^H(),\ F \leftarrow (M \to \{0,1\}^d), z \xleftarrow{\$} \{0,1\}^d, \\
&\qquad x \leftarrow I(z, m, F),\ b' \leftarrow A_1^H(x, H(x))]
\end{aligned}$$

[6] We conjecture that the term $2^{-k/4}$ is an artifact of our proof technique. By analogy to the special case [Unr14a, Lemma 14], we might hope for the bound $O(q_0 2^{-k/2} + q_1\sqrt{P_C})$.

where $a \overset{\varepsilon_1}{\approx} b$ means $|a-b| \leq \varepsilon_1 := 2^{(d-k)/2-1}$. We use the convention that if I returns $\bot$, A_1^H is not executed and $b' := 0$ (same for the algorithm J below). The last step follows because $A_C(m)$ has collision-entropy at least k for any m, and thus for uniformly random F, $\mathcal{D}_{F,m}$ has statistical distance ε_1 from uniform (even given F) by the leftover hash lemma [HILL99, Lemma 4.8] (using the fact that random functions are in particular universal hash functions).[7]

$$\cdots = \Pr[b' = 1 : H \overset{\$}{\leftarrow} (M \leftarrow N),\ m \leftarrow A_0^H(),\ X \leftarrow \mathcal{D}', x \leftarrow J(m,X),\ b' \leftarrow A_1^H(x, H(x))]$$

where $\mathcal{D}'$ returns a set $X \subseteq N$ which contains each $x \in N$ independently with probability 2^{-d}. The equality then follows by definition of J and since $F(x) = z$ with probability 2^{-d} for uniform F.

$$\begin{aligned}\cdots &= \Pr[b' = 1 : H \overset{\$}{\leftarrow} (M \leftarrow N),\ X \leftarrow \mathcal{D}',\ m \leftarrow A_0^H(),\ x \leftarrow J(m,X),\ b' \leftarrow A_1^H(x, H(x))] \\ &\overset{\varepsilon_2}{\approx} \Pr[b' = 1 : H \overset{\$}{\leftarrow} (M \leftarrow N),\ X \leftarrow \mathcal{D}',\ m \leftarrow A_0^{H\setminus X}(),\ x \leftarrow J(m,X),\ b' \leftarrow A_1^H(x, H(x))]\end{aligned}$$

where $H \setminus X$ denotes the oracle that return $H(x)$ on input $x \notin X$, $\bot$ on input $x \in X$. And $\varepsilon_2 := 2q_0 2^{-d/2}$. The $\overset{\varepsilon_2}{\approx}$-part then follows by reduction to Lemma 6 with $\gamma := 2^{-d}$: the adversary A^F picks $H \overset{\$}{\leftarrow} (M \to N)$ and runs $A_0^{H\setminus \operatorname{im} F}$ (this can be done with one F-query for each query of A_0), and $A'(F)$ computes $X := \operatorname{im} F$ and executes $x \leftarrow J(m,X),\ b' \leftarrow A_1^H(x, H(x))$.

Summarizing, we have

$$\begin{aligned} P_A^1 \overset{\varepsilon_1+\varepsilon_2}{\approx} \Pr[b' = 1 : H \overset{\$}{\leftarrow} (M \leftarrow N),\ X \leftarrow \mathcal{D}',\ m \leftarrow A_0^{H\setminus X}(), \\ x \leftarrow J(m,X),\ b' \leftarrow A_1^H(x, H(x))] =: P_A^{1a}.\end{aligned}$$

Analogously, we get

$$\begin{aligned}\hat{P}_A^1 &:= \Pr[b' = 1 : H \overset{\$}{\leftarrow} (M \to N),\ m \leftarrow A_0^H(),\ x \leftarrow A_C(m),\ B \overset{\$}{\leftarrow} N, b' \leftarrow A_1^{H_{xB}}(x, B)] \\ &\overset{\varepsilon_1+\varepsilon_2}{\approx} \Pr[b' = 1 : H \overset{\$}{\leftarrow} (M \leftarrow N),\ X \leftarrow \mathcal{D}',\ m \leftarrow A_0^{H\setminus X}(), \\ &\qquad x \leftarrow J(m,X),\ B \overset{\$}{\leftarrow} N,\ b' \leftarrow A_1^{H_{xB}}(x, B)] =: P_A^{1b}\end{aligned}$$

where H_{xB} denote the oracle that returns $H(x')$ on input $x' \neq x$, and that returns B on input x.

By construction, J either outputs $\bot$ or some $x \in X$. Thus either A_1 is not executed, or $H(x)$ is a position of the random oracle that is never queried by $A_0^{H\setminus X}$. Thus replacing H by H_{xB} cannot be noticed by A_1, thus $P_A^{1a} = P_A^{1b}$. (Notice: the second argument B of A_1 in P_A^{1b} is equal to $H_{xB}(x)$.)

Thus, summarizing we get $|P_A^1 - \hat{P}_A^1| \leq 2\varepsilon_1 + 2\varepsilon_2$.

Furthermore, we can show $|\hat{P}_A^1 - P_A^2| \leq 2q_1\sqrt{P_C}$. This is done as in [Unr14a, Lemma 14], except that in that proof every occurrence of $\beta = 2^{-n|\mathrm{dom}_H|} \cdot 2^{-n} \cdot 2^{-\ell}$ needs to be replaced by $\beta_{xm} := 2^{-n|\mathrm{dom}_H|} \cdot 2^{-n} \cdot \Pr[A_C(m) = x]$.

Summarizing, we have

$$\begin{aligned}|P_A^1 - P_A^2| &\leq 2\varepsilon_1 + 2\varepsilon_2 + 2q_1\sqrt{P_C} \leq 2^{(d-k)/2} + 4q_0 2^{-d/2} + 2q_1\sqrt{P_C} \\ &= 2^{(\lceil k/2+\log q_0\rceil - k)/2} + 4q_0 2^{-\lceil k/2+\log q_0\rceil/2} + 2q_1\sqrt{P_C} \\ &\leq \sqrt{2}\cdot 2^{-k/4+(\log q_0)/2} + 4q_0 2^{-k/4-(\log q_0)/2} + 2q_1\sqrt{P_C} \\ &= (4+\sqrt{2})\sqrt{q_0}\, 2^{-k/4} + 2q_1\sqrt{P_C}. \qquad \square\end{aligned}$$

Given the previous lemma, we can now easily generalize another lemma [Unr14a, Lemma 15] in a similar way. The following lemma shows that we can reprogram the random oracle adaptively:

Theorem 10 (Random oracle programming, adaptive) *Let $H : M \to N$ be a random oracle for finite M, N. (Infinite $M \subseteq \{0,1\}^*$ is also permissible.) Consider the following algorithms:*

- *The oracle algorithm A_0 that makes at most q_0 queries to H.*

[7]The reader may notice it would be sufficient to bound the average statistical distance of $\mathcal{D}_{F,m}$ from uniform (averaged over all m as chosen by A_1^H). Thus, we do not need a bound on the worst-case collision entropy $k := H_2(x)$ here; $k := -2\log(\mathrm{E}[2^{-H_2(x)/2}])$ would do as well.

- *The classical algorithm A_C that may access the classical part of the final state of A_0. Assume that for every initial state, the output of A_C has collision entropy at least k.*
- *The oracle algorithm A_1 that may access the final states of A_0 and A_C.*
- *The oracle algorithm A_2 that may access the final state of A_1; and A_1 and A_2 together perform at most q_{12} queries to H.*
- *Let C_1 be an oracle algorithm that on input (j, B, x) does the following: run $A_1^H(x, B)$ until (just before) the j-th query, measure the argument of the query in the computational basis, output the measurement outcome. (When A_1 makes less than j queries, C_1 outputs $\perp \notin \{0,1\}^\ell$.)*

Let

$$
\begin{aligned}
P_A^1 &:= \Pr[b' = 1 : H \stackrel{\$}{\leftarrow} (M \to N), A_0^H(), x \leftarrow A_C(), A_1^H(x, H(x)), b' \leftarrow A_2^H(x, H(x))] \\
P_A^2 &:= \Pr[b' = 1 : H \stackrel{\$}{\leftarrow} (M \to N), A_0^H(), x \leftarrow A_C(), B \stackrel{\$}{\leftarrow} N, \\
&\qquad\qquad\qquad\qquad\qquad A_1^H(x, B), H(x) := B, b' \leftarrow A_2^H(x, B)] \\
P_C &:= \Pr[x = x' : H \stackrel{\$}{\leftarrow} (M \to N), A_0^H(), x \leftarrow A_C(), \\
&\qquad\qquad\qquad\qquad\qquad B \stackrel{\$}{\leftarrow} N, j \stackrel{\$}{\leftarrow} \{1, \dots, q_{12}\}, x' \leftarrow C_1^H(j, B, x)]
\end{aligned}
$$

Then $|P_A^1 - P_A^2| \le (4 + \sqrt{2})\sqrt{q_0}\, 2^{-k/4} + 2q_{12}\sqrt{P_C}$.

Proof. The proof is almost identical to that of [Unr14a, Lemma 15], but with "$x \leftarrow A_C()$" instead of "$x \stackrel{\$}{\leftarrow} \{0,1\}^\ell$", with "$x$" instead of "$x\|m$", and with "$x = x'$" instead of "$x = x' \wedge m = m'$" everywhere, and using Lemma 9 instead of [Unr14a, Lemma 14]. □

We now state the two special cases of Theorem 10 that we will need in the remainder of this paper:

Corollary 11 *Let M, N be finite sets and $H : M \to N$ be the random oracle. Let A_0, A_C, A_2 be algorithms, where A_0^H makes at most q queries to H, A_C is classical, and the output of A_C is in M and has collision-entropy at least k given A_C's initial state. A_0, A_C, A_2 may share state.*

Then

$$
\begin{aligned}
&\big|\Pr[b = 1 : H \stackrel{\$}{\leftarrow} (M \to N), A_0^H(), x \leftarrow A_C(), B := H(x), b \leftarrow A_2^H(B)] \\
&- \Pr[b = 1 : H \stackrel{\$}{\leftarrow} (M \to N), A_0^H(), x \leftarrow A_C(), B \stackrel{\$}{\leftarrow} N, H(x) := B, b \leftarrow A_2^H(B)]\big| \\
&\qquad\qquad \le (4 + \sqrt{2})\sqrt{q}\, 2^{-k/4}.
\end{aligned}
$$

Note that A_C does not get access to H, otherwise the lemma would be false: A_C could, e.g., return x such that $H(x)$ is even. But A_2 does know x, because A_0, A_C, A_2 share state.

Also the condition that A_C is classical is necessary; the following example illustrates this: A_0^H produces the state $\sum_{x \in M} \frac{1}{\sqrt{|M|}} |x\rangle|H(x)\rangle$ (this is possible with a single H-query). A_C then measures the two registers in this state, giving x and $B' = H(x)$, and returns x. Note that x has min-entropy $\log|M|$ given A_C's initial state. And $A_2^H(B)$ returns $b := 1$ iff $B = B'$. In the first game, $\Pr[B = B'] = 1$, in the second $\Pr[B = B'] = 1/|M|$.

Proof. Let A_1 be an algorithm that does nothing. Then the first probability in the statement of the corollary is equal to P_A^1 in Theorem 10. And the second probability equals P_A^2 in Theorem 10. (In P_A^1, P_A^2, A_1 gets an additional argument x which it ignores.) So we need to bound $|P_A^1 - P_A^2|$. Let q_2 denote an upper bound on the number of oracle queries performed by A_2. By Theorem 10, $|P_A^1 - P_A^2| \le (4 + \sqrt{2})\sqrt{q}\, 2^{-k/4} + 2q_2\sqrt{P_C}$ where $P_C = \Pr[x = x' : \dots,\ x' \leftarrow C_1^H(j, B, x)]$, and C_1^H runs A_1 till the j-th oracle query (and returns $\perp$ if there is no j-th query). Since A_1 does nothing, this implies that C_1^H always returns $\perp$, thus $P_C = \Pr[x = \perp : \dots] = 0$. It follows that $|P_A^1 - P_A^2| \le (4 + \sqrt{2})\sqrt{q}\, 2^{-k/4}$. □

Corollary 12 *Let M, N be finite sets and $H : M \to N$ be the random oracle. Let A_0, A_1 be algorithms that perform at most q_0, q_1 oracle queries, respectively, and that may share state. Let A_C be a classical algorithm that may access (the classical part of) the final state of A_0. (But A_1 does not access A_C's* *state.) Assume that the output of A_C has min-entropy at least k given its initial state. Then*

$$\big|\Pr[b = 1 : H \xleftarrow{\$} (M \to N), A_0^H(), x \leftarrow A_C(), B := H(x), b \leftarrow A_1^H(B)]$$
$$- \Pr[b = 1 : H \xleftarrow{\$} (M \to N), A_0^H(), x \leftarrow A_C(), B \xleftarrow{\$} N, b \leftarrow A_1^H(B)]\big|$$
$$\le (4 + \sqrt{2})\sqrt{q_0}\, 2^{-k/4} + 2q_1 2^{-k/2}.$$

Note that x is never used except for setting $B := H(x)$, otherwise the lemma would be trivially false. Note also that A_C does not share state with A_1 as otherwise x could be leaked to A_1. Finally, note that A_C does not get access to H, and that A_C has to be classical for the same reasons as those discussed after Corollary 11.

Proof. Let A_2 be an algorithm that does nothing except output the bit b that was computed by A_1. Then the first probability in the statement of the corollary is equal to P_A^1 in Theorem 10, and the second probability is equal to P_A^2 in Theorem 10. (In P_A^1, P_A^2, A_1 gets an additional argument x which it ignores.) Thus we have to bound $|P_A^1 - P_A^2|$. Since min-entropy is a lower bound for collision entropy, A_C's output also has collision-entropy at least k. By Theorem 10, $|P_A^1 - P_A^2| \le (4 + \sqrt{2})\sqrt{q_0}\, 2^{-k/4} + 2q_1\sqrt{P_C}$. Here

$$P_C = \Pr[x = x' : H \xleftarrow{\$} (M \to N), A_0^H(), x \leftarrow A_C(), B \xleftarrow{\$} N, j \xleftarrow{\$} \{1, \dots, q_{12}\}, x' \leftarrow C_1^H(j, B, x)].$$

By construction, C_1^H uses x only as an input to the simulated A_1, which in turn ignores x. Furthermore, x has min-entropy k, given A_C's initial state (and the final state of A_C is not accessed by C_1^H), thus the probability that C_1^H outputs x is at most 2^{-k}. Thus $P_C \le 2^{-k}$ and hence

$$|P_A^1 - P_A^2| \le (4 + \sqrt{2})\sqrt{q_0}\, 2^{-k/4} + 2q_1 2^{-k/2}. \qquad \square$$

4 Online-extractable NIZK proofs

4.1 Construction

In the following, we assume a sigma protocol $\Sigma = (N_{com}, N_{ch}, N_{resp}, P_\Sigma^1, P_\Sigma^2, V_\Sigma)$ for a relation R. Assume that $N_{resp} = \{0,1\}^{\ell_{resp}}$ for some ℓ_{resp}.[8] We use this sigma protocol to construct the following non-interactive proof system:

Definition 13 (Online-extractable proof system (P_{OE}, V_{OE})) *The proof system (P_{OE}, V_{OE}) is parametrized by polynomially-bounded integers t, m where m is a power of 2 with $2 \le m \le |N_{ch}|$. We use random oracles $H : \{0,1\}^* \to \{1, \dots, m\}^t$ and $G : N_{resp} \to N_{resp}$.[9] Prover and verifier are defined in Figure 1.*

Lemma 14 (Completeness) *If Σ is complete, (P_{OE}, V_{OE}) is complete.*

Proof. Since Σ is complete, $V_\Sigma(x, com_i, ch_{i,j}, resp_{i,j}) = 1$ for all i, j with overwhelming probability. Then all checks performed by V_{OE} succeed by construction of P_{OE}. $\square$

4.2 Zero-knowledge

Theorem 15 (Zero-knowledge) *Assume that Σ is HVZK, and that the response of P_Σ^2 has superlogarithmic min-entropy (given its initial state and its input ch).*[10]

Let κ' be a lower bound on the collision-entropy of the tuple $\big((com_i)_i, (ch_{i,j})_{i,j}, (h_{i,j})_{i,j}\big)$ produced by P_{OE} (given its initial state and the oracle G, H). Assume that κ' is superlogarithmic.[11]

Then (V_{OE}, P_{OE}) is zero-knowledge with the simulator $(S_{init}^{OE}, S_{P_{OE}})$ from Figure 2.

[8] Any N_{resp} can be efficiently embedded in a set of fixed length bitstrings $\{0,1\}^{\ell_{resp}}$ (there is no need for this embedding to be surjective). So any sigma protocol can be transformed to have $N_{resp} = \{0,1\}^{\ell_{resp}}$ for some ℓ_{resp}.

[9] The definitions from Section 2.1 are formulated with respect to only a single random oracle with distribution ROdist. Having two oracles, however, can be encoded in that framework by letting ROdist be the uniform distribution over pairs of functions with the respective domains/ranges.

[10] We can always transform a sigma protocol into one with responses with superlogarithmic min-entropy by adding some random bits to the responses.

[11] This can always be achieved by adding random bits to the commitments.

$\boldsymbol{S_{P_{OE}}}$:

> **Input**: x
>
> **for** $i = 1$ **to** t **do**
> - $J_i \xleftarrow{\$} \{1,\dots,m\}$; $(com_i, ch_{i,J_i}, resp_{i,J_i}) \leftarrow S_\Sigma(x)$
> - **for** $j = 1$ **to** m **except** $j = J_i$ **do**
>   - $ch_{i,j} \xleftarrow{\$} N_{ch} \setminus \{ch_{i,J_i}, ch_{i,1}, \dots, ch_{i,j-1}\}$
>
> **for** $i = 1$ **to** t **do**
> - $h_{i,J_i} := G(resp_{i,J_i})$
> - **for** $j = 1$ **to** m **except** $j = J_i$ **do**
>   - $h_{i,j} \xleftarrow{\$} N_{resp}$
>
> $H(x, (com_i)_i, (ch_{i,j})_{i,j}, (h_{i,j})_{i,j}) := J_1 \| \dots \| J_t$
> **return** $\pi := \big((com_i)_i, (ch_{i,j})_{i,j}, (h_{i,j})_{i,j}, (resp_{i,J_i})_i\big)$

$\boldsymbol{S^{OE}_{init}}$:

> **Parameters**: upper bounds $\boldsymbol{q_G, q_H}$ on the number of queries to G and H; upper bound $\boldsymbol{\ell}$ on the length of the inputs to H; embedding $\boldsymbol{\iota_\ell}$
>
> $p_G \xleftarrow{\$} \mathrm{GF}(2^{\ell_{resp}})[X]$ with $\partial p_G \le 2q_G - 1$
> $p_H \xleftarrow{\$} \mathrm{GF}(2^{\ell^*})[X]$ with $\partial p_H \le 2q_H - 1$
>
> `// Construct circuits` G, H:
> $G(x) := p_G(x)$ **for** $x \in \{0,1\}^{\ell_{resp}}$
> $H(x) := p_H(\iota_\ell(x))_{1\dots t\log m}$ **for** $x \in \{0,1\}^{\le \ell}$
>
> **return** descriptions of G, H

S_Σ is the simulator for $(P^1_\Sigma, P^2_\Sigma, V_\Sigma)$, cf. Definition 5. $H(x) := y$ means the description of H is replaced by a new description with $H(x) = y$. Bounds q_G, q_H, ℓ include calls made by the adversary and by P_{OE}. Such bounds are known because the runtime of A is known to the simulator (cf. Definition 2). ι_ℓ is an arbitrary efficiently computable and invertible injection $\iota_\ell : \{0,1\}^{\le \ell} \to \{0,1\}^{\ell^*}$ for some $\ell^* \ge t \log m$. $p_H(\iota_\ell(x))_{1\dots t\log m}$ denotes $p_H(\iota_\ell(x))_{1\dots t\log m}$ truncated to the first $t \log m$ bits. We assume that $\mathrm{GF}(2^{\ell_{resp}}) = \{0,1\}^{\ell_{resp}}$ and $\mathrm{GF}(2^{\ell^*}) = \{0,1\}^{\ell^*}$; such a representation can be found in polynomial-time [BO81].

Figure 2: The simulator $(S_{P_{OE}}, S^{OE}_{init})$ for (P_{OE}, V_{OE}).

Proof. We prove this using a sequence of games. We start with the real model (lhs of (1)) and transform it into the ideal model (rhs of (1)) step by step, never changing $\Pr[b = 1]$ by more than a negligible amount. In each game, new code lines are marked with $\boxed{\text{new}}$ and changed ones with $\boxed{\text{chg}}$ (removed ones are simply crossed out).

Let ROdist be the uniform distribution on pairs of functions G, H (with the respective domains and ranges as in Definition 13). Then the lhs of (1) becomes:

Game 1 (Real model) $G, H \xleftarrow{\$} \mathsf{ROdist}, b \leftarrow A^{G,H,P_{OE}}$.

We now modify the prover. Instead of getting $J_1, \dots, J_t$ from the random oracle H, he chooses $J_1, \dots, J_t$ at random and programs the random oracle H to return those values $J_1, \dots, J_t$.

Game 2 $G, H \xleftarrow{\$} \mathsf{ROdist}, b \leftarrow A^{G,H,P}$ *with the following prover P:*

> $\vdots$
> **for** $i = 1$ **to** t **do**
> $\boxed{\textbf{new}}$ $J_i \leftarrow \{1,\dots,m\}$
> $com_i \leftarrow P^1_\Sigma(x, w)$
> $\vdots$
> $\cancel{J_1\|\dots\|J_t := H(x, (com_i)_i, (ch_{i,j})_{i,j}, (h_{i,j})_{i,j})}$
> $\boxed{\textbf{chg}}$ $H(x, (com_i)_i, (ch_{i,j})_{i,j}, (h_{i,j})_{i,j}) := J_1 \| \dots \| J_t$
> $\vdots$

By assumption the argument $(x, (com_i)_i, (ch_{i,j})_{i,j}, (h_{i,j})_{i,j})$ to H has superlogarithmic collision-entropy κ' (given the state at the beginning of the corresponding invocation of P_{OE}). Thus from Corollary 11 we get (using a standard hybrid argument) that $\big|\Pr[b = 1 : \text{Game 1}] - \Pr[b = 1 : \text{Game 2}]\big|$ is negligible.

Next, we change the order in which the prover produces the subproofs $(com_i, ch_{i,j}, resp_{i,j})$: For each i, the $(com_i, ch_{i,j}, resp_{i,j})$ with $j = J_i$ is produced first.

Game 3 $G, H \xleftarrow{\$} \mathsf{ROdist}, b \leftarrow A^{G,H,P}$ *with the* P *as follows:*

$$
\begin{array}{ll}
 & \vdots \\
 & \textbf{for } i = 1 \textbf{ to } t \textbf{ do} \\
 & \quad J_i \leftarrow \{1, \dots, m\};\ com_i \leftarrow P^1_\Sigma(x, w) \\
\boxed{\textbf{new}} & \quad ch_{i,J_i} \xleftarrow{\$} N_{ch};\ resp_{i,J_i} \leftarrow P^2_\Sigma(ch_{i,J_i}) \\
\boxed{\textbf{chg}} & \quad \textbf{for } j = 1 \textbf{ to } m \textbf{ except } j = J_i \textbf{ do} \\
\boxed{\textbf{chg}} & \quad\quad ch_{i,j} \xleftarrow{\$} N_{ch} \setminus \{ch_{i,J_i}, ch_{i,1}, \dots, ch_{i,j-1}\} \\
 & \quad\quad resp_{i,j} \leftarrow P^2_\Sigma(ch_{i,j}) \\
 & \vdots
\end{array}
$$

Obviously, changing the order of the P^2_Σ-invocations does not change anything because P^2_Σ has no side effects. At a first glance, it seems that the values $ch_{i,j}$ are chosen according to different distributions in both games, but in fact in both games $(ch_{i,1}, \dots, ch_{i,m})$ are uniformly distributed conditioned on being pairwise distinct. Thus $\Pr[b = 1 : \text{Game 2}] = \Pr[b = 1 : \text{Game 3}]$.

Now we change how the $h_{i,j}$ are constructed. Those $h_{i,j}$ that are never opened are picked at random.

Game 4 $G, H \xleftarrow{\$} \mathsf{ROdist}, b \leftarrow A^{G,H,P}$ *with the* P *as follows:*

$$
\begin{array}{ll}
 & \vdots \\
 & \textbf{for } i = 1 \textbf{ to } t \textbf{ do} \\
\boxed{\textbf{new}} & \quad h_{i,J_i} := G(resp_{i,J_i}) \\
\boxed{\textbf{chg}} & \quad \textbf{for } j = 1 \textbf{ to } m \textbf{ except } j = J_i \textbf{ do} \\
\boxed{\textbf{chg}} & \quad\quad h_{i,j} \xleftarrow{\$} N_{resp} \\
 & \vdots
\end{array}
$$

Note that the argument $resp_{i,j}$ to G has superlogarithmic min-entropy (given the value of all variables when $G(resp_{i,j})$ is invoked) since we assume that the responses of P^2_Σ have superlogarithmic min-entropy. Thus from Corollary 12 we get (using a standard hybrid argument) that $\big|\Pr[b = 1 : \text{Game 3}] - \Pr[b = 1 : \text{Game 4}]\big|$ is negligible. (H in the corollary is G here, and A_C in the corollary is P^2_Σ here.)

Now we omit the computation of the values $resp_{i,j}$ that are not used:

Game 5 $G, H \xleftarrow{\$} \mathsf{ROdist}, b \leftarrow A^{G,H,P}$ *with the* P *as follows:*

$$
\begin{array}{l}
\vdots \\
\quad \textbf{for } j = 1 \textbf{ to } m \textbf{ except } j = J_i \textbf{ do} \\
\quad\quad ch_{i,j} \xleftarrow{\$} N_{ch} \setminus \{ch_{i,J_i}, ch_{i,1}, \dots, ch_{i,j-1}\} \\
\quad\quad \cancel{resp_{i,j} \leftarrow P^2_\Sigma(ch_{i,j})} \\
\vdots
\end{array}
$$

We now replace the honestly generated proof $(com_i, ch_{i,J_i}, resp_{i,J_i})$ by one produced by the simulator S_Σ (from Definition 5).

Game 6 $G, H \xleftarrow{\$} \mathsf{ROdist}, b \leftarrow A^{G,H,P}$ *with the* P *as follows:*

$$
\begin{array}{ll}
 & \vdots \\
 & \textbf{for } i = 1 \textbf{ to } t \textbf{ do} \\
 & \quad J_i \leftarrow \{1, \dots, m\};\ \cancel{com_i \leftarrow P^1_\Sigma(x, w)} \\
 & \quad \cancel{ch_{i,J_i} \xleftarrow{\$} N_{ch};\ resp_{i,J_i} \leftarrow P^2_\Sigma(ch_{i,J_i})} \\
\boxed{\textbf{new}} & \quad (com_i, ch_{i,J_i}, resp_{i,J_i}) \leftarrow S_\Sigma(x) \\
 & \vdots
\end{array}
$$

$\boldsymbol{E_{P_{OE}}}$:

Input: $G = p_G, H, x, \pi = \big((com_i), (ch_{i,j}), (h_{i,j}), (resp_i)\big)$
$J_1\\|\ldots\\|J_t := H(x, (com_i)_i, (ch_{i,j})_{i,j}, (h_{i,j})_{i,j})$
for $i = 1$ **to** t **do**
for $j = 1$ **to** m **except** J_i **do**
for each $resp' \in p_G^{-1}(h_{i,j})$ **do**
if $V_\Sigma(x, com_i, ch_{i,j}, resp') = 1$ **then**
return $E_\Sigma(x, com_i, ch_{i,J_i}, resp_i, ch_{i,j}, resp')$

V_Σ and E_Σ are verifier and extractor of the sigma protocol Σ. $p_G^{-1}(h)$ is the set of preimages of h under p_G. Since p_G is a polynomial over $\mathrm{GF}(2^{\ell_{resp}})$, $p_G^{-1}(h)$ is polynomial-time computable [BO81].

Figure 3: The extractor $E_{P_{OE}}$ for (P_{OE}, V_{OE}).

Since Σ is HVZK by assumption, $\big|\Pr[b = 1 : \text{Game 5}] - \Pr[b = 1 : \text{Game 6}]\big|$ is negligible.

Note that P as defined in Game 6 does not use the witness w any more. Thus we can replace P by a simulator that depends only on the statement x. That simulator $S_{P_{OE}}$ is given in Figure 2.

Game 7 $G, H \xleftarrow{\$} \mathsf{ROdist}, b \leftarrow A^{G,H,S'_{P_{OE}}}$ *for* $S_{P_{OE}}$ *from Figure 2. (Recall that* $S'_{P_{OE}}$ *is defined in terms of* $S_{P_{OE}}$*, see Definition 2.)*

From the definition of $S_{P_{OE}}$ in Figure 2 we immediately get $\Pr[b = 1 : \text{Game 6}] = \Pr[b = 1 : \text{Game 7}]$.

Finally, we replace ROdist by oracles as chosen by S^{OE}_{init} from Figure 2. (In general, any construction of S^{OE}_{init} would do for the proof of the zero-knowledge property, as long as it returns G, H that are indistinguishable from random. However, in the proof of extractability we use that G is constructed in this specific way.)

Game 8 $G, H \xleftarrow{\$} S^{OE}_{init}, b \leftarrow A^{G,H,S'_{P_{OE}}}$ *for* $(S^{OE}_{init}, S_{P_{OE}})$ *from Figure 2.*

For the following argument, we introduce the following abbreviation: Given distributions on functions H, H', by $H \approx_{q,\ell} H'$ we denote that H and H' are perfectly indistinguishable by any quantum algorithm making at most q queries and making no queries with input longer than ℓ. We omit q or ℓ if $q = \infty$ or $\ell = \infty$. Let $p_G, p_H, \ell, \iota_\ell, \ell^*$ be as defined in Figure 2.

Let G_R denote the function $G : N_{resp} \to N_{resp}$ as chosen by ROdist, and let G_S denote the function $G = p_G$ chosen by S^{OE}_{init}. It is easy to see that a uniformly random polynomial p of degree $\le 2q - 1$ is $2q$-wise independent. [Zha12] shows that a $2q$-wise independent function is perfectly indistinguishable from a random function by an adversary performing at most q queries (the queries may be in superposition). Then $G_R \approx_{q_G} G_S$.

Similarly, let H_R and H_S denote $H : \{0,1\}^* \to \{0,1\}^{t \log m}$ as chosen by ROdist or S^{OE}_{init}, respectively. Then $p_H \approx_{2q_H} H'$ for a uniformly random function $H' : \{0,1\}^{\ell^*} \to \{0,1\}^{\ell^*}$. Hence $p_H \circ \iota_\ell \approx_{q_H} H' \circ \iota_\ell \approx H''$ for uniformly random $H'' : \{0,1\}^{\le \ell} \to \{0,1\}^{\ell^*}$. Hence $H_S = (p_H \circ \iota_\ell)_{1\ldots t \log m} \approx_{q_H} (H'')_{1\ldots t \log m}$ where $H_{1\ldots t \log m}$ means H with its output restricted to the first $t \log m$ bits.[12] And $H'' \approx_\ell H_3$ for uniformly random $H_3 : \{0,1\}^* \to \{0,1\}^{\ell^*}$. Thus $H_S \approx_{q_H} (H'')_{1\ldots t \log m} \approx_\ell (H_3)_{1\ldots t \log m} \approx H_R$, hence $H_S \approx_{q_H,\ell} H_R$.

Since q_H and q_G are upper bounds on the number of queries to H and G and ℓ bounds input length of the H-queries made by A, $G_R \approx_{q_G} G_S$ and $H_S \approx_{q_H,\ell} H_R$ imply that A cannot distinguish the oracles G_R, H_R produced by ROdist from the oracles G_S, H_S produced by S^{OE}_{init}. Thus $\Pr[b = 1 : \text{Game 7}] = \Pr[b = 1 : \text{Game 8}]$.

Summarizing, we have that $\big|\Pr[b = 1 : \text{Game 1}] - \Pr[b = 1 : \text{Game 8}]\big|$ is negligible. Since Games 1 and 8 are the games in (1), it follows that (P_{OE}, V_{OE}) is zero-knowledge. □

4.3 Online extractability

We now proceed to prove that (P_{OE}, V_{OE}) is simulation-sound online-extractable using the extractor $E_{P_{OE}}$ from Figure 3.

To analyze $E_{P_{OE}}$, we define a number of random variables and events that can occur in the execution of the simulation-soundness game (Definition 4). Remember, the game in question is $G, H \leftarrow S^{OE}_{init}, (x, \pi) \leftarrow$

[12] Notice that to see this, we need to be able to implement $(H'')_{1\ldots t \log m}$ using a single oracle query to H''. This can be done by initializing the output qubits of H'' that shall be ignored with $|+\rangle$, see [Zha13, Section 3.2].

$A^{G,H,S_{P_{OE}}}$, $ok \leftarrow V_{OE}^{G,H}(x,\pi)$, $w \leftarrow E_{P_{OE}}(H,x,\pi)$, and *simproofs* is the set of all proofs returned by $S_{P_{OE}}$ (together with the corresponding statements).

- H_0: Let H_0 denote the initial value of H as returned by S_{init}^{OE}. (H can change during the game because $S_{P_{OE}}$ programs it, see Figure 2. On the other hand, G does not change.)
- H_1: Let H_1 denote to the final value of H (as used by V_{OE} for computing ok).
- $\mathsf{ShouldEx}$: $ok = 1$ and $(x,\pi) \notin \mathit{simproofs}$. (I.e., in this case the extractor should find a witness.)
- ExFail: $ok = 1$ and $(x,\pi) \notin \mathit{simproofs}$ and $(x,w) \notin R$. (ExFail represents a successful attack.)
- $\mathsf{MallSim}$: $ok = 1$ and $(x,\pi) \notin \mathit{simproofs}$ and $(x,\pi^*) \in \mathit{simproofs}$ for some $\pi^* = \big((com_i)_i,(ch_{i,j})_{i,j},(h_{i,j})_{i,j},(resp_i^*)_i\big)$ where $\big((com_i)_i,(ch_{i,j})_{i,j},(h_{i,j})_{i,j},(resp_i)_i\big) := \pi$. (In other words, the adversary produces a valid proof that differs from one of the simulator generated proofs (for the same statement x) only in the *resp*-components).
- We call a triple $(com, ch, resp)$ Σ-valid iff $V_\Sigma(x, com, ch, resp) = 1$ (x will always be clear from the context). If R is a set, we call (com, ch, R) set-valid iff there is a $resp \in R$ such that $(com, ch, resp)$ is Σ-valid. And Σ-invalid and set-invalid are the negations of Σ-valid and set-valid.

The following technical lemma establishes that an adversary with access to the simulator $S_{P_{OE}}$ cannot produce a new valid proof by just changing the *resp*-components of a simulated proof. This will cover one of the attack scenarios covered in the proof of simulation-sound online-extractability below.

Lemma 16 (Non-malleability) *Let κ be a lower bound on the collision-entropy of the tuple $\big((com_i)_i,(ch_{i,j})_{i,j},(h_{i,j})_{i,j}\big)$ produced by $S_{P_{OE}}$ (given its initial state and the oracle G,H). Let q_G be an upper bound for the number of queries to G made by A and $S_{P_{OE}}$ and V_{OE} together. Let n be an upper bound on the number of invocations of $S_{P_{OE}}$.*

Then $\Pr[\mathsf{MallSim}] \le \frac{n(n+1)}{2}2^{-\kappa} + O\big((q_G+1)^3 2^{-\ell_{resp}}\big)$.

Proof. First, since G is chosen as a polynomial of degree $2q_G - 1$ and is thus $2q_G$-wise independent, by [Zha12] G is perfectly indistinguishable from a uniformly random G within q_G queries. Thus, for the proof of this lemma, we can assume that G is a uniformly random function.

In the definition of $\mathsf{MallSim}$, since $ok = 1$, we have that π is accepted by V_{OE}. In particular, this implies that $G(resp_i) = h_{i,J_i}$ for all i by definition of V_{OE}. And $J_1\|\dots\|J_t = H_1(x,\pi_{half})$ where $\pi_{half} := \big((com_i)_i,(ch_{i,j})_{i,j},(h_{i,j})_{i,j}\big)$ is π without the *resp*-components. Furthermore, by construction of $S_{P_{OE}}$, we have that π^* satisfies: $G(resp_i^*) = h_{i,J_i^*}$ for all i and $J_1^*\|\dots\|J_t^* = H^*(x,\pi_{half})$ where H^* denotes the value of H directly after $S_{P_{OE}}$ output π^*. (I.e., H^* might differ from H_1 if further invocations of $S_{P_{OE}}$ programmed H further.) But if $H_1(x,\pi_{half}) = H^*(x,\pi_{half})$, then $J_i = J_i^*$ for all i, and thus $G(resp_i) = G(resp_i^*)$ for all i. And since $\pi \notin \mathit{simproofs}$ and $\pi^* \in \mathit{simproofs}$ by definition of $\mathsf{MallSim}$, we have that $resp_i \neq resp_i^*$ for some i.

Thus

$$\Pr[\mathsf{MallSim}] \le \Pr[H_1(x,\pi_{half}) \neq H^*(x,\pi_{half})] + \Pr[\exists i : G(resp_i) = G(resp_i^*) \wedge resp_i \neq resp_i^*]$$

If we have $H_1(x,\pi_{half}) \neq H^*(x,\pi_{half})$, this implies that $S_{P_{OE}}$ reprogrammed H after producing π^*. This implies in particular that in two invocations of $S_{P_{OE}}$, the same tuple $\pi_{half} = \big((com_i)_i,(ch_{i,j})_{i,j},(h_{i,j})_{i,j}\big)$ was chosen. This happens with probability at most $\frac{n(n+1)}{2}2^{-\kappa}$ because each such tuple has collision-entropy at least κ.

Finally, since G is a random function that is queried at most q_G times, $\Pr[\exists i : G(resp_i) = G(resp_i^*) \wedge resp_i \neq resp_i^*] \in O\big((q_G+1)^3 2^{-\ell_{resp}}\big)$ by Theorem 8.

Thus $\Pr[\mathsf{MallSim}] \le \frac{n(n+1)}{2}2^{-\kappa} + O\big((q_G+1)^3 2^{-\ell_{resp}}\big)$. □

The following lemma states that, if H is uniformly random, the adversary cannot produce a valid proof (conditions (i),(ii)) from which is it not possible to extract a second response for one of the com_i by inverting G (condition (iii)). This lemma already implies online-extractability, because it implies that the extractor $E_{P_{OE}}$ will get a commitment com_i with two valid responses. However, it does not go the full way to showing simulation-sound online-extractability yet, because in that setting, the adversary has access to $S_{P_{OE}}$ which reprograms the random oracle H, so H cannot be treated as a random function.

Lemma 17 *Let G be an arbitrarily distributed function, and let $H_0 : \{0,1\}^{\le \ell} \to \{0,1\}^{t \log m}$ be uniformly random (and independent of G). Then it is hard to find x and $\pi = \big((com_i),(ch_{i,j}),(h_{i,j}),(resp_i)\big)$ such that:*

(i) $h_{i,J_i} = G(resp_i)$ for all i with $J_1\|\dots\|J_t := H_0(x,(com_i)_i,(ch_{i,j})_{i,j},(h_{i,j})_{i,j})$.

(ii) $(com_i, ch_{i,J_i}, resp_i)$ is Σ-valid for all i.

(iii) $(com_i, ch_{i,j}, G^{-1}(h_{i,j}))$ *is set-invalid for all* i *and* j *with* $j \neq J_i$.
More precisely, if A^{G,H_0} *makes at most* q_H *queries to* H_0*, it outputs* (x,π) *with these properties with probability at most* $2(q_H+1)2^{-(t\log m)/2}$.

Proof. Without loss of generality, we can assume that G is a fixed function and that A knows that function. Thus in the following, we only provide oracle access to H_0 to A.

For any given value of H_0, we call a tuple $\big(x,(com_i),(ch_{i,j}),(h_{i,j})\big)$ an H_0*-solution* iff:

for each i,j, we have that $(com_i, ch_{i,j}, G^{-1}(h_{i,j}))$ is set-valid iff $j = J_i$
where $J_1\|\dots\|J_t := H_0(x,(com_i)_i,(ch_{i,j})_{i,j},(h_{i,j})_{i,j})$.

(The name "H_0-solution" derives from the fact that we are trying to solve an equation in terms of H_0.)

It is easy to see that if x and $\pi = \big((com_i),(ch_{i,j}),(h_{i,j}),(resp_i)\big)$ satisfies (i)–(iii), then $\big(x,(com_i),(ch_{i,j}),(h_{i,j})\big)$ is an H_0-solution. (Note for the case $j = J_i$ that $h_{i,J_i} = G(resp_i)$ implies $resp_i \in G^{-1}(h_{i,j})$. With the Σ-validity of $(com_i, ch_{i,J_i}, resp_i)$ this implies the set-validity of $(com_i, ch_{i,j}, G^{-1}(h_{i,j}))$.)

Thus it is sufficient to prove that $A_0^H()$ making at most q_H queries outputs an H_0-solution with probability at most $2(q_H+1)2^{-(t\log m)/2}$. Fix such an adversary A_0^H; denote the probability that it outputs an H_0-solution (for uniformly random H_0) with μ.

We call $\big(x,(com_i),(ch_{i,j}),(h_{i,j})\big)$ an *candidate* iff for each i, there is exactly one J_i^* such that $(com_i, ch_{i,J_i^*}, G^{-1}(h_{i,J_i^*}))$ is set-valid. Notice that a non-candidate can never be an H_0-solution. (This justifies the name "candidate", those are candidates for being an H_0-solution, awaiting a test-call to H_0.)

For any given candidate c, for uniformly random H_0, the probability that c is an H_0-solution is $2^{-t\log m}$. (Namely c is an H_0-solution iff all $J_i = J_i^*$ for all i, i.e., there is exactly one output of $H_0(c) \in \{0,1\}^{t\log m}$ that makes c an H_0-solution.)

Let Cand denote the set of all candidates. Let $F : \mathsf{Cand} \to \{0,1\}$ be a random function with all $F(c)$ independently identically distributed with $\Pr[F(c)=1] = 2^{-t\log m}$.

Given F, we construct $H_F : \{0,1\}^* \to \{0,1\}^{t\log m}$ as follows:

- For each $c \notin \mathsf{Cand}$, assign a uniformly random $y \in \{0,1\}^{t\log m}$ to $H_F(c)$.
- For each $c \in \mathsf{Cand}$ with $F(c) = 1$, let $H_F(c) := J_1^*\|\dots\|J_t^*$ where $J_1^*,\dots,J_t^*$ are as in the definition of candidates.
- For each $c \in \mathsf{Cand}$ with $F(c) = 0$, assign a uniformly random $y \in \{0,1\}^{t\log m} \setminus \{J_1^*\|\dots\|J_t^*\}$ to $H_F(c)$.

Since $F(c) = 1$ with probability $2^{-t\log m}$, $H_F(c)$ is uniformly distributed over $\{0,1\}^{t\log m}$ for $c \in \mathsf{Cand}$. Thus H_F is a uniformly random function.

Since $A_0^H()$ outputs an H_0-solution with probability μ and H_F has the same distribution as H_0, $A^{H_F}()$ outputs an H_F-solution c with probability μ. Since an H_F-solution c must be a candidate, we have $c \in \mathsf{Cand}$. And c can only be an H_F-solution if $H_F(c) = J_1^*\|\dots\|J_t^*$, i.e., if $F(c) = 1$. Thus $A^{H_F}()$ returns some c with $F(c) = 1$ with probability μ.

However, to explicitly construct H_F, A^{H_F} needs to query all values of F, so the number of F-queries is not bounded by q_H. However, A^{H_F} can be simulated by the following algorithm S^F:

- Pick uniformly random $H_1 : \{0,1\}^{\leq \ell} \to \{0,1\}^{t\log m}$. Set $H_2(c) := J_1^*\|\dots\|J_t^*$ for all $c \in \mathsf{Cand}$. For all $c \in \mathsf{Cand}$, let $H_3(c) := y$ for uniformly random $y \in \{0,1\}^{t\log m} \setminus \{J_1^*\|\dots\|J_t^*\}$.
- Let $H_F'(c) := H_1(c)$ if $c \notin \mathsf{Cand}$, let $H_F'(c) := H_2(c)$ if $c \in \mathsf{Cand}$ and $F(c) = 1$, let $H_F'(c) := H_3(c)$ if $c \in \mathsf{Cand}$ and $F(c) = 0$.
- Run $A^{H_F'}()$.

The function H_F' constructed by S has the same distribution as H_F (given the same F). Thus S outputs c with $F(c) = 1$ with probability μ. Furthermore, no F-queries are needed to construct H_1, H_2, H_3, and a single F-query is needed for each H_F'-query performed by A^{H_F}. Thus S performs at most q_H F-queries. Thus by Lemma 7, $\mu \leq 2(q_H+1)2^{-(t\log m)/2}$. □

Theorem 18 (Simulation-sound online-extractability) *Assume that* Σ *has special soundness. Let* κ *be a lower bound on the collision-entropy of the tuple* $\big((com_i)_i,(ch_{i,j})_{i,j},(h_{i,j})_{i,j}\big)$ *produced by* $S_{P_{OE}}$ *(given its input and the oracles* G, H*). Assume that* $t\log m$ *and* κ *and* ℓ_{resp} *are superlogarithmic.*

Then (V_{OE}, P_{OE}) *is simulation-sound online-extractable with extractor* $E_{P_{OE}}$ *from Figure 3 and with respect to the simulator* $(S_{P_{OE}}, S_{init}^{OE})$ *from Figure 2.*

A concrete bound μ *on the success probability is given in* (6) *below.*

Proof. Given $\pi = \big((com_i)_i,(ch_{i,j})_{i,j},(h_{i,j})_{i,j},(resp_i)_i\big)$, let $\pi_{half} := \big((com_i)_i,(ch_{i,j})_{i,j},(h_{i,j})_{i,j}\big)$, i.e., π without the *resp*-components.

Fix an adversary A for the game in Definition 4. Assume A, $S_{P_{OE}}$, V_{OE} together perform at most q_G queries to G and q_H queries to H, and that at most n instances of $S_{P_{OE}}$ are invoked.

Let $\mathsf{Ev}_{\text{(i)}}$, $\mathsf{Ev}_{\text{(ii)}}$, $\mathsf{Ev}_{\text{(iii)}}$ denote the events that conditions (i), (ii), (iii) from Lemma 17 are satisfied.

Assume that $\mathsf{ShouldEx} \wedge \neg\mathsf{MallSim} \wedge \neg\mathsf{Ev}_{\text{(iii)}}$ occurs. The event $\mathsf{ShouldEx}$ by definition entails $ok = 1$ and $(x, \pi) \notin \mathit{simproofs}$. Furthermore, $\neg\mathsf{MallSim}$ then implies that for all $(x^*, \pi^*) \in \mathit{simproofs}$, we have that $(x^*, \pi^*_{half}) \neq (x, \pi_{half})$. In an invocation $\pi^* \leftarrow S_{P_{OE}}(x^*)$, $S_{P_{OE}}$ only reprograms H at position $H(x^*, \pi^*_{half})$, hence $H(x, \pi_{half})$ is never reprogrammed. Thus $H_0(x, \pi_{half}) = H_1(x, \pi_{half})$. Furthermore $ok = 1$ implies by definition of V_{OE} (and the fact that H_1 denotes H at the time of invocation of V_{OE}): $(com_i, ch_{i,J_i}, resp_i)$ is Σ-valid for all i and $h_{i,J_i} = G(resp_i)$ for all i, where $J_1\|\dots\|J_t := H_1(x, \pi_{half})$. Since $H_0(x, \pi_{half}) = H_1(x, \pi_{half})$, we have $J_1\|\dots\|J_t = H_0(x, \pi_{half})$ as well. And $\neg\mathsf{Ev}_{\text{(iii)}}$ implies that $(com_i, ch_{i,j}, G^{-1}(h_{i,j}))$ is set-valid for some i, j with $j \neq J_i$. Thus by construction, $E_{P_{OE}}$ outputs $w := E_\Sigma(x, com_i, ch_{i,J_i}, resp_i, ch_{i,j}, resp')$ for some $resp' \in G^{-1}(h_{i,j})$ such that $(com_i, ch_{i,j}, resp')$ is Σ-valid. Furthermore, $ok = 1$ implies by definition of V_{OE} that $ch_{i,1}, \dots, ch_{i,t}$ are pairwise distinct, in particular $ch_{i,j} \neq ch_{i,J_i}$. And $ok = 1$ implies that $(com_i, ch_{i,J_i}, resp_i)$ is Σ-valid. On such inputs, the special soundness of E_Σ (cf. Definition 5) implies that $(x, w) \in R$ with probability at least $1 - \varepsilon_{sound}$ for negligible ε_{sound}. Thus

$$\Pr[\mathsf{ShouldEx} \wedge (x,w) \in R \wedge \neg\mathsf{MallSim} \wedge \neg\mathsf{Ev}_{\text{(iii)}}] \geq \Pr[\mathsf{ShouldEx} \wedge \neg\mathsf{MallSim} \wedge \neg\mathsf{Ev}_{\text{(iii)}}] - \varepsilon_{sound}. \tag{2}$$

Then since $\mathsf{ExFail} \iff \mathsf{ShouldEx} \wedge (x, w) \notin R$,

$$\begin{aligned}
&\Pr[\mathsf{ExFail} \wedge \neg\mathsf{MallSim} \wedge \neg\mathsf{Ev}_{\text{(iii)}}] \\
&= \Pr[\mathsf{ShouldEx} \wedge \neg\mathsf{MallSim} \wedge \neg\mathsf{Ev}_{\text{(iii)}}] - \Pr[\mathsf{ShouldEx} \wedge (x,w) \in R \wedge \neg\mathsf{MallSim} \wedge \neg\mathsf{Ev}_{\text{(iii)}}] \\
&\overset{(2)}{\leq} \varepsilon_{sound}.
\end{aligned} \tag{3}$$

Then

$$\begin{aligned}
&\Pr[\mathsf{ExFail} \wedge \neg\mathsf{MallSim}] \\
&= \Pr[\mathsf{ExFail} \wedge \neg\mathsf{MallSim} \wedge \mathsf{Ev}_{\text{(iii)}}] + \Pr[\mathsf{ExFail} \wedge \neg\mathsf{MallSim} \wedge \neg\mathsf{Ev}_{\text{(iii)}}] \\
&\overset{(3)}{\leq} \Pr[\mathsf{ExFail} \wedge \neg\mathsf{MallSim} \wedge \mathsf{Ev}_{\text{(iii)}}] + \varepsilon_{sound}.
\end{aligned} \tag{4}$$

Assume $\mathsf{ExFail} \wedge \neg\mathsf{MallSim}$. As seen above (in the case $\mathsf{ShouldEx} \wedge \neg\mathsf{MallSim} \wedge \neg\mathsf{Ev}_{\text{(iii)}}$), this implies that $H_0(x, \pi_{half}) = H_1(x, \pi_{half})$ and that $(com_i, ch_{i,J_i}, resp_i)$ is Σ-valid for all i and $h_{i,J_i} = G(resp_i)$ for all i, where $J_1\|\dots\|J_t := H_1(x, \pi_{half})$. This immediately implies $\mathsf{Ev}_{\text{(i)}}$ and $\mathsf{Ev}_{\text{(ii)}}$. Thus

$$\begin{aligned}
\Pr[\mathsf{ExFail} \wedge \neg\mathsf{MallSim}] &\overset{(4)}{\leq} \Pr[\mathsf{ExFail} \wedge \neg\mathsf{MallSim} \wedge \mathsf{Ev}_{\text{(iii)}}] + \varepsilon_{sound} \\
&\overset{(*)}{=} \Pr[\mathsf{ExFail} \wedge \neg\mathsf{MallSim} \wedge \mathsf{Ev}_{\text{(i)}} \wedge \mathsf{Ev}_{\text{(ii)}} \wedge \mathsf{Ev}_{\text{(iii)}}] + \varepsilon_{sound} \\
&\leq \Pr[\mathsf{Ev}_{\text{(i)}} \wedge \mathsf{Ev}_{\text{(ii)}} \wedge \mathsf{Ev}_{\text{(iii)}}] + \varepsilon_{sound}
\end{aligned} \tag{5}$$

where $(*)$ uses $\mathsf{ExFail} \wedge \neg\mathsf{MallSim} \Rightarrow \mathsf{Ev}_{\text{(i)}} \wedge \mathsf{Ev}_{\text{(ii)}}$.

As already seen in the proof of Theorem 15, $H = H_0$ as chosen by S^{OE}_{init} is perfectly indistinguishable from a uniformly random $H_0 : \{0,1\}^{\leq \ell} \to \{0,1\}^{t \log m}$ using only q_H queries. Thus we can apply Lemma 17, and get $\Pr[\mathsf{Ev}_{\text{(i)}} \wedge \mathsf{Ev}_{\text{(ii)}} \wedge \mathsf{Ev}_{\text{(iii)}}] \leq 2(q_H + 1)2^{-(t \log m)/2}$.

And by Lemma 16, we have $\Pr[\mathsf{MallSim}] \leq \frac{n(n+1)}{2} 2^{-\kappa} + O\big((q_G + 1)^3 2^{-\ell_{resp}}\big)$. We have

$$\begin{aligned}
&\Pr[\mathsf{ExFail}] \leq \Pr[\mathsf{ExFail} \wedge \neg\mathsf{MallSim}] + \Pr[\mathsf{MallSim}] \\
&\overset{(5)}{\leq} \Pr[\mathsf{Ev}_{\text{(i)}} \wedge \mathsf{Ev}_{\text{(ii)}} \wedge \mathsf{Ev}_{\text{(iii)}}] + \varepsilon_{sound} + \Pr[\mathsf{MallSim}] \\
&\leq 2(q_H + 1)2^{-(t \log m)/2} + \varepsilon_{sound} + \frac{n(n+1)}{2} 2^{-\kappa} + O\big((q_G + 1)^3 2^{-\ell_{resp}}\big) =: \mu.
\end{aligned} \tag{6}$$

Since the adversary A is polynomial-time, q_H, q_G, n are polynomially-bounded. Furthermore $t \log m$ and κ and ℓ_{resp} are superlogarithmic by assumption. Thus μ is negligible. And since ExFail is the probability that the adversary wins in Definition 4, it follows that (P_{OE}, V_{OE}) is simulation-sound online-extractable.
□

Corollary 19 *If there is a sigma-protocol Σ that is complete and HVZK and has special soundness, then there exists a non-interactive zero-knowledge proof system with simulation-sound online extractability in the random oracle model.*

Proof. Without loss of generality, we can assume that the commitments and the responses of Σ have at least superlogarithmic collision-entropy κ'. (This can always be achieved without losing completeness, HVZK, or special soundness by adding κ' random bits to the commitments and the responses of Σ.) This also implies that ℓ_{resp} is superlogarithmic. And it implies that the tuples $\big((com_i)_i, (ch_{i,j})_{i,j}, (h_{i,j})_{i,j}, (resp_i)_i\big)$ produced by P_{OE} have superlogarithmic collision-entropy $\geq \kappa'$.

Fix polynomially-bounded t, m such that m is a power of two with $2 \leq m \leq |N_{resp}|$ and such that $t \log m$ is superlogarithmic. (E.g., t superlogarithmic and $m = 2$.) and let (V_{OE}, P_{OE}) be as in Definition 13 (with parameters t, m).

Then by Theorem 15, (V_{OE}, P_{OE}) is zero-knowledge.

Then the collision-entropy κ of the tuples $\big((com_i)_i, (ch_{i,j})_{i,j}, (h_{i,j})_{i,j}, (resp_i)_i\big)$ produced by $S_{P_{OE}}$ must be superlogarithmic. (Otherwise one could distinguish between P_{OE} and $S_{P_{OE}}$ by invoking it twice with the same argument and checking if they result in the same tuple.)

Then by Theorem 18, (V_{OE}, P_{OE}) is simulation-sound online-extractable. □

5 Signatures

A typical application of non-interactive zero-knowledge proofs of knowledge are signature schemes. E.g., the Fiat-Shamir construction [FS87] was originally described as a signature scheme. As a litmus test whether our security definitions (Definition 2 and Definition 4) are reasonable in the quantum setting, we demonstrate how to construct signatures from non-interactive simulation-sound online-extractable zero-knowledge protocols (in particular the protocol (P_{OE}, V_{OE}) from Definition 13). The construction is standard, and the proof simple, but we believe that such a sanity check for the definitions is necessary, because sometimes a definition is perfectly reasonable in the classical setting while its natural quantum counterpart is almost useless. (An example is the classical definition of "computationally binding commitments" which was shown to imply almost no security in the quantum setting [ARU14b].)

The basic idea of the construction is that to sign a message m, one needs to show that one know the knowledge of one's secret key. Thus, we need a relation R between public and secret keys, and we need an algorithm G to generate public/secret key pairs such that it is hard to guess the secret key. The following definition formalizes this:

Definition 20 (Hard instance generators) *We call an algorithm G a* hard instance generator *for a relation R iff*

- $\Pr[(p, s) \in R : (p, s) \leftarrow G()]$ *is overwhelming and*
- *for any polynomial-time A,* $\Pr[(p, s') \in R : (p, s) \leftarrow G(), s' \leftarrow A(p)]$ *is negligible.*

An example of a hard instance generator would be: $R := \{(p, s) : p = f(s)\}$ for a one-way function f, and G picks s uniformly from the domain of f, sets $p := f(s)$, and returns (p, s).

Now a signature is just a proof of knowledge of the secret key. That is, the statement is the public key, and the witness is the secret key. However, a signature should be bound to a particular message. For this, we include the message m in the statement that is proven. That is, the statement that is proven consists of a public key and a message, but the message is ignored when determining whether a given statement has a witness or not. (In the definition below, this is formalized by considering an extended relation R'.) The simulation-soundness of the proof system will then guarantee that a proof/signature with respect to one message cannot be transformed into a proof/signature with respect to another message because this would mean changing the statement.

A signature scheme consists of a key generation algorithm $(pk, sk) \leftarrow KeyGen()$. The secret key sk is used to sign a message m using the signing algorithm $\sigma \leftarrow Sign(sk, m)$ to get a signature σ. And the signature is valid iff $Verify(pk, \sigma, m) = 1$.

Definition 21 (Signatures from non-interactive proofs) *Let G be a hard instance generator for a relation R. Let $R' := \{((p, m), s) : (p, s) \in R\}$. Let (P, V) be a non-interactive proof system for R' (in the random oracle model). Then we construct the signature scheme $(KeyGen, Sign, Verify)$ as follows:*

- *$KeyGen()$: Pick $(p, s) \leftarrow G()$. Let $pk := p$, $sk := (p, s)$. Return (pk, sk).*
- *$Sign(sk, m)$ with $sk = (p, s)$: Run $\sigma \leftarrow P(x, w)$ with $x := (p, m)$ and $w := s$. Return σ.*
- *$Verify(pk, \sigma, m)$ with $pk = y$: Run $ok \leftarrow V(x, \sigma)$ with $x := (p, m)$. Return ok.*

Notice that if we use the scheme (P_{OE}, V_{OE}) from Definition 13, we do not need to explicitly find a sigma-protocol for the relation R'. This is because an HVZK sigma protocol with special soundness for R

will automatically also be an HVZK sigma protocol with special soundness for R'. Thus, the only effect of considering the relation R' is that in P_{OE}, the message m will be additionally included in the hash query $H(x, (com_i), (ch_i), (h_{i,j}))$ as part of $x = (p, m)$.

Definition 22 (Strong unforgeability) *A signature scheme* $(KeyGen, Sign, Verify)$ *is* strongly unforgeable *iff for all polynomial-time adversaries* A*,*

$$\Pr[ok = 1 \ \wedge \ (m^*, \sigma^*) \notin Q : H \leftarrow \mathsf{ROdist}, (pk, sk) \leftarrow KeyGen(), \\ (\sigma^*, m^*) \leftarrow A^{H,\mathbf{Sig}}(pk), ok \leftarrow Verify(pk, \sigma^*, m^*)]$$

is negligible. Here **Sig** *is a classical oracle that upon classical input* m *returns* $Sign(sk, m)$*. (But queries to* H *are quantum.) And* Q *is the list of all queries made to* **Sig***. (I.e., when* $\mathbf{Sig}(m)$ *returns* σ*,* (m, σ) *is added to the list* Q*.)*

If we replace $(m^*, \sigma^*) \notin Q$ *by* $\forall \sigma. (m^*, \sigma) \notin Q$*, we say the signature scheme is* unforgeable.

Theorem 23 (Unforgeability) *If* (P, V) *is zero-knowledge and has simulation-sound online-extractability, then the signature scheme* $(KeyGen, Sign, Verify)$ *from Definition 21 is strongly unforgeable.*

Proof. Fix a quantum-polynomial-time adversary A. We need to show that the following probability P_1 is negligible.

$$P_1 := \Pr[ok = 1 \ \wedge \ (m^*, \sigma^*) \notin Q : H \leftarrow \mathsf{ROdist}, (pk, sk) \leftarrow KeyGen(), \\ (\sigma^*, m^*) \leftarrow A^{H,\mathbf{Sig}}(pk), ok \leftarrow Verify(pk, \sigma^*, m^*)]$$

By definition of the signature scheme,

$$P_1 = \Pr[ok = 1 \ \wedge \ (m^*, \sigma^*) \notin Q : H \leftarrow \mathsf{ROdist}, (p, s) \leftarrow G(), (\sigma^*, m^*) \leftarrow A^{H,\mathbf{Sig}}(p), ok \leftarrow V((p, m^*), \sigma^*)]$$

And $\mathbf{Sig}(m)$ returns the proof $P((p, m), s)$. And G is the hard instance generator used in the construction of the signature scheme.

Since G is a hard instance generator, we have that $(p, s) \in R$ with overwhelming probability. Thus, with overwhelming probability, for all m, $((p, m), s) \in R'$. Thus, with overwhelming probability, **Sig** invokes $P((p, m), s)$ only with $((p, m), s) \in R'$. Since (P, V) is zero-knowledge (Definition 2), we can replace $H \leftarrow \mathsf{ROdist}$ by $H \leftarrow S_{init}()$ and $P((p, m), s)$ by $S_P((p, m))$ where (S_{init}, S_P) is the simulator for (P, V). That is, $|P_1 - P_2|$ is negligible where:

$$P_2 := \Pr[ok = 1 \ \wedge \ (m^*, \sigma^*) \notin Q : H \leftarrow S_{init}(), (p, s) \leftarrow G(), \\ (\sigma^*, m^*) \leftarrow A^{H,\mathbf{Sig}'}(p), ok \leftarrow V((p, m^*), \sigma^*)]$$

and $\mathbf{Sig}'(m)$ returns $S_P((p, m))$.

Let E be the extractor whose existence is guaranteed by the simulation-sound online-extractability of (P, V), see Definition 4. Consider the following game **G**:

$$\mathbf{G} \quad := \quad H \leftarrow S_{init}(), (p, s) \leftarrow G(), (\sigma^*, m^*) \leftarrow A^{H,\mathbf{Sig}'}(p), \\ ok \leftarrow V((p, m^*), \sigma^*), s' \leftarrow E(H, (p, m^*), \sigma^*).$$

That is, we perform the same operations as in P_2, except that we additionally try to extract a witness for the statement (p, m^*). Since the output of E is simply ignored, $\Pr[ok = 1 \ \wedge \ (m^*, \sigma^*) \notin Q : \mathbf{G}] = P_2$.

Let *simproofs* denote the list of queries made to S_P, i.e., whenever $\mathbf{Sig}'(m)$ queries $S_P((p, m))$ resulting in proof/signature σ, (p, m, σ) is appended to *simproofs*. Note that whenever some (p, m, σ) is appended to *simproofs*, (m, σ) is appended to Q. Thus $(m^*, \sigma^*) \notin Q$ implies $(p, m^*, \sigma^*) \notin \textit{simproofs}$.

Since (P, V) is simulation-sound online-extractable, $P_3 := \Pr[ok = 1 \wedge (p, m^*, \sigma^*) \notin \textit{simproofs} \wedge ((p, m^*), s') \notin R' : \mathbf{G}]$ is negligible.

Since $(m^*, \sigma^*) \notin Q$ implies $(p, m^*, \sigma^*) \notin \textit{simproofs}$, and $((p, m^*), s') \in R'$ iff $(p, s') \in R$, we have $P_3 \geq P_4$ with $P_4 := \Pr[ok = 1 \wedge (m^*, \sigma^*) \notin Q \wedge (p, s') \notin R : \mathbf{G}]$. Hence P_4 is negligible.

And since G is a hard instance generator and s is never given to any algorithm in **G**, $P_5 := \Pr[ok = 1 \wedge (m^*, \sigma^*) \notin Q \wedge (p, s') \in R : \mathbf{G}]$ is negligible.

Thus $P_2 = P_4 + P_5$ is negligible. And since $|P_1 - P_2|$ is negligible, P_1 is negligible. Since this holds for any quantum-polynomial-time A, the signature scheme is strongly unforgeable. □

Note that this proof is exactly as it would have been in the classical case (even though the adversary A was quantum). This is due to the fact that simulation-sound online-extractability as defined in Definition 4 allows us to extract a witness in a non-invasive way: we do not need to operate in any way on the quantum state of the adversary (be it by measuring or by rewinding); we get the witness purely by inspecting the classical proof/signature σ^*. This avoids the usual problem of disturbing the quantum state while trying to extract a witness.

Acknowledgments. We thank Marc Fischlin and Tommaso Gagliardoni for valuable discussions and the initial motivation for this work. This work was supported by the Estonian ICT program 2011-2015 (3.2.1201.13-0022), the European Union through the European Regional Development Fund through the sub-measure "Supporting the development of R&D of info and communication technology", by the European Social Fund's Doctoral Studies and Internationalisation Programme DoRa, by the Estonian Centre of Excellence in Computer Science, EXCS.

A Sigma-protocols with oblivious commitments

In this section we review the definition of sigma-protocols with oblivious commitments [DFG13] and explain why they directly imply NIZK proofs in the CRS model.

Definition 24 (Sigma-protocols with oblivious commitments, following [DFG13]) *A sigma-protocol $\Sigma = (N_{com}, N_{ch}, N_{resp}, P^1_\Sigma, P^2_\Sigma, V_\Sigma)$ has* oblivious commitments *if P^1_Σ simply chooses and return a uniformly random bitstring from N_{com}.*[13]

In other words, in a sigma-protocol with oblivious commitments, the first message (the commitment) is uniformly random. (While normally, we only require the second message to be uniformly random.)

Note that [DFG13] defines oblivious commitments slightly differently: the prover does not have to send a uniformly random commitment. Instead, given its commitment, it should be efficiently feasible to find randomness that leads to that commitment. But [DFG13] points out that that definition is equivalent to what we wrote in Definition 24 (in the sense that a protocol satisfying one definition can easily be transformed into one satisfying the other). Furthermore, [DFG13] actually assumes Definition 24 in their construction, so we give and discuss that definition here. [DFG13] proves (restated using the language from our paper):

Theorem 25 (Fiat-Shamir-like signatures, [DFG13]) *Assume a hard instance generator G and a sigma-protocol Σ with oblivious commitments, completeness, special-soundness, and HVZK.*

Then there is an unforgeable signature scheme (build in an efficient way from G and Σ).

The actual construction used [DFG13] is not Fiat-Shamir, but only inspired by Fiat-Shamir. The crucial difference is that the commitments are not chosen by the prover, but instead are hash values output by the random oracle (the same way as the challenges are output by the random oracle in normal Fiat-Shamir).

At the first glance this theorem might seem unrelated to the problem of constructing NIZK proofs. However, their proof of unforgeability implicitly proves the existence of an extractor (though not of a simulation-sound extractor) because it works by extracting two sigma-protocol executions and then computing a witness from those.

Note however that the proof from [DFG13] does not show that their construction is zero-knowledge. Yet, we conjecture that with the random oracle programming techniques presented here, one can show that their construction is zero-knowledge using a proof similar to ours.

Relation to CRS NIZK proofs. We now argue why sigma-protocols with oblivious commitments are quite a strong assumption. Namely, they are by themselves (without any use of a random oracle) already NIZK proofs of knowledge in the CRS model.

Given a sigma-protocol $\Sigma = (N_{com}, N_{ch}, N_{resp}, P^1_\Sigma, P^2_\Sigma, V_\Sigma)$ with oblivious commitments, we construct a proof system $\Pi_\Sigma = (CRS, P, V)$ in the CRS model as follows: The CRS crs is uniformly random from the set $crs := N_{com} \times N_{ch}$. The prover $P(crs, x, w)$ splits $crs =: (com, ch)$, runs $P^1_\Sigma(x, w)$ with the randomness that would yield com (this is possible because in a sigma-protocol with oblivious commitments,

[13] We stress that P^1_Σ needs to directly output its randomness. For example, if P^1_Σ produces $com := f(r)$ with random r using a one-way permutation f, then P^1_Σ does not have oblivious commitments, even though com is uniformly distributed. (Because P^1_Σ additionally produces a preimage of com under f.)

P^1_Σ just outputs its randomness), and runs $resp \leftarrow P^2_\Sigma(ch)$. The proof is $\pi := resp$. The verifier $V(crs, x, \pi)$ splits $crs =: (com, ch)$ and $resp := \pi$ and runs $V_\Sigma(x, com, ch, resp)$ and accepts if V_Σ accepts.

We now show that (P, V) is both zero-knowledge and a proof of knowledge in the CRS model.

Definition 26 (Zero-knowledge in the CRS model) *A non-interactive protocol* (CRS, P, V) *is* (single-theorem, non-adaptive) zero-knowledge in the CRS model for relation R *iff there exists a polynomial-time simulator* S *such that for any quantum-polynomial-time adversary* (A_1, A_2)*, the following is negligible:*

$$\begin{aligned}&\big|\Pr[(x,w) \in R \wedge b = 1 : (x,w) \leftarrow A_1(),\ crs \xleftarrow{\$} CRS,\ \pi \leftarrow P(crs, x, w),\ b \leftarrow A_2(crs, \pi)]\\ &- \Pr[(x,w) \in R \wedge b = 1 : (x,w) \leftarrow A_1(),\ crs, \pi \xleftarrow{\$} S(x),\ b \leftarrow A_2(crs, \pi)]\big|\end{aligned}$$

Notice that we have chosen the variant of zero-knowledge that is usually called single-theorem, non-adaptive zero-knowledge. That is, given one CRS, one is allowed to produce only a single proof. And the statement x that is to be proven may not depend on the CRS.

Lemma 27 *If* Σ *is a zero-knowledge sigma-protocol with oblivious commitments, then* Π_Σ *is zero-knowledge in the CRS model.*

Proof. Let $S(x)$ be a simulator that runs $(com, ch, resp) := S_\Sigma(x)$ where S_Σ is the simulator of the sigma-protocol (see Definition 5). Then S computes $crs := (com, ch)$ and $\pi := resp$ and returns (crs, π). Note that $crs = (com, ch) \xleftarrow{\$} CRS = N_{com} \times N_{ch}$ yields the same distribution of (com, ch) as $com \leftarrow P^1_\Sigma(x)$, $ch \xleftarrow{\$} N_{ch}$. Together with the fact that Σ is zero-knowledge, one easily sees that the probability difference in Definition 26 is negligible for quantum-polynomial-time (A_1, A_2). □

Definition 28 (Proofs of knowledge in the CRS model) *A non-interactive protocol* (CRS, P, V) *is a* (single-theorem, non-adaptive) proof of knowledge in the CRS model for relation R *iff there exists a polynomial-time extractor* (E_1, E_2) *such that the output of* E_1 *is quantum-computationally indistinguishable from* $crs \xleftarrow{\$} CRS$*, and such that for any quantum-polynomial-time adversary* (A_1, A_2)*, the following probability is negligible:*

$$\Pr[ok = 1 \wedge (x,w) \notin R : x \leftarrow A_1(), crs \leftarrow E_1(x), \pi \leftarrow A_2(crs), w \leftarrow E_2(\pi)] \qquad (7)$$

Note that again, we have defined a weak form of proofs of knowledge: single-theorem and non-adaptive.

Lemma 29 *Let* Σ *be a sigma-protocol with oblivious commitments. Assume that* Σ *is zero-knowledge with the following extra properties: for* $(com, ch, resp) \leftarrow S_\Sigma(x)$*,* (com, ch) *is quantum-computationally indistinguishable from uniform, and* $V_\Sigma(com, ch, resp) = 1$ *with overwhelming probability.*[14]

Then Π_Σ *is a proof of knowledge in the CRS model.*

Proof. Let $E_1(x)$ run the simulator $(com, ch, resp) \leftarrow S_\Sigma(x)$ of the sigma-protocol Σ. Then E_1 picks $ch' \xleftarrow{\$} N_{ch} \setminus ch$. Then E_1 outputs $crs := (com, ch')$.

Since (com, ch) chosen as $(com, ch, resp) \leftarrow S_\Sigma(x)$ is indistinguishable from uniform, so is (com, ch') as chosen by E_1. Thus $crs = (com, ch')$ as picked by $E_1(x)$ is quantum-computationally indistinguishable from $crs \xleftarrow{\$} CRS = N_{com} \times N_{ch}$.

The second part of the extractor, $E_2(\pi)$, sets $resp' := \pi$. This yields two executions of the sigma-protocol: $(com, ch, resp)$ and $(com, ch', resp')$ with $ch \neq ch'$. Then E_2 runs $w \leftarrow E_\Sigma(x, com, ch, resp, ch', resp')$ (the extractor of Σ) to get a witness w and returns that witness.

The first execution $(com, ch, resp)$ is valid (i.e., V_Σ accepts it) with overwhelming probability, since $(com, ch, resp)$ was produced by the simulator and thus passes verification with overwhelming probability (by assumption in the lemma). If additionally the second execution $(com, ch', resp')$ is valid (i.e., if $ok = 1$ in (7)), then E_Σ returns a correct witness with overwhelming probability (i.e., $(x, w) \in R$). Thus the case $ok = 1 \wedge (x, w) \notin R$ occurs with negligible probability, hence the probability in (7) is negligible. □

[14] At the first glance, those properties already follow from zero-knowledge and completeness of Σ. However, zero-knowledge and completeness do not apply when there exists no witness for x. So we need to explicitly require those conditions to also hold when x has no witness.

Note that the proof in [DFG13] does not need these conditions because in their setting, the statement x is the honestly generated public key of the signature scheme, and thus always has a witness. If, however, one would adapt their proof to show that their construction is actually a NIZK proof of knowledge, those conditions would be needed for the same reasons as in our proof of Lemma 29.

Summarizing, a sigma-protocol with oblivious commitments is already a NIZK proof of knowledge in the CRS model in itself. Hence sigma-protocols with oblivious commitments seem to be a much stronger assumption that just sigma-protocols. (At least we are not aware of any generic construction, classical or quantum, that transforms a sigma-protocols into a NIZK proof/proof of knowledge in the CRS model, without using random oracles.)

One may ask why the fact that sigma-protocols with oblivious commitments are already NIZK proofs of knowledge does not trivialize the construction from [DFG13] since it converts a NIZK proof of knowledge into a NIZK proof of knowledge. The crucial point is that sigma-protocols with oblivious commitments are only *single-theorem non-adaptive* NIZK proofs. So one can interpret the construction from [DFG13] as a way of strengthening a specific kind of NIZK proofs to become multi-theorem adaptive ones.[15] (Actually, seen like this, their construction becomes a very natural one: the statement is hashed using the random oracle, and the hash is used as a CRS for the proof.)

Sigma-protocols with oblivious commitments and efficient protocols. One major advantage of sigma-protocols is that they allow for very efficient constructions of sigma-protocols for complex relations from simpler ones [CDS94, Dam10]. For example, given sigma-protocols for two relations R_1, R_2, it is possible to build a sigma-protocol for the disjunction $R := \{((x_1, x_2), w) : (x_1, w) \in R_1 \vee (x_2, w) \in R_2\}$. Unfortunately, even when starting with sigma-protocols with oblivious commitments for R_1, R_2, the resulting sigma-protocol for R will not have oblivious commitments any more. This is because the protocol for R sends a commitment (com_1, com_2) where com_1 is generated by the prover of R_1, and com_2 by the simulator of R_2 (or vice versa). Since given the output of the simulator, it is in general hard to determine its randomness, it will not be possible to find the randomness that lead to com_2. Hence the protocol does not have oblivious commitments.

References

[Adi08] Ben Adida. Helios: Web-based open-audit voting. In Paul C. van Oorschot, editor, *USENIX Security Symposium 08*, pages 335–348. USENIX, 2008. Online at http://www.usenix.org/events/sec08/tech/full_papers/adida/adida.pdf.

[ARU14a] Andris Ambainis, Ansis Rosmanis, and Dominique Unruh. Quantum attacks on classical proof systems – the hardness of quantum rewinding. arXiv:1404.6898v1 [quant-ph], April 2014. Full version of [ARU14b].

[ARU14b] Andris Ambainis, Ansis Rosmanis, and Dominique Unruh. Quantum attacks on classical proof systems - the hardness of quantum rewinding. In *FOCS 2014*. IEEE, October 2014. To appear, preprint is [ARU14a].

[BCC04] Ernie Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In *ACM CCS '04*, pages 132–145, New York, NY, USA, 2004. ACM.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, pages 103–112, New York, NY, USA, 1988. ACM.

[BO81] Michael Ben-Or. Probabilistic algorithms in finite fields. In *FOCS 1981*, pages 394–398. IEEE, 1981.

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Yvo Desmedt, editor, *Crypto 94*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187. Springer, 1994.

[Dam10] Ivan Damgård. On σ-protocols. Course notes for "Cryptologic Protocol Theory", http://www.cs.au.dk/~ivan/Sigma.pdf, 2010. Retrieved 2014-03-17. Archived at http://www.webcitation.org/6O9USFecZ.

[DFG13] Özgür Dagdelen, Marc Fischlin, and Tommaso Gagliardoni. The Fiat-Shamir transformation in a quantum world. In *Asiacrypt 2013*, volume 8270 of *LNCS*, pages 62–81. Springer, 2013. Online version IACR ePrint 2013/245.

[15] Assuming that their construction can indeed be proven secure as a NIZK proof of knowledge in the random oracle model.

[Fis05] Marc Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In *Crypto 2005*, volume 3621 of *LNCS*, pages 152–168. Springer, 2005.

[FKMV12] Sebastian Faust, Markulf Kohlweiss, Giorgia Azzurra Marson, and Daniele Venturi. On the non-malleability of the Fiat-Shamir transform. In Steven Galbraith and Mridul Nandi, editors, *INDOCRYPT 2012*, volume 7668 of *LNCS*, pages 60–79. Springer, 2012. Preprint on IACR ePrint 2012/704.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Crypto '86*, number 263 in LNCS, pages 186–194. Springer, 1987.

[GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *J ACM*, 38(3):690–728, 1991.

[Gro06] Jens Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In Xuejia Lai and Kefei Chen, editors, *Asiacrypt 2006*, volume 4284 of *LNCS*, pages 444–459. Springer, 2006.

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel Smart, editor, *Eurocrypt 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 415–432. Springer, 2008.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999. Online at `http://www.icsi.berkeley.edu/~luby/PAPERS/hill.ps`.

[PS96] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In Ueli Maurer, editor, *Eurocrypt 96*, volume 1070 of *LNCS*, pages 387–398. Springer, 1996.

[Sah99] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *FOCS '99*, pages 543–553. IEEE, 1999.

[Sho94] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *FOCS 1994*, pages 124–134. IEEE, 1994.

[Unr12] Dominique Unruh. Quantum proofs of knowledge. In *Eurocrypt 2012*, volume 7237 of *LNCS*, pages 135–152. Springer, April 2012. Preprint on IACR ePrint 2010/212.

[Unr14a] Dominique Unruh. Quantum position verification in the random oracle model. IACR ePrint 2014/118, version 20140216:194504, February 2014. A short version appears at Crypto 2014.

[Unr14b] Dominique Unruh. Revocable quantum timed-release encryption. In *Eurocrypt 2014*, volume 8441 of *LNCS*, pages 129–146. Springer, 2014. Full version on IACR ePrint 2013/606.

[Wat09] John Watrous. Zero-knowledge against quantum attacks. *SIAM J. Comput.*, 39(1):25–58, 2009.

[Zha12] Mark Zhandry. Secure identity-based encryption in the quantum random oracle model. In *Crypto 2012*, volume 7417 of *LNCS*, pages 758–775. Springer, 2012.

[Zha13] Mark Zhandry. A note on the quantum collision and set equality problems. arXiv:1312.1027v3 [cs.CC], December 2013.

Symbol index

Keyword index