An abridged version of this paper appears in the Proceedings of the 10th Conference on Security and Cryptography for Networks—SCN 2016. This is the full version.

Fiat–Shamir for Highly Sound Protocols is Instantiable

Arno Mittelbach[1] Daniele Venturi[2]

[1] Cryptoplexity, Technische Universität Darmstadt, Germany
[2] Department of Computer Science, Sapienza University of Rome, Italy
mail@arno-mittelbach.de, venturi@di.uniroma1.it

Abstract. The Fiat–Shamir (FS) transformation (Fiat and Shamir, Crypto '86) is a popular paradigm for constructing very efficient non-interactive zero-knowledge (NIZK) arguments and signature schemes from a hash function and any three-move interactive protocol satisfying certain properties. Despite its wide-spread applicability both in theory and in practice, the known positive results for proving security of the FS paradigm are in the random oracle model only, i.e., they assume that the hash function is modeled as an external random function accessible to all parties. On the other hand, a sequence of negative results shows that for certain classes of interactive protocols, the FS transform cannot be instantiated in the standard model.

We initiate the study of complementary positive results, namely, studying classes of interactive protocols where the FS transform *does* have standard-model instantiations. In particular, we show that for a class of "highly sound" protocols that we define, instantiating the FS transform via a q-wise independent hash function yields NIZK arguments and secure signature schemes. In the case of NIZK, we obtain a weaker "q-bounded" zero-knowledge flavor where the simulator works for all adversaries asking an a-priori bounded number of queries q; in the case of signatures, we obtain the weaker notion of random-message unforgeability against q-bounded random message attacks.

Our main idea is that when the protocol is highly sound, then instead of using random-oracle programming, one can use complexity leveraging. The question is whether such highly sound protocols exist and if so, which protocols lie in this class. We answer this question in the affirmative in the common reference string (CRS) model and under strong assumptions. Namely, assuming indistinguishability obfuscation and puncturable pseudorandom functions we construct a compiler that transforms any 3-move interactive protocol with instance-independent commitments and simulators (a property satisfied by the Lapidot–Shamir protocol, Crypto '90) into a compiled protocol in the CRS model that is highly sound. We also present a second compiler, in order to be able to start from a larger class of protocols, which only requires instance-independent commitments (a property for example satisfied by the classical protocol for quadratic residuosity due to Blum, Crypto '81). For the second compiler we require dual-mode commitments.

We hope that our work inspires more research on classes of (efficient) 3-move protocols where Fiat–Shamir is (efficiently) instantiable.

Keywords. Fiat-Shamir transform, non-interactive zero knowledge, signature schemes, indistinguishability obfuscation, standard model

Contents

1 Introduction

The Fiat–Shamir (FS) transformation [FS87] is a popular[1] technique to build efficient non-interactive zero-knowledge (NIZK) arguments and signature schemes, starting from three-round *public-coin* (3PC) protocols satisfying certain properties. In a 3PC protocol the prover (with statement x and witness w) starts by sending a commitment α, to which the verifier replies with a challenge β drawn at random from some space $\mathcal{B}$; finally the prover sends a reply γ and the verifier's verdict is computed as a predicate of the transcript (α, β, γ) and the statement x being proven.[2]

1.1 Fiat–Shamir NIZK and Signatures

We briefly review the two main applications of the FS transform below.

NIZK. A NIZK is a non-interactive protocol in which the prover—holding a witness w for membership of a statement x in some *NP*-language L—can convince the verifier—holding just x—that $x \in L$, by sending a single message π. NIZK should satisfy three properties. First, *completeness* says that an honest prover holding a valid witness (almost) always convinces an honest verifier. Second, *soundness* says that a malicious prover should not be able to convince the honest verifier into accepting a *false* statement, i.e. a statement $x \notin L$; we speak of *arguments* (resp., *proofs*) when the soundness requirement holds for all computationally bounded (resp., computationally unbounded) provers. Third, *zero knowledge* requires that a proof does not reveal anything about the witness beyond the validity of the statement being proven.

NIZK require a setup assumption, typically in the form of a common reference string (CRS). Starting from a 3PC protocol, the FS transform makes it a NIZK by having the prover compute the verifier's challenge as a hash of the commitment α via some hash function H (with "hash key" hk); this results in a single message $\pi = (\alpha, \beta, \gamma)$, where $\beta = \mathsf{H}(\mathsf{hk}, \alpha)$, that is sent from the prover to the verifier.[3] (The description of the hash function, i.e. key hk, is included as part of the CRS.)

Apart from being a fascinating topic, NIZK have been demonstrated to be extremely useful for cryptographic applications (see, e.g., [GMW87, KZZ15, EL04, CL02, CHL05, DDN91]).

Signatures. Digital signatures are among the most important and well-studied cryptographic tools. Signature schemes allow a signer (holding a public/secret key pair $(\mathsf{pk}, \mathsf{sk})$) to generate a signature σ on a message m, in such a way that anyone possessing the public key pk can verify the validity of (m, σ). Signatures must be unforgeable, meaning that it should be hard to forge a signature on a "fresh" chosen message (even after seeing polynomially many signatures on possibly chosen messages).

Starting with a 3PC protocol, the FS transform makes it a signature scheme by having the signer compute the verifier's challenge as a hash of the commitment α, concatenated with the message m, via some hash function H (with "hash key" hk); this results in a signature $\sigma = (\alpha, \beta, \gamma)$, where $\beta = \mathsf{H}(\mathsf{hk}, \alpha||m)$.

[1] There are over 3.000 Google-Scholar-known citations to [FS87], as we type.

[2] Protocols with this shape are sometimes known as Sigma protocols; however, the definition of Sigma protocols typically assumes that the underlying protocol satisfies certain security property which will be slightly different from the ones we need.

[3] The value β is typically omitted from the proof, as the verifier can compute it by itself.

1.2 Positive and Negative Results

We refer to the non-interactive system obtained by applying the FS transform to a 3PC protocol (i.e., a NIZK or a signature scheme) as the *FS collapse*. A fundamental question in cryptography is to understand what properties the initial 3PC protocol and the hash function should satisfy in order for the FS collapse to be a NIZK argument or a secure signature scheme. This question has been studied extensively in the literature; we briefly review the current state of affairs below.

Positive results. All security proofs for the FS transform follow the random oracle methodology (ROM) of Bellare and Rogaway [BR93], i.e., they assume that the function H behaves like an external random function accessible to all parties (including the adversary). In particular, a series of papers [FS87, Oka93, PS00, AABN02] establishes that the FS transform yields a secure signature scheme in the ROM provided that the starting 3PC is a passively secure identification scheme. The first definition of NIZK in the ROM dates back to [BR93] (where a particular protocol was analyzed); in general, it is well known that, always in the ROM, the FS transform yields a NIZK satisfying sophisticated properties such as simulation-soundness [FKMV12] and simulation-extractability [BPW12].

Barak *et al.* [BLV03] put forward a new hash function property (called entropy preservation[4]) that allows to prove soundness of the FS transformation without random oracles; their result requires that the starting 3PC protocol is statistically sound, i.e. it is a *proof*. Dodis *et al.* [DRV12] show that such hash functions exist in case a conjecture on the existence of certain "condensers for leaky sources" turns out to be true. Canetti *et al.* [CCR16] study the correlation intractability of obfuscated pseudorandom functions and show a close connection between entropy preservation and correlation intractability, but it remains open whether their construction achieves entropy preservation or, in fact, whether entropy-preserving hash functions exist in the standard model. A negative indication to this question was recently presented by Bitansky *et al.* [BDG$^+$13] who show that entropy-preservation security cannot be proven via a black-box reduction to a *cryptographic game*.

Negative results. It is often difficult to interpret what a proof in the ROM means in the standard model. This is not only because concrete hash functions seem far from behaving like random oracles, but stems from the fact that there exist cryptographic schemes that can be proven secure in the ROM, but are always insecure in the standard model no matter how we instantiate the hash function [CGH98].

The FS transformation is not an exception in this respect. In their study of "magic functions", Dwork *et al.* [DNRS99] establish that whenever the initial 3PC protocol satisfies the zero-knowledge property, its FS collapse can never be (computationally) sound for any implementation of the hash function. Goldwasser and Kalai [GK03], building on previous work of Barak [Bar01], construct a specially-crafted 3PC *argument* for which the FS transform yields an insecure signature scheme for any standard model implementation of the hash function; this in particular means that the random oracle in the FS transform cannot be universally instantiated on all 3PC arguments.

Recently, Bitansky *et al.* [BGW12] and Dachman-Soled *et al.* [DJKL12] (see also [BDG$^+$13]) show an unprovability result that also covers 3PC *proofs*. More in detail, [BGW12] shows that the FS transform cannot always preserve soundness when starting with a 3PC proof, under a black-box reduction to any falsifiable assumption (even ones with an inefficient challenger). [DJKL12] shows a

[4]Entropy preservation roughly says that for all efficient adversaries that get a uniformly random hash key hk and produce a correlated value α, the conditional Shannon entropy of $\beta = \mathsf{H}(\mathsf{hk}, \alpha)$ given α, but not hk, is sufficiently large.

similar black-box separation (although only for assumptions with an efficient challenger) for any concrete proof that is honest-verifier zero knowledge against sub-exponential size distinguishers. In a related paper, Goyal *et al.* [GOSV14] obtain a negative result for non-interactive information-theoretically secure witness indistinguishable arguments.

1.3 Our Contributions

The negative results show that, for certain classes of interactive protocols, the FS transform cannot be instantiated in the standard model. We initiate the study of complementary positive results, namely, studying classes of interactive protocols where the FS transform *does* have a standard-model instantiation. We show that for a class of "highly sound" protocols that we define, instantiating the FS transform via a q-wise independent hash function yields both a NIZK argument in the CRS model and a secure signature scheme. In the case of NIZK, we obtain a weaker "q-bounded" zero-knowledge flavor where the simulator works for all adversaries asking an a-priori bounded number of queries q; in the case of signatures, we obtain the weaker notion of random-message unforgeability against q-bounded random message attacks, where the forger can only observe signatures on random messages and has to produce a forgery on a fresh random message.

Very roughly, highly sound protocols are a special class of 3PC arguments and identification schemes satisfying three additional properties: (**P1**) The honest prover computes the commitment α independently of the instance being proven and of the corresponding witness; (**P2**) The soundness error of the protocol is tiny, in particular the ratio between the soundness error and the worst-case probability of guessing a given commitment is bounded-away from one; (**P3**) Honest conversations between the prover and the verifier on a common input x can be simulated knowing just x, and moreover the simulator can fake α independently of x itself.

We are not aware of natural protocols that are directly highly sound according to our definition. (But we will later discuss that, e.g., the Lapidot–Shamir protocol [LS91] partially satisfies our requirements.) Hence, the question is whether such highly sound protocols exist and, if so, which languages and protocols lie in this class. We answer this question in the affirmative in the CRS model and under strong assumptions. Namely, assuming indistinguishability obfuscation, puncturable pseudorandom functions and equivocal commitments, we build a sequence of two compilers that transform any three-move interactive protocol with instance-independent commitments (i.e., property **P1**) into a compiled protocol in the CRS model that satisfies the required properties. Noteworthy, our compilers are language-independent, and we know that assuming one-way permutations three-move interactive protocols with instance-independent commitments exist for all of *NP*. We refer the reader to Section 1.4 for a more in-depth interpretation of our results.

Our result avoids Dwork *et al.* [DNRS99], because we start from a protocol that is honest-verifier zero knowledge rather than fully zero knowledge. Note that our approach also circumvents the negative result of [BGW12, GOSV14] as our technique applies only to a certain class of 3PC arguments. Furthermore, we circumvent the black-box impossibility result [DJKL12] by using complexity leveraging and sub-exponential security assumptions.

1.4 Perspective

The main contribution from our perspective is to initiate the study of restricted positive standard-model results for the FS transform. Namely, we show that for the class of highly sound protocols, the FS transform can be instantiated via a q-wise independent hash function (both for the case of NIZK and signatures). This is particularly interesting given the negative results in [DNRS99, GK03, BDG$^+$13].

An important complementary question is, of course, to study the class of highly sound protocols. Under strong assumptions, our compilers show that highly sound protocols exist for all languages in *NP*. However, the compilers yield protocols in the CRS model and, as we discuss now, one has to take particular care in interpreting positive results about the FS transform applied to 3PC protocols in the CRS model.

It is well known that in the CRS model one can obtain a NIZK both for *NP*-complete languages [BFM88] and for specific languages [GS08]. Given such a NIZK we can now obtain a trivial 3PC protocol in the CRS model that is highly sound according to our definition: The first message α^* and the second message β^* of the compiled protocol are equal to the empty string ε; the third message is a NIZK proof γ^* that $x \in L$. Naturally, the FS transform is secure (even without random oracles) since we started with a secure NIZK and messages α^* and β^* play no role whatsoever. Now, given this trivial 3PC protocol we can create a "false compiler" that compiles any 3PC protocol for proving membership of elements $x \in L$ into a highly sound 3PC protocol in the CRS model: The compiler ignores the *input 3PC* and instead outputs the above trivial 3PC with a NIZK as the third message. Of course, the security of this compiler is not based on the security of the starting 3PC but on the security of the NIZK. In particular, there is no security reduction of the compiled protocol to any of the security properties of the starting protocol. Hence, such a "false compiler" does not shed any light on the security of the FS transform and when it applies.

The compilers that we present in this work, instead, are very different. Although they share the fact they produce 3PCs in the CRS model, our compilers utilize all the security properties of the starting 3PC protocol. In particular, this means that we present reductions for all the security properties of the resulting protocol (i.e., soundness, completeness, and zero knowledge) to the respective properties of the initial 3PC protocol. Our compilers thus showcase how 3PC protocols could be built to satisfy properties **P1**-**P3**.

Let us stress that our FS transform works even if the starting 3PC is in the standard model (provided that it satisfies **P1**-**P3**). It is an intriguing open question whether compilers exist that are also in the standard model and that do not need to leverage the power of a CRS.

1.5 Related Work and Open Questions

On Fiat–Shamir. It is worth mentioning that using indistinguishability obfuscation and puncturable PRFs one can directly obtain a NIZK for all *NP* as shown by Sahai and Waters [SW14]. However, our main focus is not on constructions of NIZK, rather we aim at providing a better understanding of what can be proved for the FS transform without relying on random oracles. In this respect, our result shares similarities to the standard-model instantiation of Full-Domain Hash given in [HSW14].

In the case of NIZK, an alternative version of the FS transform is defined by having the prover hashing the statement x together with value α, in order to obtain the challenge β. The latter variant is sometimes called the *strong* FS transform (while the variant we analyze is known as the *weak* FS transform). Bernhard *et al.* [BPW12] show that the weak FS transform might lead to problems in certain applications where the statement to be proven can be chosen adversarially (this is the case, e.g., in the Helios voting protocol). Unfortunately, it seems hard to use our proof techniques to prove zero knowledge of the strong FS collapse, because the simulator for zero knowledge does not know the x values in advance.

Our positive result for FS signatures shares some similarities with the work of Bellare and Shoup [BS07], showing that "actively secure" 3PC protocols yield a restricted type of secure signature schemes (so-called two-tier signatures) when instantiating the hash function in the FS transform via any collision-resistant hash function.

Compilers. Our approach of first compiling any "standard" 3PC protocol into one with additional properties that suffice for proving security of the FS transform is similar in spirit to the approach taken by Haitner [Hai09], who shows how to transform any interactive argument into one for which parallel repetition decreases the soundness error at an exponential rate.

Lindell recently used a similar idea to first transform a 3PC into a new protocol in the CRS model, and then shows that the resulting 3PC when transformed with (a slightly modified version of) Fiat–Shamir satisfies zero knowledge in the standard model [Lin15]. His approach was later improved in [CPS+16c]. We note that the use of a CRS-enhanced interactive protocol is only implicit in Lindell's work as he directly analyzes the collapsed non-interactive version. On the downside, to prove soundness Lindell still requires (non-programmable) random oracles. We note that one of our compilers is essentially equivalent to the compiler used by Lindell. Before Lindell's work, interactive protocols in the CRS model have also been studied by Damgård who shows how to build 3-round concurrent zero-knowledge arguments for all *NP*-problems in the CRS model [Dam00].

Alternative transforms. Other FS-inspired transformations were considered in the literature. For instance Fischlin's transformation [Fis05] (see also [DV14]) yields a simulation-sound NIZK argument with an online extractor; as mentioned above, [Lin15, CPS+16c] defined a twist of the FS transform that allows to prove zero knowledge in the CRS model, and soundness in the non-programmable random oracle model. It is an interesting direction for future research to apply our techniques to analyze the above transformations without random oracles.

Concurrent paper. Recently, in a concurrent and independent work, Kalai, Rothblum and Rothblum [KRR16] showed a positive result for FS in the plain model, under complexity assumptions similar to ours. More in details, assuming sub-exponentially secure indistinguishability obfuscation, input-hiding obfuscation for the class of multi-bit point functions, and sub-exponentially secure one-way functions, [KRR16] shows that, when starting with any 3PC *proof*, the FS transform yields a *two-round* computationally-sound interactive protocol.

On the positive side, their result applies to any 3PC proof (while ours only covers a very special class of 3PC arguments). On the negative side, their technique only yields a positive result for a two-round interactive variant of the FS transform (while our techniques apply to the full FS collapse, both for NIZK and for signatures).

1.6 Roadmap

We provide a detailed informal overview of our main techniques in Section 2. In Section 3 we set up some notation and define the main cryptographic primitives on which we build. Section 4 contains our positive result for FS NIZK. We present our compilers for obtaining highly sound protocols (in the CRS model) in Section 5 and Section 6. Finally, we explain how to adapt our techniques to the case of FS signatures in Section 7.

2 Technical Overview

We first discuss the class of highly sound protocols for which the FS transform can be instantiated via a q-wise independent hash function. Then, we will explain how to obtain such protocols by presenting a compiler that transforms a large class of 3PC protocols into ones that are highly sound (in the CRS model). For the purpose of this overview we will only focus on the case of Fiat–Shamir NIZK, explaining only at the end how our techniques can be adapted to cover also Fiat–Shamir signatures.

2.1 Proof Idea

The security proof proceeds in two modular steps. In the first step, we prove completeness and soundness of a "selective" variant of the FS transform (which we define formally in Section 4.1); in the second step we analyze the standard FS transform using complexity leveraging. Details follow.

Consider a 3PC argument for a language L. For a hash family H, consider the following (interactive) selective adaptation of the FS transformation: The prover sends the commitment α as in the original protocol; the verifier, instead of sending the challenge $\beta \in \mathcal{B}$ directly, forwards an honestly generated hash key hk; finally the prover uses (hk, α) to compute $\beta = \mathsf{H}(\mathsf{hk}, \alpha)$ and then obtains the response γ as in the original 3PC argument. In Section 4 we prove that if the starting 3PC protocol is complete and computationally sound, so is the one obtained by applying the selective FS transform. The idea is to use a "programmable" q-wise independent hash function (e.g., a random polynomial of degree $q-1$ over a finite field) to "program" the hash function up-front; note that commitment α is computed before the hash key is generated, and hence, we can embed the challenge value β into the hash function such that it maps α to β and reduce to the soundness of the underlying 3PC argument.

Complexity leveraging. The second step in proving soundness of the FS collapse (we discuss zero knowledge below) consists in applying complexity leveraging so that we can swap the order of α and β. Note however that if β is shorter than α, and if the soundness of the protocol is $2^{-|\beta|}$, then we loose too much through complexity leveraging. Hence, this step can only be applied to protocols satisfying an additional property as we discuss next.

Let Π be the initial 3PC argument, and denote by $\overline{\Pi}$ its corresponding FS collapse. Given a malicious prover P^* breaking soundness of $\overline{\Pi}$, we construct a prover P attacking soundness of the selective FS transform as follows. P picks a random α from the set of all possible commitments, and forwards α to the verifier; after receiving the challenge hash key hk, prover P runs P^* which outputs a proof (α^*, γ^*). Prover P simply hopes that $\alpha^* = \alpha$, in which case it forwards γ^* to the verifier (otherwise it aborts). It follows that if the selective FS has soundness roughly $s(\lambda)$ (for security parameter λ), the soundness of $\overline{\Pi}$ is roughly $s(\lambda)$ divided by the probability of guessing correctly the value α^* in the first step of the reduction.

Note that for the above argument to give a meaningful bound, we need that the soundness of $\overline{\Pi}$ is bounded away from one. This leads to the following (non-standard) requirement that the initial 3PC argument should satisfy.

> **P2**: $\varrho(\lambda) := s(\lambda)/2^{-a(\lambda)} < 1$, where $s(\lambda)$ is the soundness error and $a(\lambda)$ is the maximum bit-length associated to the commitment α.

Zero knowledge. We assume that the initial 3PC is honest-verifier zero knowledge (HVZK)—i.e., that it is zero knowledge for honest verifiers. We need to show that $\overline{\Pi}$ satisfies zero knowledge. Here, we require two additional properties as explained below; interactive protocols obeying the first property already appeared in the literature under the name of "input-delayed" protocols [CPS+16a, HV16, CPS+16b].

> **P1**: The value α output by the prover is computed independently of the instance x being proven (and of the corresponding witness w).

> **P3**: The value α output by the simulator is computed independently of the instance x being proven.

We now discuss the reduction for the zero-knowledge property and explain where **P1** and **P3** are used. We need to construct an efficient simulator that is able to simulate arguments for adaptively chosen (true) statements—without knowing a witness for such statements. The output of the simulator should result in a distribution that is computationally indistinguishable from the distribution generated by the real prover. The simulator gets extra power, as it can produce a "fake" CRS together with some trapdoor information tk (on which the simulator can rely) such that the "fake" CRS is indistinguishable from a real CRS.

In order to build some intuition, it is perhaps useful to recall the random-oracle-based proof for the zero-knowledge property of the FS transform. There, values α_i and β_i corresponding to the i-th adversarial query are computed by running the HVZK simulator and are later "matched" relying on the programmability of the random oracle. Roughly speaking, in our standard-model proof we take a similar approach, but we cannot use *adaptive* programming of the hash function. Instead, we rely on **P1** and **P3** to program the hash function in advance. More specifically, the trapdoor information will consist of q random tapes r_i (one for simulating each proof queried by the adversary) and the corresponding q challenges β_i (that can be pre-computed as a function of r_i, relying on **P1**). Since the challenges have the correct distribution, we can use the underlying HVZK simulator to simulate the proofs; here is where we need **P3**, as the simulator has to pre-compute the values α_i in order to embed the β_i values on the correct points.

A caveat is that our simulator needs to know the value of q in advance; for this reason we only get a weaker *bounded* flavor of the zero-knowledge property where there exists a "universal" simulator that works for all adversaries asking q queries, for some a-priori fixed value of q. Note, however, that the CRS—as it contains the description of a q-wise independent hash function—needs to grow with q, and hence bound q should be seen as a parameter of the construction rather than a parameter of the simulator.

It is an interesting open problem whether this limitation can be removed, thus proving that actually our transformation achieves unbounded zero knowledge.

2.2 Compilers

Wrapping up the above discussion, we can show that for 3PC protocols that satisfy completeness, computational soundness, HVZK and additionally **P1**-**P3**, the FS transform can be instantiated by a (programmable) q-wise independent hash function. We informally refer to protocols that satisfy all of the above properties as *highly sound* arguments.

Unfortunately we do not know of a natural highly sound 3PC argument. However, we do know of protocols that partially satisfy our requirements. Recall, for instance, the classical 3PC argument for quadratic residuosity due to Blum [Blu81] (all operations are modulo an integer N which is the product of two Blum integers): (i) The prover chooses a random r in $\mathbb{Z}_N^*$, and sends $\alpha = r^2$; (ii) The verifier selects a random bit $\beta \in \{0, 1\}$; (iii) The prover computes $\gamma = w^\beta \cdot r$, and finally the verifier checks that $\gamma^2 = x^\beta \cdot \alpha$. While the above clearly satisfies **P1** and moreover can be shown to achieve completeness, soundness, and HVZK, one can easily see that **P2** and **P3** are not directly met. **P2** is not met, because β consists only of a single bit and the soundness parameter is $\frac{1}{2}$, and to see that **P3** is not met, one needs to consider the simulator for this protocol which—for readers familiar with the protocol—computes its first message depending on the statement.

Another interesting example is given by the Lapidot–Shamir protocol for the *NP*-complete problem of graph Hamiltonicity [LS91] (see also [OV12, Appendix B]). Here, the prover's commitment consists of a (statistically binding) commitment to the adjacency matrix of a random k-vertex cycle, where k is the size of the Hamiltonian cycle.[5] Hence, the protocol clearly satisfies **P1**. Additionally

[5]Note that the value k can be included in the language, and thus considered as public.

the simulator fakes the prover's commitment by either committing to a random k-vertex cycle, or by committing to the empty graph. Hence, the protocol also satisfies **P3**. As a corollary, we know that assuming non-interactive statistically binding commitment schemes (which follow from one-way permutations [Blu81]), for all languages in *NP*, there exist 3PC protocols that satisfy completeness, computational soundness, and HVZK, as well as **P1** and **P3**.

Motivated by the above examples, we turn to the question whether it is possible to compile a 3PC protocol (with completeness, soundness, and HVZK) satisfying either **P1** or **P1** and **P3**, into a highly sound argument. We refer the reader to Section 4.5 for a high-level overview how this can be achieved. We only mention here that our compilers rely on several cryptographic tools (including indistinguishability obfuscation, puncturable PRFs, complexity leveraging and equivocal commitment schemes), and yield a 3PC in the CRS model; note that this means that we obtain an interactive protocol with a CRS even if the original protocol was in the standard model. It is an intriguing open problem if a highly sound argument can be constructed in the standard model, or whether a CRS is, in fact, necessary.

2.3 The Case of Signatures

Finally, let us explain how our techniques can be adapted to the case of FS signatures. To this end, we introduce a notion of *highly-sound* canonical identification schemes that need to satisfy similar requirements to the properties **P1**, **P2**, and **P3** discussed above for the case of 3PC arguments.

Recall that in order to apply our main technique, we need to program the q-wise independent hash function up-front. For this reason we are only able to show that our standard-model instantiation of FS signatures achieves the weaker notion of random-message unforgeability against q-bounded random-message attacks (q-bounded RUF-RMA)—in which the adversary can only observe signatures on up-to q randomly chosen messages, and also has to forge on an additional fresh random message. While strictly weaker that standard existential unforgeability against chosen-message attacks (EUF-CMA), RUF-RMA is still useful for some applications (e.g., to secure authentication [FHN$^+$12, NVZ14]). We refer the reader directly to Section 7 for the details.

3 Preliminaries

3.1 Notation

By $\lambda \in \mathbb{N}$, we denote the security parameter that we give to all algorithms implicitly in unary representation 1^λ. By $\{0,1\}^\ell$ we denote the set of all bit-strings of length ℓ, and by $\{0,1\}^*$ the set of all bit-strings of finite length. If $x, y \in \{0,1\}^*$ are two bit strings, then $x\|y$ denotes concatenation. The length of x is denoted by $|x|$. We denote vectors of strings in bold face, for example, $\mathbf{x}$ and denote the i-th component by $\mathbf{x}[i]$. For a finite set X, we denote the action of sampling x uniformly at random from X by $x \leftarrow\!\!\$\, X$, and denote the cardinality of X by $|X|$. We denote by $[i]$ the set $\{1, 2, \ldots, i\}$. Algorithms are assumed to be randomized, unless otherwise stated. In particular polynomial-time refers to *deterministic* polynomial-time computable algorithms, while PPT refers to probabilistic polynomial-time. We write $x \leftarrow\!\!\$\, \mathcal{A}(\cdot)$ to denote that probabilistic algorithm $\mathcal{A}$ is run on freshly sampled random coins and produces output x. We write $x \leftarrow \mathcal{A}(\cdot; r)$ to denote that $\mathcal{A}$ runs on coins r. Similarly, we write $x \leftarrow \mathcal{A}(\cdot)$ to denote that deterministic algorithm $\mathcal{A}$ outputs x. We say a function $\mathsf{negl}(\lambda)$ is negligible if $\mathsf{negl}(\lambda) \in \lambda^{-\omega(1)}$. We say a function $\mathsf{poly}(\lambda)$ is polynomial if $\mathsf{poly}(\lambda) \in \lambda^{\mathcal{O}(1)}$.

Function families. We formalize families of functions F by considering a tuple of algorithms $\mathsf{F.KGen}$, $\mathsf{F.kl}$, $\mathsf{F.Eval}$, $\mathsf{F.il}$ and $\mathsf{F.ol}$. Algorithm $\mathsf{F.KGen}$ is a PPT algorithm taking the security parameter 1^λ and outputting a key $\mathsf{k} \in \{0,1\}^{\mathsf{F.kl}(\lambda)}$ where $\mathsf{F.kl} : \mathbb{N} \to \mathbb{N}$ denotes the key length. Functions $\mathsf{F.il} : \mathbb{N} \to \mathbb{N}$ and $\mathsf{F.ol} : \mathbb{N} \to \mathbb{N}$ denote the input and output length functions associated to F and for any $x \in \{0,1\}^{\mathsf{F.il}(\lambda)}$ and $\mathsf{k} \leftarrow\!\!\$\, \mathsf{F.KGen}(1^\lambda)$ we have that $\mathsf{F.Eval}(\mathsf{k}, x) \in \{0,1\}^{\mathsf{F.ol}(\lambda)}$, where the PPT algorithm $\mathsf{F.Eval}$ denotes the "evaluation" function associated to F. Depending on the function, Eval may be renamed to a more speaking name and additional algorithms might be added. If the functionality is randomized then we let $\mathsf{F.rl}(\lambda)$ denote the randomness length.

Asymptotic security. In this paper we allow adversaries to be probabilistic polynomial-time (PPT) and ask that the success probability be smaller than some function $\epsilon(\lambda)$ in the security parameter λ. However, when we fix a function $\epsilon(\lambda)$, then for finitely many λ, a specific PPT adversary might be more successful than $\epsilon(\lambda)$. Hence, when defining ϵ-security for a scheme, we say that for all PPT adversaries $\mathcal{A}$, the advantage function $\mathsf{Adv}_{\mathcal{A}}(\lambda)$ is asymptotically smaller than ϵ, denoted $\mathsf{Adv}_{\mathcal{A}}(\lambda) \overset{\text{asym}}{\leq} \epsilon(\lambda)$, which means that there is some value λ_0 such that for all $\lambda \geq \lambda_0$, it holds that $\mathsf{Adv}(\lambda) \leq \epsilon(\lambda)$.

For two random variables X and Y, we say that they are ϵ-indistinguishable, denoted $X \approx_\epsilon Y$, if for all PPT distinguishers the distinguishing advantage is asymptotically smaller than ϵ. In case X and Y are identically distributed, we simply write $X \equiv Y$.

3.2 One-Wayness

Within our compilers we make use of standard cryptographic one-way functions and here, for completeness, present a definition tailored to our notation.

Definition 3.1 (One-way function) *We call a family of functions* $\mathsf{F} = (\mathsf{F.KGen}, \mathsf{F.Eval}, \mathsf{F.il}, \mathsf{F.ol})$ one-way *if for all* PPT *adversaries* $\mathcal{A}$ *the advantage* $\mathsf{Adv}^{\text{owf}}_{\mathsf{F},\mathcal{A}}(\lambda)$ *defined as*

$$\mathsf{Adv}^{\text{owf}}_{\mathsf{F},\mathcal{A}}(\lambda) := \Pr\left[\mathsf{OWF}^{\mathsf{F}}_{\mathcal{A}}(\lambda)\right]$$

is negligible and where game $\mathsf{OWF}^{\mathsf{F}}_{\mathcal{A}}$ *is defined as:*

$\underline{\mathsf{OWF}^{\mathsf{F}}_{\mathcal{A}}(\lambda)}$
$\mathsf{k} \leftarrow\!\!\$\, \mathsf{F.KGen}(1^\lambda)$
$x \leftarrow\!\!\$\, \{0,1\}^{\mathsf{F.il}(\lambda)}$
$y \leftarrow \mathsf{F.Eval}(\mathsf{k}, x)$
$x^* \leftarrow\!\!\$\, \mathcal{A}(1^\lambda, \mathsf{k}, y)$
return $\mathsf{F.Eval}(\mathsf{k}, x^*) = y$

3.3 q-Wise Independent Hashing

We recall the standard notion of a q-wise independent hash function.

Definition 3.2 (q-Wise independent hashing) *A family of functions* $\mathsf{H} := (\mathsf{H.KGen}, \mathsf{H.kl}, \mathsf{H.Eval}, \mathsf{H.il}, \mathsf{H.ol})$ *is called* q-wise independent *if for all* $\lambda \in \mathbb{N}$*, any sequence of* $x_1, \ldots, x_q \in \{0,1\}^{\mathsf{H.il}(\lambda)}$*, and any* $y_1, \ldots, y_q \in \{0,1\}^{\mathsf{H.ol}(\lambda)}$*, we have that*

$$\Pr\left[\mathsf{H.Eval}(\mathsf{hk}, x_1) = y_1 \wedge \ldots \wedge \mathsf{H.Eval}(\mathsf{hk}, x_q) = y_q : \mathsf{hk} \leftarrow\!\!\$\, \mathsf{H.KGen}(1^\lambda)\right] \leq 2^{-q \cdot \mathsf{H.ol}(\lambda)}.$$

We call a q-wise independent hash function programmable *if there exists an additional procedure* $\mathsf{H.KGen}(1^\lambda, \mathbf{x}, \mathbf{y})$ *such that for any two q-size vectors* $\mathbf{x}$ *and* $\mathbf{y}$*, with* $\mathbf{x}[i] \in \{0,1\}^{\mathsf{H.il}(\lambda)}$ *for all* $i \in [q]$ *(with* $\mathbf{x}[i] \neq \mathbf{x}[j]$ *for all* $i \neq j$*) and* $\mathbf{y}[i] \in \{0,1\}^{\mathsf{H.ol}(\lambda)}$ *for all* $i \in [q]$*, we have that*

$$\Pr\left[\mathsf{H.Eval}(\mathsf{hk}, \mathbf{x}[i]) = \mathbf{y}[i] : \mathsf{hk} \leftarrow\!\!\$\, \mathsf{H.KGen}(1^\lambda, \mathbf{x}, \mathbf{y})\right] = 1.$$

Furthermore, the distributions $\mathsf{H.KGen}(1^\lambda)$ *and* $\mathsf{H.KGen}(1^\lambda, \mathbf{x}, \mathbf{y})$ *are identical, where the second distribution is also over the uniformly random choice of vectors* $\mathbf{x}$ *and* $\mathbf{y}$*.*

Note that we can construct a programmable q-wise independent hash function, for example, by considering polynomials of degree $q-1$ over finite fields. Here, programmability is obtained by using polynomial interpolation. An interesting special case is when considering a 1-wise independent hash function. We can obtain a 1-wise independent hash function by a constant function (i.e., a polynomial of degree 0). Hence, the hash function family of constant functions defined as $\mathsf{H}(\mathsf{hk}, x) := \mathsf{hk}$ fulfills the requirements of a 1-wise independent and programmable hash function. In this case the programming is performed by choosing the key to match the (single) target value y.

3.4 Interactive and Non-Interactive Arguments

Let $R : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ be a polynomial-time computable relation together with a polynomial $p(\cdot)$, defining the *NP*-language

$$L_R := \{x : \exists w \text{ s.t. } |w| < p(|x|) \text{ and } R(x,w) = 1\}.$$

In the rest of the paper, we will drop the bound $p(\cdot)$ for ease of presentation. An interactive argument system for R consists of three PPT algorithms $(\mathsf{K}, \mathsf{P}, \mathsf{V})$. Algorithm K takes as input 1^λ and outputs a common reference string (CRS) $\mathsf{crs} \in \{0,1\}^*$. Later the prover P interacts with the verifier V to convince him into accepting a common input $x \in L_R$ (where both P and V are also given crs); the honest prover additionally holds a witness w for x, i.e. $R(x,w) = 1$. At the end of the protocol execution, the verifier outputs a bit (representing his decision); we write $\langle \mathsf{P}(w), \mathsf{V}\rangle(\mathsf{crs}, x)$ for the random variable corresponding to the verifier's verdict. Similarly, we write $\mathsf{P}(\mathsf{crs}, x, w) \rightleftarrows \mathsf{V}(\mathsf{crs}, x)$ for the random variable corresponding to transcripts of honest protocol executions.[6]

An interactive argument should satisfy at least two properties, completeness and soundness. Completeness says that an honest prover (holding a valid witness) is able to convince an honest verifier.

Definition 3.3 (Completeness) *Let* $\Pi = (\mathsf{K}, \mathsf{P}, \mathsf{V})$ *be an interactive argument system for a polynomial-time computable relation* R*. We say that* Π *satisfies c-completeness if for all* (x,w) *such that* $R(x,w) = 1$ *we have*

$$\Pr\left[\langle \mathsf{P}(w), \mathsf{V}\rangle(\mathsf{crs}, x) = 1 : \ \mathsf{crs} \leftarrow\!\!\$\, \mathsf{K}(1^\lambda)\right] \overset{asym}{\geq} 1 - c(\lambda),$$

where the probability is taken over the randomness of algorithms P*,* V *and* K*.*

Soundness informally says that, whenever $x \notin L_R$, no computationally bounded prover can convince the verifier into accepting x.

[6] We stress that interactive arguments typically do not require a CRS. Looking ahead, the reason for defining interactive arguments in the CRS model is that our compilers in Section 5 and 6 will produce an interactive argument in the CRS model (to be used in our instantiation of the FS transform).

Definition 3.4 (Soundness) *Let $\Pi = (\mathsf{K}, \mathsf{P}, \mathsf{V})$ be an interactive argument system for a polynomial-time computable relation R. We say that Π satisfies s-soundness if for all PPT algorithms P^*, and for any $x \notin L_R$, we have that*

$$\Pr\left[\langle \mathsf{P}^*, \mathsf{V}\rangle(\mathsf{crs}, x) = 1 : \ \mathsf{crs} \leftarrow_\$ \mathsf{K}(1^\lambda)\right] \overset{asym}{\leq} s(\lambda),$$

where the probability is taken over the randomness of algorithms P^*, V and K.

Completeness and soundness do not quantify how much information an interactive argument reveals about the witness, which in turn can be covered by notions such as witness indistinguishability and zero knowledge. In this paper we will use different flavors of the zero-knowledge property. We will postpone the actual definitions to the place in the paper where they are actually used.

Standard-model interactive arguments. We can cast the case where the interactive argument is in the standard model, i.e., it does not rely on a CRS (which is typically the case), by saying that the algorithm K returns the empty string; similarly P and V do not take the CRS as input (or take an empty string as additional input). When we write $\Pi = (\mathsf{P}, \mathsf{V})$, we denote an interactive argument in the standard model. Adapting the definitions of completeness and soundness to the standard model works by replacing the CRS generation algorithm by an algorithm that outputs the empty string.

Non-interactive arguments. We speak of non-interactive arguments in case the protocol consists of a single message π sent from the prover to the verifier. Non-interactive arguments that satisfy a zero-knowledge property typically require a setup assumption, such as a CRS.[7] Syntactically a non-interactive argument system for a polynomial-time computable relation R consists of three PPT algorithms $\overline{\Pi} := (\mathsf{K}, \mathsf{P}, \mathsf{V})$ specified as follows: (i) Algorithm K takes as input 1^λ and outputs a CRS $\mathsf{crs} \in \{0,1\}^*$; (ii) Algorithm P takes as input (crs, x, w) such that $R(x, w) = 1$ and outputs a proof π; (iii) Algorithm V takes as input (crs, x, π) and outputs a bit indicating whether π is a valid proof for x (under crs) or not.

A non-interactive argument should satisfy three main properties, which are analogous to the definitions of completeness, soundness and zero knowledge for interactive arguments. We define these properties below.

Definition 3.5 (Completeness of non-interactive arguments) *Let $\overline{\Pi} = (\mathsf{K}, \mathsf{P}, \mathsf{V})$ be a non-interactive argument system for a polynomial-time computable relation R. We say that $\overline{\Pi}$ satisfies c-completeness if for all (x, w) such that $R(x, w) = 1$ we have*

$$\Pr\left[\mathsf{V}(\mathsf{crs}, x, \pi) = 1 : \mathsf{crs} \leftarrow_\$ \mathsf{K}(1^\lambda); \pi \leftarrow_\$ \mathsf{P}(\mathsf{crs}, x, w)\right] \overset{asym}{\geq} 1 - c(\lambda),$$

where the probability is taken over the randomness of algorithms P, V and K.

Definition 3.6 (Soundness of non-interactive arguments) *Let $\overline{\Pi} = (\mathsf{K}, \mathsf{P}, \mathsf{V})$ be a non-interactive argument system for a polynomial-time computable relation R. We say that $\overline{\Pi}$ satisfies s-soundness if for all PPT algorithms P^*, and for any $x \notin L_R$, we have that*

$$\Pr\left[\mathsf{V}(\mathsf{crs}, x, \pi) = 1 : \mathsf{crs} \leftarrow_\$ \mathsf{K}(1^\lambda); \pi \leftarrow_\$ \mathsf{P}^*(\mathsf{crs}, x)\right] \overset{asym}{\leq} s(\lambda),$$

where the probability is taken over the randomness of algorithms P^*, V and K.

[7] In particular, assuming a CRS is necessary for obtaining non-interactive zero knowledge [Gol01, Chapter 4].

Computational zero knowledge captures the intuition that an honestly computed non-interactive argument for a statement in the language does not reveal anything beyond the fact that the statement is indeed true. This is formalized through the existence of an efficient simulator that can fake arguments for true statements without knowing a corresponding witness. Below, we consider a variant where indistinguishability holds as long as the distinguisher asks up to q arguments, where q is fixed a priori.

Definition 3.7 (q-Bounded computational zero knowledge) *Let $\overline{\Pi} = (\mathsf{K}, \mathsf{P}, \mathsf{V})$ be a non-interactive argument system for a polynomial-time computable relation R. We say that $\overline{\Pi}$ satisfies q-bounded ϵ-computational zero knowledge if for all binary* PPT *adversaries $\mathcal{A}$ there exists a* PPT *simulator $\mathcal{S} := (\mathcal{S}', \mathcal{S}'')$ such that $\mathsf{rNIZK}^{\overline{\Pi}}_{\mathcal{A}}(\lambda) \approx_\epsilon \mathsf{sNIZK}^{\overline{\Pi}}_{\mathcal{S}',\mathcal{S}''}(\lambda)$, where experiments* rNIZK *and* sNIZK *are defined below*

$\underline{\mathsf{rNIZK}^{\overline{\Pi}}_{\mathcal{A}}(\lambda)}$

$\mathsf{crs} \leftarrow\!\$\, \mathsf{K}(1^\lambda)$

return $\mathcal{A}^{\textsc{Prov}(\mathsf{crs},\cdot,\cdot)}(1^\lambda, \mathsf{crs})$

$\underline{\mathsf{sNIZK}^{\overline{\Pi}}_{\mathcal{S}',\mathcal{S}''}(\lambda)}$

$(\mathsf{crs}, \mathsf{tk}) \leftarrow\!\$\, \mathcal{S}'(1^\lambda)$

return $\mathcal{A}^{\textsc{Simu}(\mathsf{crs},\mathsf{tk},\cdot,\cdot)}(1^\lambda, \mathsf{crs})$

$\underline{\textsc{Prov}(\mathsf{crs}, x, w)}$

if $R(x,w) = 1$ **then**

  return $\pi \leftarrow\!\$\, \mathsf{P}(\mathsf{crs}, x, w)$

else return $\bot$

$\underline{\textsc{Simu}(\mathsf{crs}, \mathsf{tk}, x, w)}$

if $R(x,w) = 1$ **then**

  return $\pi \leftarrow\!\$\, \mathcal{S}''(\mathsf{crs}, \mathsf{tk}, x)$

else return $\bot$

and $\mathcal{A}$ can ask $q(\lambda)$ queries to its oracle. Additionally we say that $\overline{\Pi}$ satisfies unbounded *computational non-interactive zero knowledge if indistinguishability of the above experiments holds for an arbitrary polynomial $q(\lambda)$.*

Note that we quantify over all *binary* adversaries, that is, we consider only adversaries that output a bit. This is without loss of generality, but makes notation in later game-hop proofs easier. For brevity, we sometimes write that $\overline{\Pi}$ is a (c, s, q, ϵ)-NIZK to denote that $\overline{\Pi}$ satisfies c-completeness, s-soundness, and q-bounded ϵ-computational zero knowledge.

3.5 Obfuscation

Indistinguishability obfuscation [BGI+01, BGI+12] intuitively captures that the obfuscation of two functionally equivalent circuits cannot be distinguished. We here give a game-based definition, following the definitional framework of [BST14] which captures indistinguishability obfuscation based notions via the IO security game and a class of samplers Sam.

Definition 3.8 (Obfuscation scheme) *A* PPT *algorithm* O *is called an* obfuscation scheme *if, on input the security parameter 1^λ and a description of a circuit C, it returns (a description of) a circuit $\overline{C}$ such that $\forall x: \; C(x) = \overline{C}(x)$. We call a* PPT *algorithm* Sam *a* circuit sampler *if on input the security parameter 1^λ sampler* Sam *outputs (C_0, C_1, aux) where C_0 and C_1 are descriptions of circuits and* aux *is a string. If* Sam *is a circuit sampler and* O *is an obfuscation scheme we define the advantage $\mathsf{Adv}^{\mathrm{io}}_{\mathsf{O},\mathsf{Sam},\mathsf{D}}(\cdot)$ for a distinguisher* D *relative to game* IO*:*

$$\mathsf{Adv}^{\mathrm{io}}_{\mathsf{O},\mathsf{Sam},\mathsf{D}}(\lambda) := 2 \cdot \Pr\left[\mathsf{IO}^{\mathsf{O}}_{\mathsf{D},\mathsf{Sam}}(\lambda)\right] - 1$$

$\underline{\mathsf{IO}^{\mathsf{O}}_{\mathsf{D},\mathsf{Sam}}(\lambda)}$
$b \leftarrow\!\!\$\ \{0,1\}$
$(C_0, C_1, \mathsf{aux}) \leftarrow\!\!\$\ \mathsf{Sam}(1^\lambda)$
$\overline{C} \leftarrow\!\!\$\ \mathsf{O}(1^\lambda, C_b)$
$b' \leftarrow\!\!\$\ \mathsf{D}(1^\lambda, \overline{C}, \mathsf{aux})$
return $(b = b')$

If $\mathcal{S}$ is a class of circuit samplers, we call an obfuscation scheme O ϵ_{O}-secure for $\mathcal{S}$, if for all $\mathsf{Sam} \in \mathcal{S}$ and all PPT distinguishers D advantage $\mathsf{Adv}^{\mathrm{io}}_{\mathsf{O},\mathsf{Sam},\mathsf{D}}(\lambda) \overset{asym}{\leq} \epsilon_{\mathsf{O}}(\lambda)$.

We can now capture indistinguishability obfuscation via restricting the class of samplers to so-called *equality samplers*. As we only use efficient samplers we can further restrict the class of samplers.

Definition 3.9 (Equality circuit sampler) *We call a PPT algorithm Sam an* equality circuit sampler *if for all security parameters $\lambda \in \mathbb{N}$ it outputs a triple (C_0, C_1, aux) consisting of two circuit descriptions and a string such that with overwhelming probability over the coins of Sam we have that the circuits C_0 and C_1 have the same size, number of inputs and number of outputs and are functionally equivalent, that is*

$$\Pr_{(C_0,C_1,\mathsf{aux}) \leftarrow\!\!\$\ \mathsf{Sam}(1^\lambda)}[|C_0| = |C_1| \wedge \forall x : C_0(x) = C_1(x)] \geq 1 - \mathsf{negl}(\lambda)\,.$$

A beautiful result that we will use is the relationship between differing-inputs obfuscation and indistinguishability obfuscation proved by Boyle, Chung and Pass [BCP14], who show that any general purpose indistinguishability obfuscator is also a differing-inputs obfuscator for circuits that differ only on a few (at most polynomially many) inputs. We first define differing-inputs obfuscation by restricting samplers to be differing-inputs samplers.

Definition 3.10 (Differing-inputs circuit sampler) *Let Sam be a circuit sampler. We call Sam a* differing-inputs circuit sampler *if advantage $\mathsf{Adv}^{\mathrm{diff}}_{\mathsf{Sam},\mathsf{Ext}}(\cdot)$ is negligible for all PPT algorithms Ext and where the advantage is defined as (relative to game Diff on the right):*

$$\mathsf{Adv}^{\mathrm{diff}}_{\mathsf{Sam},\mathsf{Ext}}(\lambda) := \Pr\left[\mathsf{Diff}^{\mathsf{Ext}}_{\mathsf{Sam}}(\lambda)\right]$$

$\underline{\mathsf{Diff}^{\mathsf{Ext}}_{\mathsf{Sam}}(\lambda)}$
$(C_0, C_1, \mathsf{aux}) \leftarrow\!\!\$\ \mathsf{Sam}(1^\lambda)$
$x \leftarrow\!\!\$\ \mathsf{Ext}(1^\lambda, C_0, C_1, \mathsf{aux})$
return $(C_0(x) \neq C_1(x))$

With that we are ready to formulate the result due to Boyle, Chung and Pass [BCP14].

Theorem 3.11 ([BCP14]) *Let iO be an indistinguishability obfuscator for all circuits in P/poly. Let Sam be a differing-inputs circuit sampler for which there exists a polynomial $d : \mathbb{N} \to \mathbb{N}$, such that*

$$\Pr\left[|\{x : C_0(x) \neq C_1(x)\}| \leq d(\lambda) \;\middle|\; (C_0, C_1, \mathsf{aux}) \leftarrow\!\!\$\ \mathsf{Sam}(1^\lambda)\right] \geq 1 - \mathsf{negl}(\lambda)\,.$$

Then iO is a differing-inputs obfuscator for Sam, i.e., obfuscator iO is $\{\mathsf{Sam}\}$-secure.

3.6 Puncturable Pseudorandom Functions

A key ingredient in the compilers are so-called puncturable pseudorandom functions (PRFs) [SW14]. A family of puncturable PRFs is a function family that additionally comes with a PPT *puncturing* algorithm Pntr which on input a polynomial-size set $S \subseteq \{0,1\}^{\mathsf{il}(\lambda)}$, outputs a special key k_S.

Definition 3.12 (Puncturable PRF) *A family of functions* $\mathsf{F} := (\mathsf{F.KGen}, \mathsf{F.Pntr}, \mathsf{F.kl}, \mathsf{F.Eval}, \mathsf{F.il}, \mathsf{F.ol})$ *is called an* ϵ_{prf}*-secure, puncturable PRF if the following holds.*

Functionality preserved under puncturing. *For every* PPT *adversary* $\mathcal{A}$ *such that* $\mathcal{A}(1^\lambda)$ *outputs a polynomial-size set* $S \subseteq \{0,1\}^{\mathsf{F.il}(\lambda)}$*, it holds for all* $x \in \{0,1\}^{\mathsf{F.il}(\lambda)} \setminus S$ *that:*

$$\Pr\Big[\mathsf{F.Eval}(\mathsf{k}, x) = \mathsf{F.Eval}(\mathsf{k}_S, x) : \mathsf{k} \leftarrow\!\!\$\, \mathsf{F.KGen}(1^\lambda), \mathsf{k}_S \leftarrow\!\!\$\, \mathsf{F.Pntr}(\mathsf{k}, S)\Big] = 1.$$

Pseudorandom at punctured points. *For every* PPT *adversary* $(\mathcal{A}_1, \mathcal{A}_2)$*, the advantage* $\mathsf{Adv}^{\mathrm{pprf}}_{\mathsf{F},\mathcal{A}_1,\mathcal{A}_2}(\cdot)$ *is asymptotically smaller than* ϵ_{prf}*, i.e.,*

$$\mathsf{Adv}^{\mathrm{pprf}}_{\mathsf{F},\mathcal{A}_1,\mathcal{A}_2}(\lambda) = 2 \cdot \Pr\Big[\mathsf{pPRF}^{\mathsf{F}}_{\mathcal{A}_1,\mathcal{A}_2}(\lambda)\Big] - 1 \overset{asym}{\leq} \epsilon_{\mathsf{prf}}$$

where game pPRF *is defined as*

$\mathsf{pPRF}^{\mathsf{F}}_{\mathcal{A}_1,\mathcal{A}_2}(\lambda)$

$S \leftarrow \emptyset; b \leftarrow\!\!\$\, \{0,1\}$
$\mathsf{k} \leftarrow\!\!\$\, \mathsf{F.KGen}(1^\lambda)$
$\mathsf{state} \leftarrow\!\!\$\, \mathcal{A}_1^{\textsc{Challenge}}(1^\lambda)$
$\mathsf{k}^* \leftarrow\!\!\$\, \mathsf{F.Pntr}(\mathsf{k}, S)$
$b' \leftarrow\!\!\$\, \mathcal{A}_2(1^\lambda, \mathsf{state}, \mathsf{k}^*)$
return $(b = b')$

$\textsc{Challenge}(x)$

if $x \in S$ **then return** $\bot$
$S \leftarrow S \cup \{x\}$
if $b = 0$ **then**
  $y \leftarrow \mathsf{F.Eval}(\mathsf{k}, x)$
else $y \leftarrow\!\!\$\, \{0,1\}^{\mathsf{F.ol}(\lambda)}$
return y

4 Fiat–Shamir NIZK

We show that under specific assumptions on the underlying protocol, a q-wise independent hash function is enough to instantiate the random oracle in the Fiat–Shamir collapse yielding a secure NIZK with q-bounded computational zero knowledge.

After recalling the standard FS transform in Section 4.1, we present a "selective" interactive variant of the transformation, and establish its completeness and soundness in Section 4.2. Later, in Section 4.3, we put forward three properties of the initial 3PC argument that allow to prove completeness, soundness, and q-bounded computational zero knowledge of the FS-collapse in the standard model; the proof of completeness and soundness reduce directly to the completeness and soundness of the above selective FS transform. Our main theorem is summarized in Section 4.4. Finally, in Section 4.5, we take a closer look at the required properties and discuss how to achieve them (on a high level).

4.1 The Fiat–Shamir Transform

The Fiat–Shamir (FS) transform [FS87] is a generic way to remove interaction from certain argument systems, using a hash function. For the rest of the paper, we consider only interactive arguments consisting of three messages—which we denote by (α, β, γ)—where the first message is sent by the

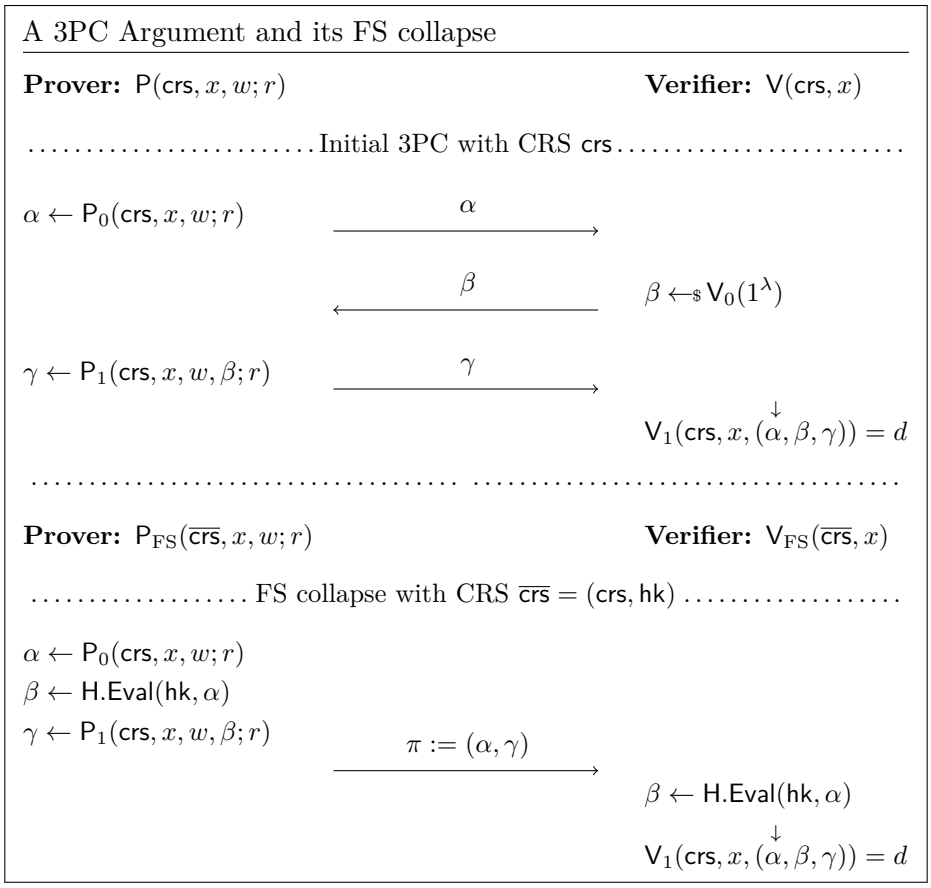


Figure 1: Message flow of a typical 3PC argument system and its corresponding FS collapse. In case the initial 3PC is in the standard model we simply have $\Pi = (\mathsf{P}, \mathsf{V})$ and $\overline{\mathsf{crs}}$ contains only the hash key. Note also that we consider public-coin protocols and thus do not specify the randomness of the verifier (the randomness of V_0 is β and V_1 is deterministic given β).

prover. We also focus on so-called *public-coin* protocols where the verifier's message β is uniformly random over some space $\mathcal{B}$ (e.g., $\beta \in \{0,1\}^k$ for some $k \in \mathbb{N}$). We call this a 3PC argument system for short.

For 3PC arguments it is convenient to think of the prover algorithm as being split into two sub-algorithms $\mathsf{P} := (\mathsf{P}_0, \mathsf{P}_1)$, where P_0 takes as input a pair (x, w) and outputs the prover's first message α (the so-called commitment) and P_1 takes as input (x, w) as well as the verifier's challenge β to produce the prover's second message γ (the so-called response). In general P_0 and P_1 are allowed to share the same random tape, which we denote by $r \in \{0,1\}^*$. In a similar fashion we can think of the verifier's algorithm as split into two sub-algorithms $\mathsf{V} = (\mathsf{V}_0, \mathsf{V}_1)$, where V_0 outputs a uniformly random value $\beta \in \mathcal{B}$ and V_1 is deterministic and corresponds to the verifier's verdict (i.e., V_1 takes as input x and a transcript (α, β, γ) and returns a decision bit $d \in \{0,1\}$).

Non-interactive version. The FS transform allows to remove interaction from any 3PC argument system for a polynomial-time computable relation R as specified below (see also Fig. 1). Let $\Pi = (\mathsf{K}, \mathsf{P}, \mathsf{V})$ be the initial 3PC argument system. Additionally, consider a family of hash functions H consisting of algorithms $\mathsf{H.KGen}$, $\mathsf{H.kl}$, $\mathsf{H.Eval}$, $\mathsf{H.il}$ and $\mathsf{H.ol}$ (see Section 3.1); here $\mathsf{H.il}$ and $\mathsf{H.ol}$ correspond, respectively, to the bit lengths of messages α and β (as a function of the security parameter λ).

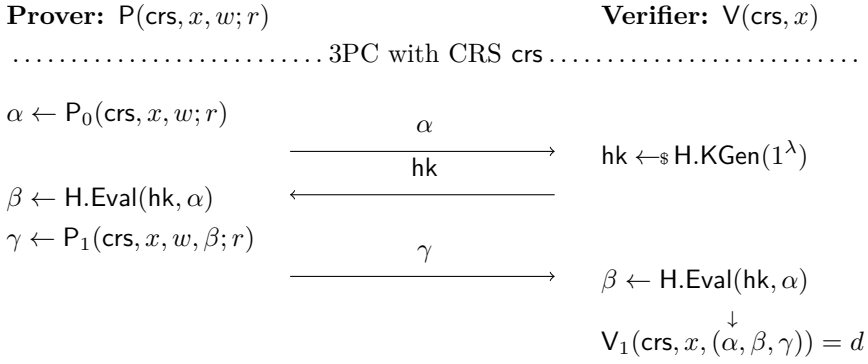
The FS collapse of Π using H is a triple of algorithms $\overline{\Pi}_{\mathrm{FS},\mathsf{H}} := (\mathsf{K}_{\mathrm{FS}}, \mathsf{P}_{\mathrm{FS}}, \mathsf{V}_{\mathrm{FS}})$ defined as follows.

- Algorithm K_{FS} takes as input the security parameter, samples $\mathsf{hk} \leftarrow\!\!\$\, \mathsf{H.KGen}(1^\lambda)$, $\mathsf{crs} \leftarrow\!\!\$\, \mathsf{K}(1^\lambda)$, and publishes $\overline{\mathsf{crs}} := (\mathsf{crs}, \mathsf{hk})$.
- Algorithm P_{FS} takes as input $(\overline{\mathsf{crs}}, x, w)$ and runs $\mathsf{P}_0(\mathsf{crs}, x, w)$ in order to obtain the commitment $\alpha \in \{0,1\}^{\mathsf{H.il}(\lambda)}$; next P_{FS} defines the challenge as $\beta := \mathsf{H.Eval}(\mathsf{hk}, \alpha)$ and runs $\mathsf{P}_1(\mathsf{crs}, x, w, \beta)$ in order to obtain the response γ. Finally P_{FS} outputs $\pi := (\alpha, \gamma)$.
- Algorithm V_{FS} takes as input $(\overline{\mathsf{crs}}, x, \pi)$ and returns 1 if and only if verifier $\mathsf{V}_1(\mathsf{crs}, x, (\alpha, \beta, \gamma)) = 1$ where $\beta = \mathsf{H.Eval}(\mathsf{hk}, \alpha)$.

In a nutshell the result of Fiat and Shamir says that whenever $\Pi = (\mathsf{P}, \mathsf{V})$ is a (standard model) 3PC argument satisfying completeness, computational soundness, and honest-verifier zero knowledge (in addition to a basic requirement on the min-entropy of the prover's commitment), its FS collapse $\overline{\Pi}_{\mathrm{FS},\mathsf{H}}$ is a NIZK argument system if H is modeled as a random oracle.

4.2 A Selective Variant of Fiat–Shamir

As an intermediate step in the proof of soundness of our standard model instantiation of the FS transform, we will consider a selective variant of the FS transform of a 3PC argument system which basically translates into allowing the hash function to depend on the commitment α. Note that this selective variant is still *interactive* since we consider the prover to be split into two algorithms, where the first algorithm is identical to P_0 and the second algorithm first computes β using the received hash key and later runs P_1 in order to obtain γ; similarly the verifier is split into two algorithms, where the first algorithm now generates the hash key (instead of sampling β directly) and the second algorithm is identical to V_1:



Note that the verifier in the above protocol accepts if and only if (α, β, γ) is an accepting proof for x and moreover $\beta = \mathsf{H.Eval}(\mathsf{hk}, \alpha)$. We write $\Pi_{\text{sel-FS},\mathsf{H}}$ for the above selective (interactive) version of the FS transform, and define its completeness and soundness properties below.

Definition 4.1 (Completeness of the selective FS transform) *Let $\Pi = (\mathsf{K}, (\mathsf{P}_0, \mathsf{P}_1), (\mathsf{V}_0, \mathsf{V}_1))$ be a 3PC argument system for a polynomial-time computable relation R, and let $\Pi_{\text{sel-FS},\mathsf{H}}$ be the corresponding selective FS transform using hash function family H. We say that $\Pi_{\text{sel-FS},\mathsf{H}}$ satisfies c-completeness if for all (x, w) such that $R(x, w) = 1$ we have*

$$\Pr\left[\mathsf{V}_1(\mathsf{crs}, x, (\alpha, \beta, \gamma)) = 1 : \begin{array}{l} \mathsf{crs} \leftarrow\!\!\$\, \mathsf{K}(1^\lambda); \alpha \leftarrow\!\!\$\, \mathsf{P}_0(\mathsf{crs}, x, w); \\ \mathsf{hk} \leftarrow\!\!\$\, \mathsf{H.KGen}(1^\lambda); \\ \beta = \mathsf{H.Eval}(\mathsf{hk}, \alpha); \gamma \leftarrow\!\!\$\, \mathsf{P}_1(\mathsf{crs}, x, w, \beta) \end{array}\right] \overset{asym}{\geq} 1 - c(\lambda),$$

where the probability is taken over the randomness of algorithms P_0, P_1, K *and over the choice of the hash key.*

Definition 4.2 (Soundness of the selective FS transform) *Let* $\Pi = (\mathsf{K}, (\mathsf{P}_0, \mathsf{P}_1), (\mathsf{V}_0, \mathsf{V}_1))$ *be a 3PC argument system for a polynomial-time computable relation* R*, and let* $\Pi_{\text{sel-FS},\mathsf{H}}$ *be the corresponding selective FS transform using hash function family* H*. We say that* $\Pi_{\text{sel-FS},\mathsf{H}}$ *satisfies* s*-soundness if for all* PPT *algorithms* $\mathsf{P}^* = (\mathsf{P}_0^*, \mathsf{P}_1^*)$*, and for all* $x \notin L_R$*, we have that*

$$\Pr\left[\mathsf{V}_1(\mathsf{crs}, x, (\alpha, \beta, \gamma)) = 1 : \begin{array}{l} \mathsf{crs} \leftarrow\!\!\$\, \mathsf{K}(1^\lambda); \alpha \leftarrow\!\!\$\, \mathsf{P}_0^*(\mathsf{crs}, x); \\ \mathsf{hk} \leftarrow\!\!\$\, \mathsf{H.KGen}(1^\lambda); \\ \beta = \mathsf{H.Eval}(\mathsf{hk}, \alpha); \gamma \leftarrow\!\!\$\, \mathsf{P}_1^*(\mathsf{crs}, x, \mathsf{hk}, \alpha, \beta) \end{array}\right] \overset{asym}{\leq} s(\lambda),$$

where the probability is taken over the randomness of algorithms P^*, K *and over the choice of the hash key.*

Completeness and soundness for selective FS. We can now move on to state our first result: If H is a 1-wise independent hash function, then the selective FS transform instantiated with H maintains completeness and computational soundness of the starting 3PC argument.[8] Recall that, as discussed in Section 3.3, already the hash function family of constant functions defined as $\mathsf{H}(\mathsf{hk}, x) = \mathsf{hk}$ is 1-wise independent and thus fulfills the requirements of the following theorem.

Theorem 4.3 *Let* $\Pi = (\mathsf{K}, \mathsf{P}, \mathsf{V})$ *be a 3PC argument system for a polynomial-time computable relation* R*, that is* c*-complete and* s*-sound, and let* H *be a* 1*-wise independent hash function. Then, the selective FS transform* $\Pi_{\text{sel-FS},\mathsf{H}}$ *of* Π *using* H *is* c*-complete and* s*-sound for relation* R*.*

Proof. The proof for completeness and soundness follows directly from noting that β is distributed uniformly at random in $\{0,1\}^{\mathsf{H.ol}(\lambda)}$ over the choice of the hash key, and as the hash key is chosen independently of α the proof reduces directly to the completeness and soundness of the interactive version of the underlying 3PC. □

4.3 The FS-Collapse

We now consider the standard FS collapse and discuss each property (completeness, soundness, and zero knowledge) in turn. We reduce soundness and completeness to the soundness and completeness of the *selective FS* transform, and reduce zero knowledge directly to the (instance-independent honest-verifier) zero-knowledge property of the underlying 3PC argument. Instance independence is a new property for protocols that we define in this section.

Note that, for our final theorem, we require the starting 3PC protocol to satisfy three "non-standard" requirements (that we introduce along the way), including for example, the previously mentioned instance-independence property.

4.3.1 Completeness and Soundness

We start by showing that if the underlying 3PC argument satisfies completeness, so does the resulting FS non-interactive argument.

Lemma 4.4 *Let* $\Pi = (\mathsf{K}, \mathsf{P}, \mathsf{V})$ *be a 3PC argument system for a polynomial-time computable relation* R *satisfying* c*-completeness. Then, assuming* H *is a* 1*-wise independent hash function, the FS collapse* $\overline{\Pi}_{\text{FS},\mathsf{H}}$ *of* Π *using* H *satisfies* c*-completeness.*

[8] Note that we do not prove zero knowledge of the selective FS transform; this is because we will later prove directly non-interactive zero knowledge of the FS collapse.

Proof. The proof follows by noting that when both the prover and the verifier of the non-interactive protocol are honest, the probability that the verifier accepts is the same as in the (interactive) selective variant of the FS transform applied to Π. The statement then follows from Theorem 4.3. $\square$

Let $a(\lambda) \in \mathbb{N}$ be the maximum bit-length of the commitment α. To capture soundness of the FS-collapse we need an additional property of the underlying 3PC protocol, namely, a gap between the worst-case probability $2^{-a(\lambda)}$ with which one can guess the first message of the protocol and the soundness error $s(\lambda)$. We can interpret that $s/2^{-a} < 1$ as follows: even if we allow a loss of 2^{-a} to guess the first message, then still there remains a level of security that can be leveraged to obtain soundness. The soundness of our standard-model instantiation depends upon the above ratio, which we call the soundness-error-to-guessing ratio.

Definition 4.5 (Soundness-error-to-guessing ratio) *Let λ be a security parameter, and consider a 3PC argument system $\Pi = (\mathsf{K}, \mathsf{P}, \mathsf{V})$ for a polynomial-time computable relation R with commitments of maximum bit-length $a(\lambda)$ and satisfying $s(\lambda)$-soundness. The soundness-error-to-guessing ratio (SEGR) associated to Π is defined as $\varrho(\lambda) := s(\lambda)/2^{-a(\lambda)}$.*

Armed with a "sub-one" soundness-error-to-guessing ratio we can now quantify the soundness of our instantiation of the FS-collapse.

Lemma 4.6 *Let $\Pi = (\mathsf{K}, \mathsf{P}, \mathsf{V})$ be a 3PC argument system for a polynomial-time computable relation R, with SEGR ϱ and s-soundness. Then, assuming H is a 1-wise independent hash function, the FS collapse $\overline{\Pi}_{\mathrm{FS},\mathsf{H}} = (\mathsf{K}_{\mathrm{FS}}, \mathsf{P}_{\mathrm{FS}}, \mathsf{V}_{\mathrm{FS}})$ of Π using H satisfies ϱ-soundness.*

Proof. Let algorithm $\mathsf{P}^*_{\mathrm{FS}}$ be an adversary for the non-interactive FS collapse. Let $x \notin L_R$ and let

$$\mu(\lambda) := \Pr\left[\mathsf{V}_{\mathrm{FS}}((\mathsf{crs}, \mathsf{hk}), x, (\alpha^*, \gamma^*)) = 1 : \begin{array}{l} \mathsf{crs} \leftarrow\!\!\$\, \mathsf{K}(1^\lambda); \mathsf{hk} \leftarrow\!\!\$\, \mathsf{H.KGen}(1^\lambda); \\ (\alpha^*, \gamma^*) \leftarrow\!\!\$\, \mathsf{P}^*_{\mathrm{FS}}((\mathsf{crs}, \mathsf{hk}), x) \end{array}\right]$$

the advantage of $\mathsf{P}^*_{\mathrm{FS}}$ in breaking soundness of the FS collapse. We show how to use $\mathsf{P}^*_{\mathrm{FS}}$ to construct a malicious prover $\mathsf{P}^* := (\mathsf{P}^*_0, \mathsf{P}^*_1)$ breaking soundness of $\Pi_{\text{sel-FS},\mathsf{H}}$ as follows. The prover P^*_0 picks a value α uniformly at random from the set of all possible commitments, and sends α to the verifier. It gets back a hash-function key hk that is independent from the value α that P^*_0 sent to the verifier in the first message. Now, prover P^*_1 runs prover $\mathsf{P}^*_{\mathrm{FS}}$ on $((\mathsf{crs}, \mathsf{hk}), x)$ to obtain a pair (α^*, γ^*). If $\alpha^* = \alpha$, then P^*_1 passes γ^* to the verifier. Else, P^*_1 aborts.

Observe that the success probability of P^* is lower bounded by the success probability of $\mathsf{P}^*_{\mathrm{FS}}$ times the probability that α^* is equal to α. If the selective FS transform of Π has soundness $s'(\lambda)$ we obtain

$$\mu(\lambda) \cdot 2^{-a(\lambda)} \overset{\text{asym}}{\leq} s'(\lambda) = s(\lambda)$$

where the last equality is due to Theorem 4.3. The statement now follows by a division of the inequality by $2^{-a(\lambda)}$ and by the definition of soundness-error-to-guessing ratio. $\square$

4.3.2 Zero Knowledge

In order to prove zero knowledge we need two additional properties of the underlying 3PC. The first property requires that the prover chooses its commitment α independently of the instance x and the witness w. We call this property *instance-independent commitment.*

Definition 4.7 (Instance-independent commitments) *Let $\Pi = (\mathsf{K}, \mathsf{P} = (\mathsf{P}_0, \mathsf{P}_1), \mathsf{V})$ be a 3PC argument system for a polynomial-time computable relation R. We say that Π has instance-independent commitments if $\mathsf{P}_0(\mathsf{crs}, x, w; r) := \mathsf{P}_0(\mathsf{crs}; r)$ for any choice of randomness r, instance x and witness w.*

Interactive protocols obeying the above requirement are sometimes also known under the name of "input-delayed" protocols [CPS+16a, HV16, CPS+16b].[9] One example of a 3PC protocol that has instance-independent commitments is the 3PC argument due to Blum for the quadratic residuosity [Blu81]. Another example is given by the Lapidot–Shamir protocol for graph Hamiltonicity [LS91].

The second property we need is an analogue to instance-independent commitments for honest-verifier zero-knowledge (HVZK) simulators. Standard HVZK means that a 3PC argument satisfies zero knowledge provided that the verifier is honest; here, we additionally require that the simulator can choose α and β independently of the instance. Note that while instance-independent commitments has nothing to do with the challenge β, it follows from the definition of 3PC protocols that β is chosen independently of the instance by verifier V_0. Additionally, we consider a weaker zero-knowledge guarantee, that is, we require that the adversary can only make q-many oracle queries where q is an arbitrary (but fixed) polynomial. (Looking ahead, this suffices because later we will show a similar q-bounded zero-knowledge guarantee for the NIZK obtained via the FS collapse.)

We define the property in the CRS model and, again, note that the standard-model version of this definition is obtained by replacing the crs with an empty string.

Definition 4.8 (q-Bounded instance-independent HVZK) *Let $\Pi = (\mathsf{K}, \mathsf{P}, \mathsf{V})$ be a 3PC argument system in the CRS model for a polynomial-time computable relation R, with instance-independent commitments. We say that Π satisfies q-bounded instance-independent ϵ-computational honest-verifier zero knowledge (instance-independent (q, ϵ)-HVZK for short) if there exists a PPT simulator $\mathcal{S} := (\mathcal{S}', (\mathcal{S}_0'', \mathcal{S}_1''))$ such that for all* binary PPT *adversaries $\mathcal{A}$ we have that $\mathsf{rIPS}^{\Pi}_{\mathcal{A}}(\lambda) \approx_\epsilon \mathsf{sIPS}^{\Pi}_{\mathcal{S}',(\mathcal{S}_0'',\mathcal{S}_1''),\mathcal{A}}(\lambda)$, where experiments rIPS and sIPS are defined below:*

$\underline{\mathsf{rIPS}^{\Pi}_{\mathcal{A}}(\lambda)}$

for $i = 1, \ldots, q(\lambda)$ **do**
  $\mathbf{r}_{\mathsf{P}}[i] \leftarrow\$ \{0,1\}^{\mathsf{P}.\mathsf{rl}(\lambda)}$
  $\mathbf{r}_{\mathsf{V}}[i] \leftarrow\$ \{0,1\}^{\mathsf{V}.\mathsf{rl}(\lambda)}$
$\mathsf{crs} \leftarrow\$ \mathsf{K}(1^\lambda)$
return $\mathcal{A}^{\textsc{Prov}(\mathbf{r}_{\mathsf{P}}, \mathbf{r}_{\mathsf{V}}, \mathsf{crs}, \cdot, \cdot, \cdot, \cdot, \cdot)}(1^\lambda, \mathsf{crs})$

$\underline{\mathsf{sIPS}^{\Pi}_{\mathcal{S}',(\mathcal{S}_0'',\mathcal{S}_1''),\mathcal{A}}(\lambda)}$

for $i = 1, \ldots, q(\lambda)$ **do**
  $\mathbf{r}_{\mathcal{S}}[i] \leftarrow\$ \{0,1\}^{\mathcal{S}.\mathsf{rl}(\lambda)}$
$(\mathsf{crs}, \mathsf{tk}) \leftarrow\$ \mathcal{S}'(1^\lambda)$
return $\mathcal{A}^{\textsc{Simu}(\mathbf{r}_{\mathcal{S}}, \mathsf{crs}, \mathsf{tk}, \cdot, \cdot, \cdot, \cdot, \cdot)}(1^\lambda, \mathsf{crs})$

$\underline{\textsc{Prov}(\mathbf{r}_{\mathsf{P}}, \mathbf{r}_{\mathsf{V}}, \mathsf{crs}, x, w, b, i)}$

if $R(x, w) = 0 \vee i \notin [q(\lambda)]$ **then**
  return $\bot$
if $T[i, b] \neq \bot$ **then**
  return $T[i, b]$
if $b = 0$ **then**
  $\alpha \leftarrow \mathsf{P}_0(\mathsf{crs}; \mathbf{r}_{\mathsf{P}}[i])$
  $\beta \leftarrow \mathsf{V}_0(\mathsf{crs}; \mathbf{r}_{\mathsf{V}}[i])$ // *i.e.* $\beta = \mathbf{r}_{\mathsf{V}}[i]$
  $\gamma \leftarrow \bot$
else
  $\alpha \leftarrow \mathsf{P}_0(\mathsf{crs}; \mathbf{r}_{\mathsf{P}}[i])$
  $\beta \leftarrow \mathsf{V}_0(\mathsf{crs}; \mathbf{r}_{\mathsf{V}}[i])$ // *i.e.* $\beta = \mathbf{r}_{\mathsf{V}}[i]$
  $\gamma \leftarrow \mathsf{P}_1(\mathsf{crs}, x, w; \mathbf{r}_{\mathsf{P}}[i])$
$T[i, b] \leftarrow (\alpha, \beta, \gamma)$
return (α, β, γ)

$\underline{\textsc{Simu}(\mathbf{r}_{\mathcal{S}}, \mathsf{crs}, \mathsf{tk}, x, w, b, i)}$

if $R(x, w) = 0 \vee i \notin [q(\lambda)]$ **then**
  return $\bot$
if $T[i, b] \neq \bot$ **then**
  return $T[i, b]$
if $b = 0$ **then**
  $(\alpha, \beta) \leftarrow \mathcal{S}_0''(\mathsf{crs}, \mathsf{tk}; \mathbf{r}_{\mathcal{S}}[i])$
  $\gamma \leftarrow \bot$
else
  $(\alpha, \beta, \gamma) \leftarrow \mathcal{S}_1''(\mathsf{crs}, \mathsf{tk}, x; \mathbf{r}_{\mathcal{S}}[i])$
$T[i, b] \leftarrow (\alpha, \beta, \gamma)$
return (α, β, γ)

[9] While the term "input-delayed" captures the point that the input (i.e., instance x and witness w) is only needed in the last round of the prover's computation, our term "instance-independent commitment" stresses the fact that the commitment is independent of both instance and witness. As this much better captures the role the property plays in this work, we decided to rename the property.

Adversary $\mathcal{A}$ can ask $q(\lambda)$ queries to its oracle and outputs a single bit. Additionally we say that Π satisfies instance-independent ϵ-HVZK if indistinguishability of the above experiments holds for an arbitrary polynomial $q(\lambda)$.

Note that a standard q-bound variant (i.e., without instance independence) is obtained by fixing bit b in oracles PROV and SIMU to 1 and provide P_0 with x and w as additional input, as then the oracles either return a complete honest transcript or a complete simulated transcript. One example of a protocol readily satisfying (unbounded) instance-independent HVZK is given by the Lapidot–Shamir protocol for graph Hamiltonicity [LS91].

We are now in a position to quantify the zero-knowledge property of our instantiation of the FS collapse.

Lemma 4.9 *Let $\Pi = (\mathsf{K}, \mathsf{P}, \mathsf{V})$ be a 3PC argument system for a polynomial-time computable relation R, such that Π has instance-independent commitments and satisfies q-bounded instance-independent ϵ-HVZK. Then, assuming H is a programmable q-wise independent hash function, the FS collapse $\overline{\Pi}_{\mathrm{FS},\mathsf{H}} = (\mathsf{K}_{\mathrm{FS}}, \mathsf{P}_{\mathrm{FS}}, \mathsf{V}_{\mathrm{FS}})$ of Π using H satisfies q-bounded ϵ-computational zero knowledge.*

Proof. For ease of notation let us write $\overline{\Pi} := \overline{\Pi}_{\mathrm{FS},\mathsf{H}}$. We start with the real distribution $\mathsf{rNIZK}_{\mathcal{A}}^{\overline{\Pi}}(\lambda)$, where the CRS is defined as $\overline{\mathsf{crs}} = (\mathsf{crs}, \mathsf{hk})$ for $\mathsf{crs} \leftarrow\!\!\$\, \mathsf{K}(1^\lambda)$ and $\mathsf{hk} \leftarrow\!\!\$\, \mathsf{H.KGen}(1^\lambda)$. Here the adversary $\mathcal{A}$, given the CRS, can ask q adaptive queries (x_i, w_i) to oracle PROV which replies with $\pi \leftarrow\!\!\$\, \mathsf{P}_{\mathrm{FS}}(\overline{\mathsf{crs}}, x_i, w_i)$ (provided that $R(x_i, w_i) = 1$).

We describe a sequence of intermediate hybrid experiments and argue indistinguishability via a standard hybrid argument. The hybrids are depicted in Fig. 2 and are described below. The arrows and accompanying labels (both in Fig. 2 and in the following textual description) hint at the reduction target for showing that two games (or hybrids) are computationally close. That is, in the following, the two hybrids are close down to the q-wise independence of hash function H.

$\mathsf{rNIZK}_1(\lambda)$**:** This is identical to the real experiment, but now the randomness r_i used to generate the q proofs π_i corresponding to $\mathcal{A}$'s queries is pre-sampled. Additionally, values $\alpha_i = \mathsf{P}_0(1^\lambda; r_i)$ are pre-computed—note that this is possible because of instance-independent commitments—which allows us to also pre-compute values β_i as $\beta_i = \mathsf{H.Eval}(\mathsf{hk}, \alpha_i)$ where hk is the hash key. All of these values are stored in a trapdoor $\overline{\mathsf{tk}}$ which is given to the hybrid oracle (which is now a mixture between PROV and SIMU). We write $\mathsf{P.rl}$ for the length of the random tape required for $(\mathsf{P}_0, \mathsf{P}_1)$. Note that the PROV oracle now additionally takes as input $\overline{\mathsf{tk}} = (\boldsymbol{\beta}, r_1, \ldots, r_q)$ and uses r_i as random tape of both P_0 and P_1 (and thus of P_{FS}).

Observe that all of the above steps (pre-computing values) are clearly just syntactical changes, and thus $\mathsf{rNIZK}_{\mathcal{A}}^{\overline{\Pi}}(\lambda) \equiv \mathsf{rNIZK}_1(\lambda)$.

q-wise

$\mathsf{rNIZK}_2(\lambda)$**:** In the second and last step we replace all values β_i with uniformly random values sampled from the range of hash function H and the key hk is chosen via programming the hash function. Down to the programmable q-wise independence property of H the distribution corresponding to $\mathsf{rNIZK}_1(\lambda)$ and $\mathsf{rNIZK}_2(\lambda)$ are identical, i.e., $\mathsf{rNIZK}_2(\lambda) \equiv \mathsf{rNIZK}_1(\lambda)$.

Simulator. Let $\mathcal{S} = (\mathcal{S}', \mathcal{S}'' = (\mathcal{S}''_0, \mathcal{S}''_1))$ be the instance-independent HVZK simulator for the underlying 3PC protocol (cf. Definition 4.8). Let q denote a bound on the number of oracle queries of adversary $\mathcal{A}$. We write $\mathcal{S}''.\mathsf{rl}$ to denote the length of the random tape required for algorithm $\mathcal{S}''$.

We construct simulator $\mathsf{Sim} = (\mathsf{Sim}', \mathsf{Sim}'')$ for q-bounded computational zero knowledge (cf. Definition 3.7). Simulator Sim' first runs $\mathcal{S}'$ in order to obtain a pair $(\mathsf{crs}, \mathsf{tk})$. Afterwards Sim'

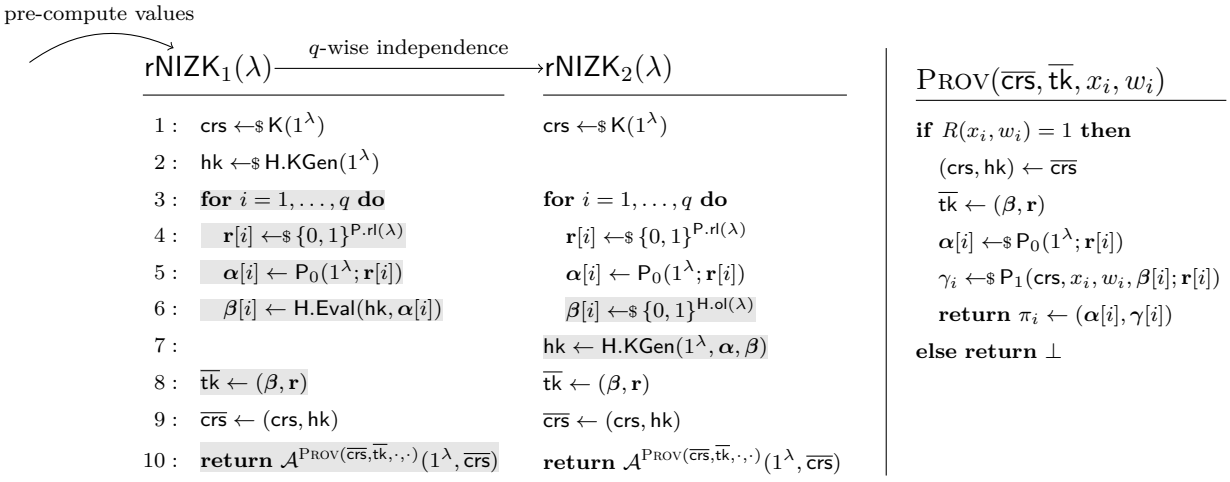


Figure 2: The game hops needed for the proof of Lemma 4.9. The highlighted lines mark those lines that change from step to step. By $\mathsf{H.KGen}(1^\lambda, \boldsymbol{\alpha}, \boldsymbol{\beta})$ in line 7 of rNIZK_2 we denote the "programming" of the q-wise independent hash function such that $\mathsf{H.Eval}(\mathsf{hk}, \alpha[i]) = \beta[i]$ for all $i \in [q]$. Note that by choosing H to be a $(q-1)$ degree polynomial over an appropriate finite field we can do the programming via polynomial interpolation.

$\mathsf{Sim}'(1^\lambda)$

$(\mathsf{crs}, \mathsf{tk}) \leftarrow\!\!\$\ \mathcal{S}'(1^\lambda)$
for $i = 1, \ldots, q(\lambda)$ **do**
  $\mathbf{r}[i] \leftarrow\!\!\$\ \{0,1\}^{\mathcal{S}''.\mathsf{rl}(\lambda)}$
  $(\boldsymbol{\alpha}[i], \boldsymbol{\beta}[i]) \leftarrow \mathcal{S}''_0(1^\lambda; \mathbf{r}[i])$
$\mathsf{hk} \leftarrow\!\!\$\ \mathsf{H.KGen}(1^\lambda, \boldsymbol{\alpha}, \boldsymbol{\beta})$
$\overline{\mathsf{crs}} := (\mathsf{crs}, \mathsf{hk})$
$\overline{\mathsf{tk}} := (\mathsf{tk}, \mathbf{r})$
return $(\overline{\mathsf{crs}}, \overline{\mathsf{tk}})$

$\mathsf{Sim}''(\overline{\mathsf{crs}}, \overline{\mathsf{tk}}, x, i)$

if $i \notin [q(\lambda)]$ **then**
  return $\perp$
if $T[i] \neq \perp$ **then**
  return $T[i]$
$(\mathsf{crs}, \mathsf{hk}) \leftarrow \overline{\mathsf{crs}}$
$\overline{\mathsf{tk}} \leftarrow (\mathsf{tk}, \mathbf{r})$
$(\alpha, \beta, \gamma) \leftarrow \mathcal{S}''(\mathsf{crs}, \mathsf{tk}, x; \mathbf{r}[i])$
$T[i] \leftarrow (\alpha, \beta, \gamma)$
return (α, β, γ)

Figure 3: The HVZK simulator for the proof of Lemma 4.9.

chooses q random strings $r_1, \ldots, r_q$ of length $|r_i| = \mathcal{S}''.\mathsf{rl}(\lambda)$. It then runs $\mathcal{S}''_0$ on randomness r_i to obtain q pairs (α_i, β_i), and chooses a "programmed" hash key $\mathsf{hk} \leftarrow\!\!\$\ \mathsf{H.KGen}(1^\lambda, \boldsymbol{\alpha}, \boldsymbol{\beta})$. It outputs key hk together with crs as common reference string $\overline{\mathsf{crs}}$ for the FS collapse, and tk together with the list of all r_i's as trapdoor $\overline{\mathsf{tk}}$. It thus perfectly simulates the setup in rNIZK_2.

Upon input $\overline{\mathsf{crs}}$ and $\overline{\mathsf{tk}}$, together with the i-th query x_i, simulator Sim'' extracts randomness r_i from tk and calls $\mathcal{S}''$ on input $(\mathsf{crs}, \mathsf{tk}, x_i)$ and with random coins r_i to obtain a proof $\pi_i = (\alpha, \beta, \gamma)$ which it returns. We give the pseudocode of $\mathsf{Sim} = (\mathsf{Sim}', \mathsf{Sim}'')$ in Figure 3. Note that the adversary can only make q queries to its oracle and we provide index i as explicit input to simulator Sim''.

Analysis. Note that simulator Sim' can pre-compute the values α_i using the HVZK simulator $\mathcal{S}''_0$ of the underlying 3PC protocol. This is because $\mathcal{S}''$ is instance-independent. It remains to show that for all q-query adversaries $\mathcal{A}$, the distributions $\mathsf{rNIZK}_2(\lambda)$ and $\mathsf{sNIZK}_{\mathsf{Sim}'\mathsf{Sim}''}(\lambda)$ are computationally close which follows by q-bounded HVZK of the initial 3PC protocol. For this note that games $\mathsf{rNIZK}_2(\lambda)$ and $\mathsf{sNIZK}_{\mathsf{Sim}'\mathsf{Sim}''}(\lambda)$ differ only in how values α, β and γ are computed. In game

$\mathsf{rNIZK}_2(\lambda)$ they are computed with the honest prover, while in game $\mathsf{sNIZK}_{\mathsf{Sim}'\mathsf{Sim}''}(\lambda)$ they are computed using the instance-independent HVZK simulator $\mathcal{S}$. Hence, an adversary against the instance-independent HVZK property of the underlying protocol can perfectly simulate games $\mathsf{rNIZK}_2(\lambda)$ and $\mathsf{sNIZK}_{\mathsf{Sim}'\mathsf{Sim}''}(\lambda)$, by running the steps of $\mathsf{rNIZK}_2(\lambda)$ and using its oracle to obtain α_i, β_i and later using its oracle to complete proofs and obtain γ_i. If the adversary is connected to PROV it perfectly simulates game $\mathsf{rNIZK}_2(\lambda)$ and otherwise it perfectly simulates $\mathsf{sNIZK}_{\mathsf{Sim}'\mathsf{Sim}''}(\lambda)$. Thus, if the underlying protocol is instance-independent (q, ϵ)-HVZK, we obtain

$$|\Pr[\mathsf{rNIZK}_2(\lambda) = 1] - \Pr[\mathsf{sNIZK}_{\mathsf{Sim}'\mathsf{Sim}''}(\lambda) = 1]| \overset{\text{asym}}{\leq} \epsilon(\lambda). \qquad \square$$

4.4 Putting it Together

Combining the results in the previous section we obtain the following theorem stating that the FS transform is instantiable with a q-wise independent hash function given that the underlying 3PC satisfies three additional properties.

Theorem 4.10 *Let $\Pi = (\mathsf{K}, \mathsf{P}, \mathsf{V})$ be a 3PC argument system for a polynomial-time computable relation R. Let H be a programmable q-wise independent hash function. Assume that Π is c-complete and s-sound and additionally satisfies the following three properties:*

- ***(P1)*** *instance-independent commitment;*
- ***(P2)*** *SEGR $\varrho < 1$;*
- ***(P3)*** *instance-independent (q, ϵ)-HVZK.*

Then, the FS collapse $\overline{\Pi}_{\mathrm{FS},\mathsf{H}}$ of Π using H is a $(c, \varrho, q, \epsilon)$-NIZK for the relation R.

4.5 Obtaining the Required Properties

It remains the question of which 3PC arguments (if any) satisfy properties **P1**-**P3**. While we know of protocols directly satisfying **P1** (e.g., the Blum protocol [Blu81]), and of at least one candidate satisfying both **P1** and **P3** (i.e., the Lapidot–Shamir protocol [LS91]), we do not know any 3PC argument already satisfying all properties.

Hence, we turn to the question how to compile a 3PC argument into one satisfying all the properties we need. We do so using two compilers, as outlined below:

- Given a 3PC argument satisfying **P1** and **P3**, we show a compiler yielding a 3PC argument that additionally satisfies **P2**, that is, it has a small soundness-error-to-guessing ratio while retaining properties **P1** and **P3**. This compiler requires a CRS and relies heavily on indistinguishability obfuscation (see Section 3.5), and is presented in details in Section 5.

- Given a 3PC argument satisfying **P1** and (a variant of) HVZK (properties, for example, present in the classical 3PC argument for quadratic residuosity due to Blum [Blu81]), we show a compiler yielding a 3PC argument that additionally satisfies **P3**, that is, it has instance-independent HVZK while retaining property **P1**. This compiler—which is presented in details in Section 6—also requires a CRS, and is inspired by the recent work of Lindell [Lin15]. It relies on so-called *dual-mode commitments* [CV05, CV07] which can be set up either to be perfectly binding or to be equivocal.

  Intuitively, the protocol is changed such that the prover, instead of sending α, sends a commitment c to α which it opens in the last message. As in an honest setup the commitment

is perfectly binding, soundness and completeness follow easily; for zero knowledge the simulator can setup the commitment scheme such that it is equivocal, which allows it to choose its first message as a simulated commitment and later open this to the message α as obtained by the underlying simulator. Note that, in particular, this allows the simulator to choose its first message independently of the instance x.

Open questions are whether we can similarly find compilers that do not require a CRS and whether there exist compilers also for protocols which do not already satisfy **P1** and HVZK.

5 Obtaining Small Soundness-Error-to-Guessing Ratio

In this section we present a compiler that turns a 3PC argument (possibly in the CRS model) with instance-independent commitments and instance-independent (q-bounded) HVZK (Definition 4.7) into a 3PC argument which has the soundness-error-to-guessing ratio (Definition 4.5) needed for the complexity leveraging in Lemma 4.6. We note that the resulting protocol will be in the CRS model regardless whether the starting protocol is in the CRS model or in the standard model. The idea for the compiler is to provide a mechanism that allows to produce many challenges β given only a single commitment α. To this effect the CRS will contain two obfuscated circuits to help the prover and the verifier run the protocol. For obfuscation we use an indistinguishability obfuscator (see Section 3.5). The first circuit C_0 is used by the prover to generate a *pre-commitment* α^* which it sends over to the verifier. The verifier will then use the second circuit C_1 and run it on α^* to obtain multiple commitments. For this $C_1[\mathsf{k}, \mathsf{crs}]$ has a PRF key and the crs for algorithm P_0 of the underlying protocol hardcoded, and computes ℓ commitments as follows:

$C_1[\mathsf{k}, \mathsf{crs}](\alpha^*)$

for $i = 1, \ldots, \ell$ **do**
  $r^* \leftarrow \mathsf{F.Eval}(\mathsf{k}, \alpha^* + i)$
  $\boldsymbol{\alpha}[i] \leftarrow \mathsf{P}_0(\mathsf{crs}; r^*)$
return $\boldsymbol{\alpha}$

Using C_1 the compiled verifier V^* can generate ℓ real commitments $\boldsymbol{\alpha}[1]$ to $\boldsymbol{\alpha}[\ell]$ given the single (short) pre-commitment α^*. The verifier will then run the underlying verifier V on all these commitments to receive $\beta_1, \ldots, \beta_\ell$ which it sends back to the prover.

In order to correctly continue the prover's computation (which was started on the verifier's side) the compiled prover P^* needs to somehow obtain the randomnesses r^* used within C_1. For this, we will build a backdoor into C_1 which allows to obtain the randomness r^* if one knows the randomness that was used to generate α^*. Once the prover has recovered randomnesses $r_1^*, \ldots, r_\ell^*$ it can run the underlying prover P on this randomness and the corresponding challenges β_i to get correct values γ_i which it sends back to the verifier. In a final step, verifier V^* runs the original verifier on the implicit transcripts $(\alpha_i, \beta_i, \gamma_i)_{i=1,\ldots,\ell}$ and returns 1 if and only if the original verifier returns 1 on all the transcripts.

We will next present a formal description of the compiler and then show that it achieves the claimed properties and retains soundness, completeness and zero knowledge.

5.1 The Compiler

Let $\Pi = (\mathsf{K}, \mathsf{P}, \mathsf{V})$ be a 3PC argument system where the prover generates instance-independent commitments and that satisfies instance-independent HVZK. Let rl denote an upper bound on the

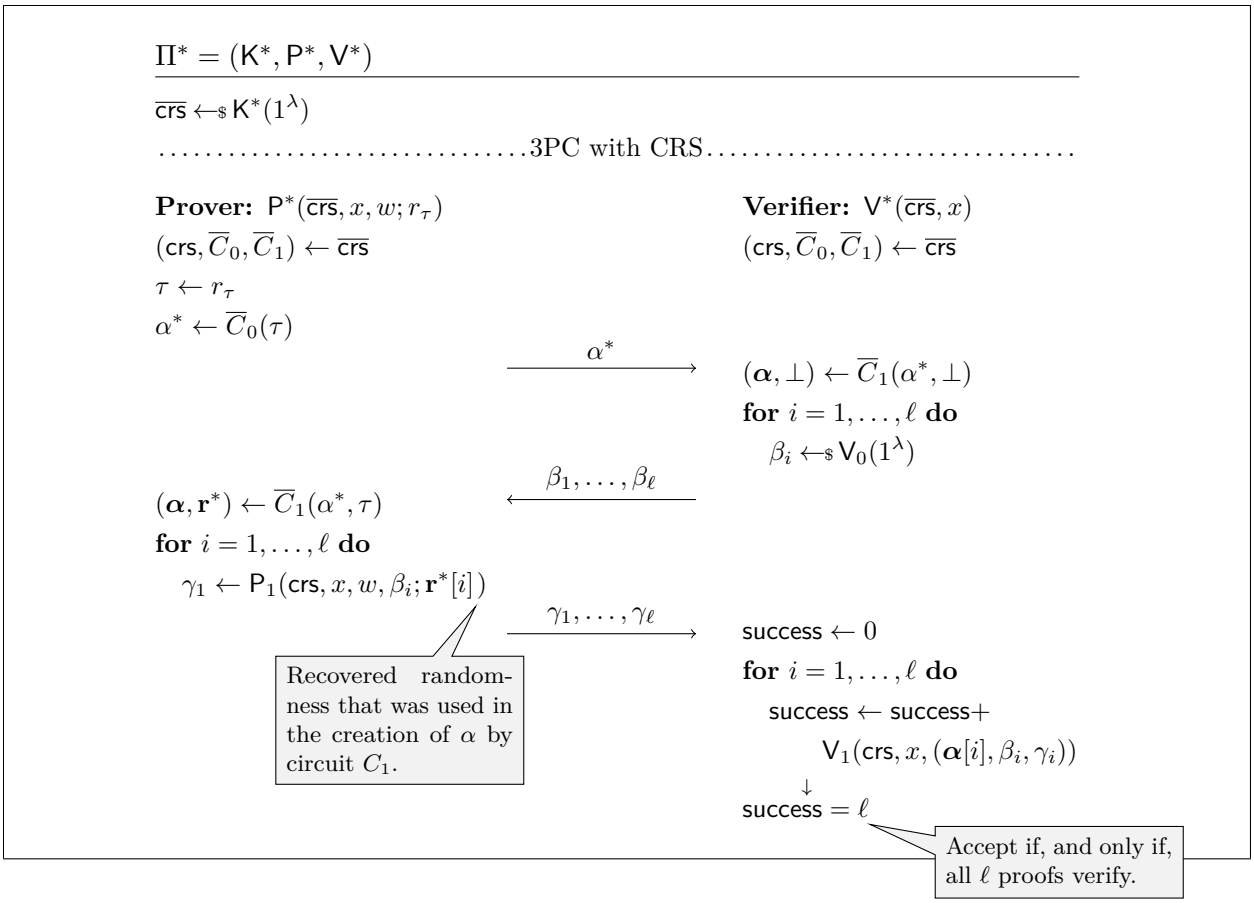


Figure 4: The compiled protocol from Section 5.1 to turn a 3PC protocol into one that has a small soundness-error-to-guessing ratio (in the CRS model).

randomness used by the prover (i.e., $\mathsf{P}.\mathsf{rl}$) and HVZK simulator (i.e., $\mathcal{S}.\mathsf{rl}$). Let F_1 be a puncturable pseudorandom function which is length doubling. Let F_2 be a puncturable pseudorandom function with $\mathsf{F}_2.\mathsf{il} = \mathsf{F}_1.\mathsf{ol}$ and with $\mathsf{F}_2.\mathsf{ol} = \mathsf{rl}$. Let ℓ be a polynomial. We construct an argument system $\Pi^* = (\mathsf{K}^*, \mathsf{P}^*, \mathsf{V}^*)$ in the CRS model as follows. On input the security parameter, K^* will construct an obfuscation of the following two circuits:

$\underline{\mathsf{K}^*(1^\lambda)}$

$\mathsf{crs} \leftarrow\!\!\$\ \mathsf{K}(1^\lambda)$
$\mathsf{k}_1 \leftarrow\!\!\$\ \mathsf{F}_1.\mathsf{KGen}(1^\lambda)$
$\mathsf{k}_2 \leftarrow\!\!\$\ \mathsf{F}_2.\mathsf{KGen}(1^\lambda)$
$\overline{C}_0 \leftarrow\!\!\$\ \mathsf{iO}(C_0[\mathsf{k}_1])$
$\overline{C}_1 \leftarrow\!\!\$\ \mathsf{iO}(C_1[\mathsf{k}_1, \mathsf{k}_2, \ell, \mathsf{crs}])$
$\overline{\mathsf{crs}} \leftarrow (\mathsf{crs}, \overline{C}_0, \overline{C}_1)$
return $\overline{\mathsf{crs}}$

$\underline{C_0[\mathsf{k}_1](\tau)}$

$\alpha^* \leftarrow \mathsf{F}_1.\mathsf{Eval}(\mathsf{k}_1, \tau)$
return α^*

$\underline{C_1[\mathsf{k}_1, \mathsf{k}_2, \ell, \mathsf{crs}](\alpha^*, \tau)}$

for $i = 1, \ldots, \ell$ **do**
$\mathbf{r}^*[i] \leftarrow \mathsf{F}_2.\mathsf{Eval}(\mathsf{k}_2, \alpha^* + i)$
$\boldsymbol{\alpha}[i] \leftarrow \mathsf{P}_0(\mathsf{crs}; \mathbf{r}^*[i])$
if $\alpha^* \neq \mathsf{F}_1.\mathsf{Eval}(\mathsf{k}_1, \tau)$ **then**
$\mathbf{r}^*[i] \leftarrow \perp$
return $(\boldsymbol{\alpha}, \mathbf{r}^*)$

Note that we assume that the underlying protocol is in the CRS model and has a setup algorithm K. If this is not the case, one recovers the transformation for a 3PC in the standard model by assuming that K outputs the empty string ε. The compiled 3PC $\Pi^* = (\mathsf{K}^*, \mathsf{P}^*, \mathsf{V}^*)$ is then constructed as in Figure 4.

5.2 Security Analysis

It remains to show that the compiled protocol is computationally sound, achieves (bounded) instance-independent HVZK, is complete, and that it has instance-independent commitments and a sufficient soundness-error-to-guessing ratio:

Theorem 5.1 *Let $\Pi = (\mathsf{K}, \mathsf{P}, \mathsf{V})$ be a 3PC argument system for a polynomial-time computable relation R such that Π is c-complete and s-sound, has instance-independent commitments, and satisfies q-bounded instance-independent HVZK. Let iO be an indistinguishability obfuscator and F_1 and F_2 puncturable pseudorandom functions. Let ℓ be a polynomial. Then, in the CRS model, the compiled protocol $\Pi^* = (\mathsf{K}^*, \mathsf{P}^*, \mathsf{V}^*)$ is $(\ell \cdot c + \mathsf{negl}(\lambda))$-complete, $(2 \cdot s^\ell + 2^{\mathsf{F}_1.\mathsf{ol}(\lambda)}(s^\ell + \mathsf{negl}(\lambda))$-sound, and satisfies q/ℓ-bounded instance-independent HVZK. Furthermore, the compiled protocol has instance-independent commitments and a negligible SEGR, as long as $\mathsf{F}_1.\mathsf{ol}(\lambda) \in \mathsf{polylog}(\lambda)$.*

We discuss each of these properties in turn.

Completeness. Consider an honest protocol execution of the compiled protocol that does not end with an accepting vote of the verifier. As the final verification uses the underlying verifier and also α, β, and γ are constructed using the underlying algorithms P and V, this thus yields also an honest protocol execution of the underlying protocol where the verifier does not accept. There are two things to note: the compiled protocol "internally" runs the underlying protocol multiple times (ℓ times), and algorithm P_0 is run on pseudorandom coins rather than on truly random coins. Thus, if the underlying protocol is c-complete, the compiled protocol is at most $(\ell \cdot c)$-complete plus the (negligible) distinguishing probability for the pseudorandom function.

Soundness. We analyze the soundness in two steps: Firstly, if the original protocol has soundness s, then its ℓ-parallel repetition version has soundness $s^\ell + \mathsf{negl}(\lambda)$, as the protocol is public-coin [PV07, HPWP10, CL09, CP15]. In a second step, we consider that $\alpha[1], \ldots, \alpha[\ell]$ are generated via applying the first stage of the prover P_0 to the output of an obfuscated pPRF rather than using truly uniformly random coins. We loose a factor of $2^{-|\alpha^*|}$, where $|\alpha^*| = \mathsf{F}_1.\mathsf{ol}(\lambda)$, for the number of choices that the prover can make for the input to the obfuscated pPRF and else reduce to iO and the pPRF via a standard puncturing argument. Hence, when using iO that is $2^{-|\alpha^*|\ell \log s}$-secure, and using a pPRF that is $2^{-|\alpha^*|\ell \log s}$-secure, adding up soundness error $s^\ell + \mathsf{negl}(\lambda)$, iO-security $2^{-|\alpha^*|\ell \log s}$ and pPRF-security $2^{-|\alpha^*|\ell \log s}$ and multiplying them all by $2^{|\alpha^*|}$, we obtain that the soundness of the compiled protocol is $2 \cdot s^\ell + 2^{|\alpha^*|}(s^\ell + \mathsf{negl}(\lambda))$.

Soundness-error-to-guessing ratio. Note that the commitment of the compiled protocol α^* is generated by evaluating pseudorandom function F_1 on a random value τ. This gives us $a^*(\lambda) = \mathsf{F}_1.\mathsf{ol}(\lambda)$ (where $a^*(\lambda)$ is the maximum bit-length of the commitment associated with the compiled protocol Π^*). In particular, since this value is independent of the soundness amplification parameter ℓ we obtain for the SEGR:

$$\varrho(\lambda) = s^*(\lambda)/2^{-a^*(\lambda)} = 2 \cdot 2^{a^*} \cdot s^\ell + 2^{2a^*}(s^\ell + \mathsf{negl}(\lambda)).$$

When choosing $\mathsf{F}_1.\mathsf{ol}(\lambda) \in \mathsf{polylog}(\lambda)$ it follows that SEGR becomes negligible, i.e., that $\varrho(\lambda) \in \mathsf{negl}(\lambda)$.

Instance independence. The compiled protocol has instance independent commitments (see Definition 4.7). Furthermore, the protocol retains instance-independent HVZK simulators as we discuss below.

Zero knowledge. Finally, we show that the compiled protocol satisfies instance-independent q/ℓ-bounded HVZK if the underlying protocol satisfies instance-independent q-bounded HVZK (see Definition 4.8). We show the proof for the simplified setting where the amplification parameter ℓ is set to 1. The proof for the general case is analogous. For the argument we assume the existence of an injective one-way function owf with domain $\mathsf{owf.il} = \mathsf{F}_1.\mathsf{il}$. The one-way function, as it will be used can be regarded as a very simple point obfuscation scheme. To "obfuscate" a point x we store $\overline{p}_x \leftarrow \mathsf{owf}(x)$ which allows us to later compare a point x' by simply checking if $\overline{p}_x = \mathsf{owf}(x')$.

Let $\Pi = (\mathsf{K}, \mathsf{P}, \mathsf{V})$ be a 3PC protocol which satisfies the conditions for the compiler and which is instance-independent HVZK. We need to show that the compiled protocol $\Pi^* = (\mathsf{K}^*, \mathsf{P}^*, \mathsf{V}^*)$ achieves bounded instance-independent HVZK in the CRS model. Towards this, we will consider several intermediate hybrid experiments, where the first game is the real world setting rIPS and the last game is identical to the simulated setting sIPS (with a simulator to be defined). We first describe the individual games and present the accompanying pseudocode in Figures 7, 8, and 9 (starting on page 49). We present a formal analysis of the individual game hops after the description of all games and denote the reduction target for each step next to the game description.

$\mathsf{Game}_1(\lambda)$**:** The game is identical to the real world setting rIPS where the crs is honestly generated by K^* and the adversary, on querying oracle PROV on an instance x in the language with witness w, obtains a transcript of an honest execution between prover $\mathsf{P}^*(1^\lambda, \mathsf{crs}, x, w)$ and verifier $\mathsf{V}^*(1^\lambda, \mathsf{crs}, x)$. Without loss of generality we assume that prover P^* on the i-th query uses random coins τ_i sampled uniformly at random from $\{0,1\}^{\mathsf{P.rl}(\lambda)} = \{0,1\}^{\mathsf{F}_1.\mathsf{il}(\lambda)}$.

iO

$\mathsf{Game}_2(\lambda)$**:** The game is identical to the previous game except that setup K^* now punctures k_1 on all values τ. For this, it chooses q random values $\tau_1, \ldots, \tau_q$ in $\{0,1\}^{\mathsf{F}_1.\mathsf{il}(\lambda)}$ (if the values are not distinct we abort and define that the adversary wins). Additionally simple forms of "point obfuscations" are generated for all q points τ_i by running each τ_i through one-way function owf to obtain

$$\overline{p}_{\tau_i} \leftarrow \mathsf{owf}(\tau_i).$$

It then punctures key k_1 on values $\tau_1, \ldots, \tau_q$ to receive punctured key k_1^* and hardcodes this into both circuits. To not change the functionality on these circuits the original PRF values are hardcoded, that is $\alpha_i^* \leftarrow \mathsf{F}_1.\mathsf{Eval}(\mathsf{k}_1, \tau_i)$ is hardcoded into both circuits C_0 and C_1. The branching and test operation whether an input is for a punctured value is done with the recomputation of the one-way function.

In addition, on the i-th call to oracle PROV prover P^* is run on randomness τ_i. Note that τ_i is the only randomness of the prover, and prover P^* then choses its commitment as α_i.

pPRF

$\mathsf{Game}_3(\lambda)$**:** The game is identical to the previous game except that now values α_i^* are chosen uniformly at random in $\{0,1\}^{\mathsf{F}_1.\mathsf{ol}(\lambda)}$.

bad_{α^*}

$\mathsf{Game}_4(\lambda)$**:** The game is identical to the game before except that we define that the adversary wins in case any of the values $\boldsymbol{\alpha}^*[i]$ is chosen to be in the image of the PRF for key k_1. Note that since F_1 is length doubling this happens only with probability $q \cdot 2^{-\lambda}$.

iO

$\mathsf{Game}_5(\lambda)$**:** As before but we again use the unpunctured key k_1. The hardcoded values for $\tau_1, \ldots, \tau_q$ (i.e., their images under the the one-way function), as well as the α_i^*, remain hardcoded.

diO+BCP14+OWF

$\mathsf{Game}_6(\lambda)$**:** We now change the if-branch in circuit C_1, such that it does not depend on hardcoded values $\overline{\mathbf{p}}_\tau$ anymore. Note that the if-branch was changed in the second step in order not to change the functionality of the circuits. This step changes the functionality by reverting the

iO

if-branch to check only whether input α^* is different from $\mathsf{F}_1.\mathsf{Eval}(\mathsf{k}_1, \tau)$ (i.e., the original check).

pPRF

$\mathsf{Game}_7(\lambda)$**:** The game is identical to the previous game but now key k_2^* is punctured on values $\alpha_1^*, \ldots, \alpha_q^*$. To not change the functionality on these circuits the original PRF values are hardcoded, that is $\mathbf{r}^*[i] \leftarrow \mathsf{F}_2.\mathsf{Eval}(\mathsf{k}_2, \alpha_i^*)$ is hardcoded into circuit C_1.

iO

$\mathsf{Game}_8(\lambda)$**:** The game is identical to the previous game but now hardcoded values r_i^* are sampled uniformly at random.

$\mathsf{Game}_9(\lambda)$**:** In this step we precompute the actual values α instead of having values $\mathbf{r}^*$ hardcoded. That is, we change circuit K^* to compute additionally $\boldsymbol{\alpha}[i] \leftarrow \mathsf{P}_0(\mathsf{crs}; \mathbf{r}^*[i])$ and hardcode these into C_1.

hvzk

$\mathsf{Game}_{10}(\lambda)$**:** Let $\mathcal{S} = (\mathcal{S}', \mathcal{S}'' = (\mathcal{S}_0'', \mathcal{S}_1''))$ denote the honest verifier zero knowledge simulator of the underlying 3PC protocol. The setting is identical to the previous game, but now we "switch to oracle SIMU". The crs is generated as before with three exceptions: (1) The simulator of the underlying protocol $\mathcal{S}'$ is run to generate a CRS and trapdoor crs and tk for the underlying protocol; (2) Values α are computed with simulator $\mathcal{S}_0''$ of the underlying protocol (with coins $\mathbf{r}^*[i]$); (3) As we are now in the simulated setting, we generate the trapdoor $\overline{\mathsf{tk}}$ to contain the randomness values $r_1^*, \ldots, r_q^*$ as well as pre-commitments $\alpha_1^*, \ldots, \alpha_q^*$.

On the i-th query $(\mathsf{crs}, \mathsf{tk}, x, w, b, i)$ oracle SIMU answers as follows: if $b = 0$, then it runs simulator $\mathcal{S}_0''$ of the underlying 3PC protocol on input its CRS and trapdoor and on randomness r_i to obtain α and β (which are identical as the precomputed ones) which are returned to the adversary. If, on the other hand, $b = 1$, it runs simulator $\mathcal{S}'$ of the underlying 3PC protocol on input its CRS and trapdoor, as well as instance x and with randomness r_i^*, to obtain a proof $\pi = (\alpha, \beta, \gamma)$. Note that by definition, as $\mathcal{S}$ has instance-independent commitments, $\mathcal{S}$ will generate a commitment and challenge as $(\alpha, \beta) \leftarrow \mathcal{S}_0''(\mathsf{crs}, \mathsf{tk}, r_i^*)$ (where crs and tk are the CRS and trapdoor from the underlying protocol). It replaces α for $\boldsymbol{\alpha}^*[i]$ and returns proof $\pi^* := (\boldsymbol{\alpha}^*[i], \beta, \gamma)$.

We give a complete description of simulator $\mathsf{Sim} = (\mathsf{Sim}', \mathsf{Sim}'' = (\mathsf{Sim}_0'', \mathsf{Sim}_1''))$ for the compiled protocol as pseudocode in Figure 5.

Analysis. First note that simulator Sim is indeed instance-independent. What remains to show is that the view of the adversary between Game_1 and the final game Game_{10} does only negligibly change. We discuss each step in turn below.

Game_1 to Game_2. By construction the generated circuits C_0 and C_1 are equivalent in both games, and hence a distinguisher for the two games yields a distinguisher against the indistinguishability obfuscator. Note that we are not dealing with a single circuit, but two circuits, and thus we have that

$$|\Pr[\mathsf{Game}_1(\lambda) = 1] - \Pr[\mathsf{Game}_2(\lambda) = 1]| \leq 2 \cdot \mathsf{Adv}^{\mathrm{io}}_{\mathsf{iO},\mathsf{Sam},\mathsf{D}}(\lambda)$$

where sampler Sam and distinguisher D are the adversary induced by games Game_1 and Game_2.

Game_2 to Game_3. By construction the only change is that the answers on "punctured points" are now chosen as uniformly random values. Thus, a distinguisher between the games induces a distinguisher against the puncturable pseudorandom function F_1.

$$|\Pr[\mathsf{Game}_2(\lambda) = 1] - \Pr[\mathsf{Game}_2(\lambda) = 1]| \leq \mathsf{Adv}^{\mathrm{pprf}}_{\mathsf{F}_1,\mathcal{A}_1,\mathcal{A}_2}(\lambda).$$

$\underline{\mathsf{Sim}'(1^\lambda)}$

$(\mathsf{crs}, \mathsf{tk}) \leftarrow\!\$\, \mathcal{S}'(1^\lambda)$
$\mathsf{k}_1 \leftarrow\!\$\, \mathsf{F}_1.\mathsf{KGen}(1^\lambda)$
$\mathsf{k}_2 \leftarrow\!\$\, \mathsf{F}_2.\mathsf{KGen}(1^\lambda)$
for $i = 1, \ldots, q$ **do**
  $\boldsymbol{\tau}[i] \leftarrow\!\$\, \{0,1\}^{\mathsf{F.il}(\lambda)}$
  $\overline{\mathbf{p}}_\tau[i] \leftarrow \mathsf{owf}(\boldsymbol{\tau}[i])$
  $\boldsymbol{\alpha}^*[i] \leftarrow\!\$\, \{0,1\}^{\mathsf{F.ol}(\lambda)}$
  $\mathbf{r}^*[i] \leftarrow\!\$\, \{0,1\}^{\mathsf{F.ol}(\lambda)}$
  $(\boldsymbol{\alpha}[i], \beta_{\mathsf{tmp}}) \leftarrow\!\$\, \mathcal{S}''_0(\mathsf{crs}, \mathsf{tk}; \mathbf{r}^*[i])$
$\mathsf{k}_2^* \leftarrow\!\$\, \mathsf{F}_2.\mathsf{Pntr}(\mathsf{k}_2, \{\boldsymbol{\alpha}^*[1], \ldots, \boldsymbol{\alpha}^*[q]\})$
$\overline{C}_0 \leftarrow\!\$\, \mathsf{iO}(C_0[\mathsf{k}_1, \overline{\mathbf{p}}, \boldsymbol{\alpha}^*])$
$\overline{C}_1 \leftarrow\!\$\, \mathsf{iO}(C_1[\mathsf{k}_1, \boldsymbol{\alpha}^*, \mathsf{k}_2^*, \boldsymbol{\alpha}])$
$\overline{\mathsf{crs}} \leftarrow (\mathsf{crs}, \overline{C}_0, \overline{C}_1)$
$\overline{\mathsf{tk}} \leftarrow (\mathsf{tk}, \boldsymbol{\alpha}^*, \mathbf{r}^*)$
return $(\overline{\mathsf{crs}}, \overline{\mathsf{tk}})$

$\underline{\mathsf{Sim}''_0(1^\lambda, \overline{\mathsf{crs}}, \overline{\mathsf{tk}})}$

$(\mathsf{crs}, \overline{C}_0, \overline{C}_1) \leftarrow \overline{\mathsf{crs}}$
$(\mathsf{tk}, \boldsymbol{\alpha}^*, \mathbf{r}^*) \leftarrow \overline{\mathsf{tk}}$

……on i-th query……

$(\alpha, \beta) \leftarrow \mathcal{S}''_0(\mathsf{crs}, \mathsf{tk}; \mathbf{r}^*[i])$
return $(\boldsymbol{\alpha}[i], \beta)$

$\underline{\mathsf{Sim}''_1(1^\lambda, \overline{\mathsf{crs}}, \overline{\mathsf{tk}}, x)}$

$(\mathsf{crs}, \overline{C}_0, \overline{C}_1) \leftarrow \overline{\mathsf{crs}}$
$(\mathsf{tk}, \boldsymbol{\alpha}^*, \mathbf{r}^*) \leftarrow \overline{\mathsf{tk}}$

…..on i-th query…..

$\gamma \leftarrow \mathcal{S}''_1(\mathsf{crs}, \mathsf{tk}, x; \mathbf{r}^*[i])$
return γ

$\underline{C_0[\mathsf{k}_1, \overline{\mathbf{p}}, \boldsymbol{\alpha}^*](\tau)}$

if $\exists i \in [q] : \overline{\mathbf{p}}[i] = \mathsf{owf}(\tau)$ **then**
  return $\boldsymbol{\alpha}^*[i]$
$\alpha^* \leftarrow \mathsf{F}_1.\mathsf{Eval}(\mathsf{k}_1, \tau)$
return α^*

$\underline{C_1[\mathsf{k}_1, \overline{\mathbf{p}}, \boldsymbol{\alpha}^*, \mathsf{k}_2^*, \boldsymbol{\alpha}](\alpha^*, \tau)}$

if $\exists i \in [q] : \alpha^* = \boldsymbol{\alpha}^*[i]$
  $\alpha \leftarrow \boldsymbol{\alpha}[i]$
else
  $r^* \leftarrow \mathsf{F}_2.\mathsf{Eval}(\mathsf{k}_2^*, \alpha^*)$
  $\alpha \leftarrow \mathsf{P}_0(r^*)$
if $\alpha^* \neq \mathsf{F}_1.\mathsf{Eval}(\mathsf{k}_1, \tau)$ **then**
  $r^* \leftarrow \perp$
return (α, r^*)

Figure 5: The complete pseudocode for q-bounded HVZK simulator Sim for the compiled protocol. Note that simulator Sim''_1 runs the underlying simulator on the same random coins as simulator Sim''_0 thus "generating" the same values (α, β).

Game_3 to Game_4. By the fundamental lemma of the game playing technique [BR06], games Game_3 and Game_4 are identical unless event bad_{α^*} occurs which we can upper bound by $q \cdot 2^{-\mathsf{F.il}(\lambda)}$ as the PRF by definition has stretch 2.

$$|\Pr[\mathsf{Game}_2(\lambda) = 1] - \Pr[\mathsf{Game}_2(\lambda) = 1]| \leq q \cdot 2^{-\mathsf{F.il}(\lambda)}.$$

Game_4 to Game_5. Noting that from Game_4 to Game_5 the circuits only change syntactically, but not functionally (as the unpunctured key k_1 is used instead of k_1^*), allows us to perform an analysis analogously to the first game hop. Hence,

$$|\Pr[\mathsf{Game}_4(\lambda) = 1] - \Pr[\mathsf{Game}_5(\lambda) = 1]| \leq 2 \cdot \mathsf{Adv}^{\mathrm{io}}_{\mathsf{iO},\mathsf{Sam},\mathsf{D}}(\lambda).$$

Game_5 to Game_6. We will reduce the distinguishing advantage of any distinguisher between the two games Game_5 and Game_6 to the security of the indistinguishability obfuscator iO and the security of the injective one-way function owf. For this we rely on the result of Boyle *et al.* [BCP14] who relate indistinguishability obfuscation and restricted differing-inputs obfuscation. We recall their result as Theorem 3.11 on page 15. For this we consider a circuit sampler Sam that runs the steps of Game_5 up to and including line 12. It outputs as auxiliary information circuit $\overline{C}_0$ and string crs, that is, it sets $\mathsf{aux} \leftarrow (\overline{C}_0, \mathsf{crs})$. As circuits, it constructs the circuit C_1 from line 13 once as in Game_5 and once as in Game_6.

Additionally we construct a diO distinguisher that gets as input an obfuscation $\overline{C}_1$ of either of the two circuits and the auxiliary information $\mathsf{aux} \leftarrow (\overline{C}_0, \mathsf{crs})$. It sets $\overline{\mathsf{crs}} \leftarrow (\mathsf{crs}, \overline{C}_0, \overline{C}_1)$, and then runs the game distinguisher on input $\overline{\mathsf{crs}}$ and outputs whatever it outputs.

If $\overline{C}_1$ is as in Game_5 then together sampler and distinguisher perfectly simulate game Game_5, and otherwise they perfectly simulate Game_6. We want to argue that

$$|\Pr[\mathsf{Game}_6(\lambda) = 1] - \Pr[\mathsf{Game}_7(\lambda) = 1]| \leq \mathsf{Adv}^{\mathrm{io}}_{\mathsf{iO},\mathsf{Sam},\mathsf{D}}(\lambda) + \mathsf{negl}(\lambda) + q \cdot \mathsf{Adv}^{\mathrm{owf}}_{\mathsf{owf},\mathcal{A}}(\lambda),$$

where the negligible term denotes the loss from using the indistinguishability obfuscator in a diO setting [BCP14] (cf. Theorem 3.11), and $\mathsf{Adv}^{\mathrm{owf}}_{\mathsf{owf},\mathcal{A}}(\lambda)$ is a factor due to the use of injective one-way function owf. For this we need to show that sampler Sam is differing-inputs, and the circuits differ only on polynomially many points. First note that the two circuits, as prepared by sampler Sam, differ only on inputs $\boldsymbol{\tau}[i]$ for $i \in [q]$. We can further reduce the advantage of any extractor in the Diff game to the inversion advantage $\mathsf{Adv}^{\mathrm{owf}}_{\mathsf{owf},\mathcal{A}}(\lambda)$ of an adversary $\mathcal{A}$ against one-way function owf. To see this, consider that all values in $\boldsymbol{\tau}$ are sampled uniformly at random and no extra information about these values is available in aux. Thus, an adversary $\mathcal{A}$ against the one-wayness of owf, that gets as input a random image y, can simply choose $q-1$ additional values $\mathsf{owf}(\tau_i)$ and construct auxiliary information and circuits as does sampler Sam, but using y as one of the values in $\boldsymbol{\tau}$. An extractor that is successful in finding a differing input will with probability $1/q$ have inverted y which concludes the argument.

Game_6 to Game_7. Again, the circuits are only changed syntactically allowing for an analogous analysis as in the first game hop. Thus,

$$|\Pr[\mathsf{Game}_6(\lambda)=1] - \Pr[\mathsf{Game}_7(\lambda)=1]| \leq 2 \cdot \mathsf{Adv}^{\mathrm{io}}_{\mathsf{iO},\mathsf{Sam},\mathsf{D}}(\lambda).$$

Game_7 to Game_8. Similarly to the second game hop the hardcoded values on punctured inputs are now chosen uniformly at random, allowing for an analysis analogous to the second game hop. We obtain,

$$|\Pr[\mathsf{Game}_7(\lambda)=1] - \Pr[\mathsf{Game}_8(\lambda)=1]| \leq \mathsf{Adv}^{\mathrm{pprf}}_{\mathsf{F}_2,\mathcal{A}_1,\mathcal{A}_2}(\lambda).$$

Game_8 to Game_9. Pre-computing values α for the punctured target points does not change the functionality of circuit C_1, and an analysis analogous to the first game hop allows us to reduce the distinguishing probability to the security of the indistinguishability obfuscator. Note that here only C_1 is changed, and thus:

$$|\Pr[\mathsf{Game}_8(\lambda)=1] - \Pr[\mathsf{Game}_9(\lambda)=1]| \leq \mathsf{Adv}^{\mathrm{io}}_{\mathsf{iO},\mathsf{Sam},\mathsf{D}}(\lambda).$$

Game_9 to Game_{10}. On the i-th query simulator Sim runs the simulator $\mathcal{S}$ of the underlying 3PC protocol on random coins $\mathbf{r}^*[i]$ to obtain (α,β,γ). As by assumption $\mathcal{S}$ is instance-independent, it will generate (α,β,γ) such that $(\alpha,\beta) = \mathcal{S}''_0(\mathsf{crs},\mathsf{tk};\mathbf{r}^*[i])$ where crs and tk are the CRS and trapdoor of the underlying simulator. Simulator Sim returns as protocol transcript $(\boldsymbol{\alpha}^*[i],\beta,\gamma)$. By construction C_1 will map $\boldsymbol{\alpha}^*[i]$ to α, thus presenting the adversary with an identical simulation as in the previous step except that (α,β,γ) are now generated by the HVZK simulator $\mathcal{S}$. Thus, a distinguisher between the two games can be used to construct a distinguisher against the HVZK simulator.

To construct an attacker against HVZK note that in line 8 of games Game_9 and Game_{10} the randomness $\mathbf{r}^*$ is not encoded in any of the circuits any longer, but it is only used for running P_0 and $\mathcal{S}''_0$, respectively. Thus we construct an adversary $\mathcal{A}$ that gets as input the crs and then runs game Game_9 up to but not including line 8. (The for loop is executed for the lines before line 8.) It then calls its oracle on all indexes $i = 1,\ldots,q$ and with bit $b = 0$ (and x set to, for example, $\perp$, see also Definition 4.8). The oracle returns $(\alpha_i,\beta_i)_{i=1,\ldots,q}$ which allows adversary $\mathcal{A}$ to construct vector $\boldsymbol{\alpha}$ as is done in line 9. It then runs the remaining steps of the setup to obtain $(\mathsf{crs},\overline{C}_0,\overline{C}_1)$ which it passes on to the distinguisher. Queries of the distinguisher are passed on to its own oracle and it outputs whatever the distinguisher outputs.

If the oracle implements PROV then the adversary perfectly simulates game Game_9 and otherwise it perfectly simulates Game_{10}. Thus, if the underlying protocol achieves (q, ϵ)-bounded instance-independent HVZK then

$$|\Pr[\mathsf{Game}_8(\lambda) = 1] - \Pr[\mathsf{Game}_9(\lambda) = 1]| \overset{\text{asym}}{\leq} \epsilon.$$

Note that for the general case with $\ell > 1$ one needs to simulate $q \cdot \ell$ oracle queries, and thus the reduction looses a factor of $\ell\epsilon$.

6 Obtaining Instance Independence

In this section, we present a compiler that turns a 3PC protocol with HVZK and instance-independent commitments into a 3PC protocol in the CRS model that has instance-independent commitments *and* instance-independent simulators, that is, the HVZK simulator produces α and β independently of the instance.

The idea is inspired by Lindell's compiler [Lin15]. Namely, we replace α by a commitment α^* to α where the deployed commitment scheme can come in one of two modes: if honestly generated the commitment will be *perfectly binding* thus allowing us to directly argue that the resulting compiled protocol retains soundness and completeness. On the other hand, the commitment scheme can be initialized to be *equivocal* (looking indistinguishably from the honest commitment setup), in such a way that a simulator can open a commitment to arbitrary values. This way, the simulator can first commit to an arbitrary α^* and then, using the trapdoor in the CRS, it can open α^* to some arbitrary value α. In particular, in the reduction to the HVZK property, the verifier can choose α^* before knowing the statement that the simulator of the underlying protocol needs in order to produce α.

Such *dual-mode* commitment schemes were studied and constructed by Catalano and Visconti [CV05, CV07] who called them *hybrid commitments*. We here give the definition due to Lindell [Lin15]. We write $\mathsf{C.Com}(\mathsf{pp}, m)$ taking public parameters pp and message m to obtain a commitment c and opening δ, and write $\mathsf{C.Vf}(\mathsf{pp}, m, c, \delta)$ to denote verification of a commitment. We also let $\mathsf{C.il} : \mathbb{N} \to \mathbb{N}$ (resp., $\mathsf{C.ol} : \mathbb{N} \to \mathbb{N}$) denote the input (resp., output) length functions corresponding to C.

Definition 6.1 (Dual-Mode Commitment) *A* dual-mode commitment scheme *is a tuple of* PPT *algorithms* ($\mathsf{C.GenPP}$, $\mathsf{C.Com}, \mathsf{C.Vf}, \mathsf{C.il}, \mathsf{C.ol}$) *with* PPT *commitment simulator* ($\mathcal{S}_{\mathsf{com}}.\mathsf{GenPP}$, $\mathcal{S}_{\mathsf{com}}.\mathsf{Com}$, $\mathcal{S}_{\mathsf{com}}.\mathsf{Open}$) *such that the following holds.*

PUBLIC PARAMETERS. *On input the security parameter* 1^λ *algorithm* GenPP *outputs public parameters* pp.

PERFECT COMPLETENESS. *For all security parameters* $\lambda \in \mathbb{N}$ *and all messages* $m \in \{0,1\}^{\mathsf{C.il}(\lambda)}$ *it holds that*

$$\Pr\Big[\mathsf{C.Vf}(\mathsf{pp}, c, m, \delta) = 1 : \mathsf{pp} \leftarrow\!\!\$\, \mathsf{GenPP}(1^\lambda), (c, \delta) \leftarrow\!\!\$\, \mathsf{C.Com}(\mathsf{pp}, m)\Big] = 1.$$

PERFECTLY BINDING. *For all security parameters* $\lambda \in \mathbb{N}$ *and all public parameters* $\mathsf{pp} \leftarrow\!\!\$\, \mathsf{GenPP}(1^\lambda)$ *algorithm* $\mathsf{C.Com}$ *is a perfectly-binding non-interactive commitment scheme, that is, for all security parameters* $\lambda \in \mathbb{N}$*, all messages* $m, m' \in \{0,1\}^{\mathsf{C.il}(\lambda)}$ *such that* $m \neq m'$*, and all openings* $\delta' \in \{0,1\}^*$ *we have that*

$$\Pr\Big[\mathsf{C.Vf}(\mathsf{pp}, c, m', \delta') = 0 : \mathsf{pp} \leftarrow\!\!\$\, \mathsf{GenPP}(1^\lambda), (c, \delta) \leftarrow\!\!\$\, \mathsf{C.Com}(\mathsf{pp}, m)\Big] = 1$$

EQUIVOCAL. *For every* PPT *adversary* $\mathcal{A}$*, advantage* $\mathsf{Adv}^{\mathrm{com}}_{\mathsf{C},\mathcal{S}_{\mathsf{com}},\mathcal{A}}(\lambda)$ *defined as*

$$\mathsf{Adv}^{\mathrm{com}}_{\mathsf{C},\mathcal{S}_{\mathsf{com}},\mathcal{A}}(\lambda) := 2 \cdot \Pr\left[\mathsf{COM}^{\mathcal{A}}_{\mathsf{C},\mathcal{S}_{\mathsf{com}}}(\lambda)\right] - 1$$

should be negligible, where game COM *is defined as*

$\mathsf{COM}^{\mathcal{A}}_{\mathsf{C},\mathcal{S}_{\mathsf{com}}}(\lambda)$	$\textsc{Com}(m)$
$b \leftarrow\!\!\$\ \{0,1\}$	**if** $b = 0$ **then**
$\mathsf{pp}_0 \leftarrow\!\!\$\ \mathsf{C.GenPP}(1^\lambda)$	$(c,\delta) \leftarrow\!\!\$\ \mathsf{C.Com}(\mathsf{pp}_0, m)$
$(\mathsf{pp}_1, \mathsf{tk}_{\mathsf{com}}) \leftarrow\!\!\$\ \mathcal{S}_{\mathsf{com}}.\mathsf{GenPP}(1^\lambda)$	**else**
$b' \leftarrow\!\!\$\ \mathcal{A}^{\textsc{Com}}(\mathsf{pp}_b)$	$c \leftarrow\!\!\$\ \mathcal{S}_{\mathsf{com}}.\mathsf{Com}(\mathsf{pp}_1, \mathsf{tk}_{\mathsf{com}})$
return $b = b'$	$\delta \leftarrow\!\!\$\ \mathcal{S}_{\mathsf{com}}.\mathsf{Open}(\mathsf{pp}_1, \mathsf{tk}_{\mathsf{com}}, c, m)$
	return (c,δ)

Additionally to dual-mode commitments we require that the underlying 3PC protocol satisfies so-called *special* HVZK that essentially allows the simulator to simulate arguments for a given challenge $\beta \in \mathcal{B}$.

Definition 6.2 (Special HVZK) *Let* $\Pi = ((\mathsf{P}_0, \mathsf{P}_1), \mathsf{V})$ *be a 3PC argument system for a polynomial-time computable relation* R*. We say that* Π *satisfies* special ϵ*-HVZK if there exists a* PPT *machine* $\mathcal{S}$ *such that for all* $x \in L_R$ *and for all* $\beta \in \mathcal{B}$ *the following two distributions are* ϵ*-computationally indistinguishable*

$$\{(\alpha,\beta,\gamma) :\ \alpha \leftarrow\!\!\$\ \mathsf{P}_0(x,w); \gamma \leftarrow\!\!\$\ \mathsf{P}_1(x,w,\beta)\} \quad \textit{and} \quad \{(\alpha,\beta,\gamma) :\ (\alpha,\gamma) \leftarrow\!\!\$\ \mathcal{S}(x,\beta)\}\,.$$

In other words, for any $\beta \in \mathcal{B}$*, the simulator outputs an argument* (α,β,γ) *with this challenge, which is indistinguishable from a real argument with challenge* β*.*

As observed by Fischlin [Fis05] common protocols obey this special zero-knowledge notion and if, furthermore, the challenge size is logarithmic in the security parameter then assuming special HVZK is without loss of generality.

6.1 The Compiler

Given a 3PC protocol $\Pi = (\mathsf{P}, \mathsf{V})$ with HVZK simulator $\mathcal{S}$ which satisfies the conditions above, we construct a compiled protocol $\Pi^* = (\mathsf{K}^*, \mathsf{P}^*, \mathsf{V}^*)$ as follows. Let $\mathsf{C} = (\mathsf{C.GenPP}, \mathsf{C.Com}, \mathsf{C.Open}, \mathsf{C.il}, \mathsf{C.ol})$ be a dual-mode commitment scheme, where C.GenPP is the honest public parameter generator for the perfectly binding variant of the commitment scheme. On input the security parameter setup algorithm K^* simply runs the public parameter generation of the commitment scheme.

$\mathsf{K}^*(1^\lambda)$

$\mathsf{pp} \leftarrow\!\!\$\ \mathsf{C.GenPP}(1^\lambda)$
$\mathsf{crs} \leftarrow \mathsf{pp}$
return crs

We present the compiled protocol in Figure 6. What remains to show is that the compiled protocol retains instance-independent commitments, completeness, soundness and that it achieves instance-independent HVZK simulators.

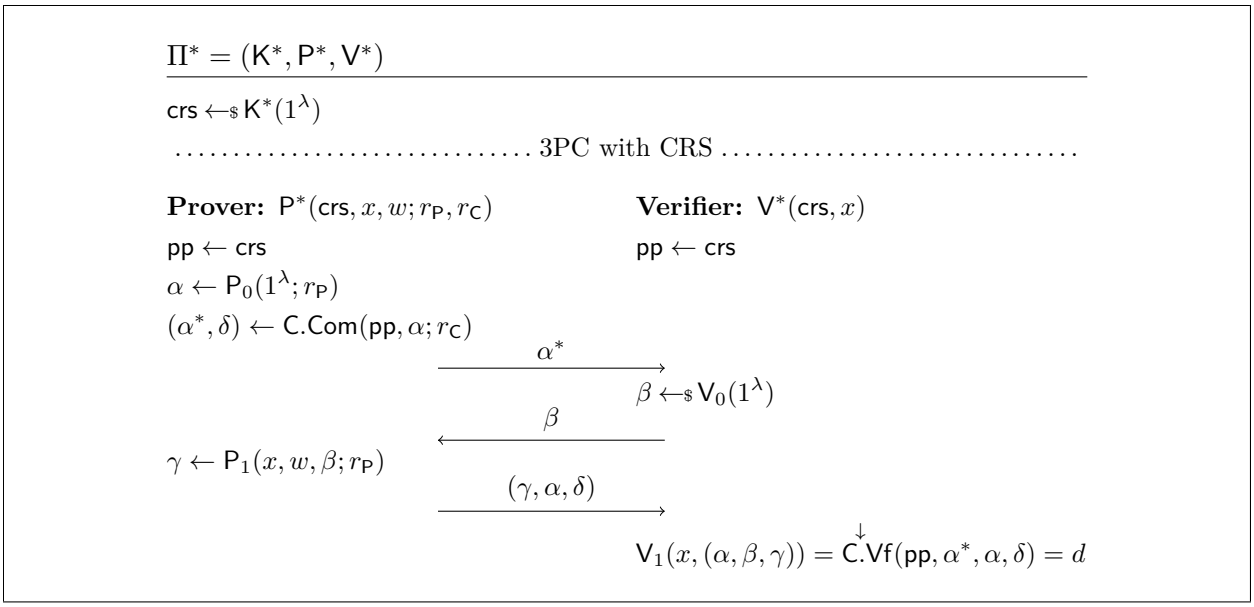


Figure 6: The compiled protocol achieving instance-independent HVZK simulators.

6.2 Security Analysis

Theorem 6.3 *Let $\Pi = (\mathsf{K}, \mathsf{P}, \mathsf{V})$ be a 3PC argument system for a polynomial-time computable relation R such that Π is c-complete and s-sound and has instance-independent commitments and satisfies special HVZK. Let $\mathsf{C} = (\mathsf{C.GenPP}, \mathsf{C.Com}, \mathsf{C.Open}, \mathsf{C.il}, \mathsf{C.ol})$ be a dual-mode commitment scheme. Then, in the CRS model, the compiled protocol $\Pi^* = (\mathsf{K}^*, \mathsf{P}^*, \mathsf{V}^*)$ is c-complete, s-sound and satisfies unbounded instance-independent HVZK. Furthermore the compiled protocol has instance-independent commitments.*

We discuss the various properties in turn.

Instance-independent commitments. As instead of the original commitment α now we use α^* which is a commitment to α, the compiled protocol retains instance-independent prover commitments.

Completeness. If the public parameters are generated honestly for the commitment scheme, the scheme is perfectly binding. Hence, if on an honest execution the verifier does not accept it would not accept for the same execution on the underlying protocol. Hence, if the underlying protocol is c-complete then so is the compiled protocol.

Soundness. Again soundness directly follows from the underlying protocol and the fact that the commitment scheme is perfectly binding if the public parameters are honestly created. Thus, if the underlying 3PC protocol has s-soundness then so does the compiled protocol.

Zero knowledge. We show that if the underlying protocol satisfies special-HVZK, then the compiled protocol satisfies q-bounded instance-independent HVZK (see Definition 4.8). Note that, in fact we can prove an unbounded variant, but for our purpose the bounded variant suffices as our FS-collapse only supports bounded zero knowledge. Further note that we require the underlying protocol to satisfy special-HVZK but claim that the compiled protocol does not achieve *special* any

longer. We note that the compiled protocol, in fact, achieves a *special*-instance-independent HVZK variant but again, as we do not need this for our result, we chose not to additionally formalize it.

The proof will be down to the security of the commitment scheme and the special-HVZK of the underlying protocol, and consists of two game hops.

$\mathsf{Game}_1(\lambda)$**:** The first game is equivalent to real world setting $\mathsf{rIPS}_{\mathcal{A}}^{\Pi^*}$ where the adversary has access to the PROV oracle which runs honest executions of the protocol to obtain the transcripts for the adversary.

C

$\mathsf{Game}_2(\lambda)$**:** The setting is similar to the previous game with the exception that now the commitment is returning fake commitments. In order for this to work we switch to the SIMU oracle and let the trapdoor contain the commitment trapdoor. That is, the setup algorithm is changed to run the simulated commitment setup and then the first message of the prover is replaced by a simulated commitment. To make the view consistent on calling SIMU with $b = 1$ the simulated commitment will be opened to the correct α as obtained by the underlying protocol. For this note that at this point the adversary already supplied a valid instance and witness. That is the protocol transcript is still computed with the underlying prover and simulator. The current setting is best visualized with an adapted protocol:

$\Pi^* = (\mathsf{K}^*, \mathsf{P}^*, \mathsf{V}^*)$

$(\mathsf{pp}, \mathsf{tk}_{\mathsf{com}}) \leftarrow\$ \mathcal{S}_{\mathsf{com}}.\mathsf{GenPP}(1^\lambda)$
$\mathsf{crs} \leftarrow \mathsf{pp}$

3PC with CRS

Prover: $\mathsf{P}^*(\mathsf{crs}, x, w)$		**Verifier:** $\mathsf{V}^*(\mathsf{crs}, x)$
$\mathsf{pp} \leftarrow \mathsf{crs}$		$\mathsf{pp} \leftarrow \mathsf{crs}$
$\alpha \leftarrow\$ \mathsf{P}_0(1^\lambda)$		
$\alpha^* \leftarrow\$ \mathcal{S}_{\mathsf{com}}.\mathsf{Com}(\mathsf{pp}, \mathsf{tk}_{\mathsf{com}})$	$\xrightarrow{\alpha^*}$	
	$\xleftarrow{\beta}$	$\beta \leftarrow\$ \mathsf{V}_0(1^\lambda)$
$\gamma \leftarrow\$ \mathsf{P}_1(x, w, \beta)$		
$\delta \leftarrow\$ \mathcal{S}_{\mathsf{com}}.\mathsf{Open}(\mathsf{pp}, \mathsf{tk}_{\mathsf{com}}, \alpha^*, \alpha)$	$\xrightarrow{(\gamma, \alpha, \delta)}$	
		$\mathsf{V}_1(x, (\alpha, \beta, \gamma)) = \underset{\downarrow}{\mathsf{C}}.\mathsf{Vf}(\mathsf{pp}, \alpha^*, \alpha, \delta) = 1$

hvzk

$\mathsf{Game}_3(\lambda)$**:** The setting now switches to use the HVZK simulator $\mathcal{S}$ of the underlying protocol. Calls to SIMU with bit $b = 0$ are answered as before with a simulated commitment. Note that while in the last setting β was chosen by the verifier V_0 we now choose a random value $\beta \in \mathcal{B}$. (Technically this is not a change, since we consider 3PC protocols.) If later for the same index a call to SIMU with bit $b = 1$ is made, then the special HVZK simulator is run on challenge β (that was chosen before) and instance x to obtain $(\alpha, \gamma) \leftarrow\$ \mathcal{S}(x, \beta)$. Then, as before the commitment is opened to α and $(\alpha^*, \beta, (\gamma, \alpha, \delta))$ is returned to the adversary. Before we present the pseudocode of the simulator, we again present a view of the *adapted protocol* to visualize the setting:

$\Pi^* = (\mathsf{K}^*, \mathsf{P}^*, \mathsf{V}^*)$

$(\mathsf{pp}, \mathsf{tk}_{\mathsf{com}}) \leftarrow\!\$\ \mathcal{S}_{\mathsf{com}}.\mathsf{GenPP}(1^\lambda)$
$\mathsf{crs} \leftarrow \mathsf{pp}$

3PC with CRS

Prover: $\mathsf{P}^*(\mathsf{crs}, x, w)$		**Verifier:** $\mathsf{V}^*(\mathsf{crs}, x)$
$\mathsf{pp} \leftarrow \mathsf{crs}$		$\mathsf{pp} \leftarrow \mathsf{crs}$
$\alpha^* \leftarrow\!\$\ \mathcal{S}_{\mathsf{com}}.\mathsf{Com}(\mathsf{pp}, \mathsf{tk}_{\mathsf{com}})$	$\xrightarrow{\alpha^*}$	
		$\beta \leftarrow\!\$\ \mathcal{B}$
	$\xleftarrow{\beta}$	
$(\alpha, \gamma) \leftarrow\!\$\ \mathcal{S}(x, \beta)$		
$\delta \leftarrow\!\$\ \mathcal{S}_{\mathsf{com}}.\mathsf{Open}(\mathsf{pp}, \mathsf{tk}_{\mathsf{com}}, \alpha^*, \alpha)$	$\xrightarrow{(\gamma, \alpha, \delta)}$	
		$\mathsf{V}_1(x, (\alpha, \beta, \gamma)) = \overset{\downarrow}{\mathsf{C}}.\mathsf{Vf}(\mathsf{pp}, \alpha^*, \alpha, \delta) = 1$

We next present the pseudocode of the simulator $\mathsf{Sim} = (\mathsf{Sim}', (\mathsf{Sim}''_0, \mathsf{Sim}''_1))$. Note that it is assumed that Sim''_0 and Sim''_1 are always run on matching random coins, that is, Sim''_1 can recover (α^*, β) as output by Sim''_0 (see Definition 4.8).

$\mathsf{Sim}'(1^\lambda)$

$(\mathsf{pp}, \mathsf{tk}_{\mathsf{com}}) \leftarrow\!\$\ \mathcal{S}_{\mathsf{com}}.\mathsf{GenPP}(1^\lambda)$
$\mathsf{tk} \leftarrow \mathsf{tk}_{\mathsf{com}}$
$\mathsf{crs} \leftarrow \mathsf{pp}$
return $(\mathsf{crs}, \mathsf{tk})$

$\mathsf{Sim}''_0(\mathsf{crs}, \mathsf{tk})$

$\mathsf{tk}_{\mathsf{com}} \leftarrow \mathsf{tk}$
$\mathsf{pp} \leftarrow \mathsf{crs}$
$\alpha^* \leftarrow\!\$\ \mathcal{S}_{\mathsf{com}}.\mathsf{Com}(\mathsf{pp}, \mathsf{tk}_{\mathsf{com}})$
$\beta \leftarrow\!\$\ \mathcal{B}$
return (α^*, β)

$\mathsf{Sim}''_1(\mathsf{crs}, \mathsf{tk}, x)$

$\mathsf{tk}_{\mathsf{com}} \leftarrow \mathsf{tk}$
$\mathsf{pp} \leftarrow \mathsf{crs}$
$(\alpha^*, \beta) \leftarrow\!\$\ \mathsf{Sim}''_0(\mathsf{crs}, \mathsf{tk})$
$(\alpha, \gamma) \leftarrow\!\$\ \mathcal{S}(x, \beta)$
$\delta \leftarrow\!\$\ \mathcal{S}_{\mathsf{com}}.\mathsf{Open}(\mathsf{pp}, \mathsf{tk}_{\mathsf{com}}, \alpha^*, \alpha)$
return $(\alpha^*, \beta, (\gamma, \alpha, \delta))$

What is left to show is that the two game hops are negligibly close. We prove this below.

Game_1 to Game_2. The difference between the settings in Game_1 and Game_2 matches exactly the commitment game. That is, a distinguisher can be turned into an adversary against the commitment scheme. The commitment scheme adversary takes as input the public parameters pp which it forwards to the distinguisher between games Game_1 and Game_2. On an oracle query for index i with bit b the adversary distinguishes between b being set to 0 or to 1 and answers in these cases as follows:

$b = 0$ It runs the underlying prover P_0 on input the security parameter to obtain message α. It then sends α to its commitment oracle to receive commitment α^* and opening δ. Finally, it runs verifier V_0 on input the security parameter to receive β and returns (α^*, β) to the distinguisher.

$b = 1$ It recovers α, α^* and β (or if the corresponding call with $b = 0$ was not yet made, it performs the above steps). It then calls prover P_1 on input x, w and β to receive γ. (Note that P_1 is run on the same random coins as prover P_0 before, and is thus able to recover α.) Finally, it returns to the distinguisher $(\alpha^*, \beta, (\gamma, \alpha, \delta))$.

It is easy to see that if the commitment oracle returns simulated commitments then the adversary perfectly simulates Game_2 and otherwise Game_1. Thus, we have:

$$|\Pr[\mathsf{Game}_1(\lambda) = 1] - \Pr[\mathsf{Game}_2(\lambda) = 1]| \leq \mathsf{Adv}^{\mathrm{com}}_{\mathsf{C}, \mathcal{S}_{\mathsf{com}}, \mathcal{A}}(\lambda).$$

Game_2 **to** Game_3**.** The difference between the two games matches exactly the setup of the special-HVZK setting. That is, a distinguisher between the two games can be turned into an adversary $\mathcal{A}$ against the special-HVZK property of the underlying protocol. For this, adversary $\mathcal{A}$ runs the simulated setup of the commitment scheme to obtain $(\mathsf{pp}, \mathsf{tk}_{\mathsf{com}}) \leftarrow\!\!\$\, \mathcal{S}_{\mathsf{com}}.\mathsf{GenPP}(1^\lambda)$ and forwards pp as CRS to the distinguisher. On an oracle query for index i with bit b the adversary $\mathcal{A}$ distinguishes between b being set to 0 or to 1 and answers these cases as follows:

$b = 0$ It runs commitment simulator $\mathcal{S}_{\mathsf{com}}.\mathsf{Com}$ to obtain a simulated commitment α^*. It then picks a random value $\beta \in \mathcal{B}$ and returns (α^*, β).

$b = 1$ It recovers (α^*, β) (or freshly generates them if no corresponding oracle call with bit $b = 0$ has been made). It then calls its own oracle on x, w and β to receive a transcript (α, β, γ). (Note that Definition 6.2 is not stated in an oracle setting, but it can be restated as an adversary that gets oracle access to either the left distribution or the right distribution that it can call on inputs x, w and β for values x in the language and where w is a valid witness to this effect.) Adversary $\mathcal{A}$ then uses the commitment simulator to open the commitment α^* to α and it returns $(\alpha^*, \beta, (\gamma, \alpha, \delta))$.

If the adversary receives honest protocol transcripts then it perfectly simulates Game_2 and otherwise it perfectly simulates Game_3. Thus, the distinguishing probability between the two settings can be upper bounded by:

$$|\Pr[\mathsf{Game}_2(\lambda) = 1] - \Pr[\mathsf{Game}_3(\lambda) = 1]| \overset{\text{asym}}{\leq} \epsilon$$

assuming that the underlying 3PC protocol is special ϵ-HVZK. This concludes the proof.

7 Fiat–Shamir Signatures

We explain how to extend our techniques in order to obtain a standard model instantiation of FS signatures, under similar complexity assumptions as in the case of FS NIZK. In particular we will identify a certain class of so-called highly sound identification (ID) schemes, such that we can instantiate the hash function in the corresponding FS collapse via a q-wise independent hash function. On the positive side, we remark that the obtained signature scheme is in the standard model (i.e., without a CRS) even if the starting identification scheme is in the CRS model; the reason is that in the case of signatures we can always include the CRS as part of the public key of the verifier. On the negative side, the obtained signature scheme satisfies only a weak unforgeability flavour.

We recall the definition of ID and signature schemes in Section 7.1. Section 7.2 contains our positive result for FS signatures in the standard model. Finally, in Section 7.3, we explain how to obtain the desired properties by relying on similar tools as the ones used in the compilers from Section 5 and Section 6.

7.1 Identification and Signature Schemes

The FS transform can be used in order to generically obtain signature schemes starting from ID schemes satisfying certain properties.

Canonical ID schemes. An ID scheme consists of three PPT algorithms $(\mathsf{K}, \mathsf{P}, \mathsf{V})$. Algorithm K takes as input 1^λ and outputs a CRS $\mathsf{crs} \in \{0,1\}^*$, together with a pair of keys $(\mathsf{pk}, \mathsf{sk})$, where pk is called the public key and sk is called the secret key. (In case the ID scheme is in the standard model, then we simply set $\mathsf{crs} = \varepsilon$.) Later the prover P interacts with the verifier V to convince him he knows the secret key sk corresponding to pk (where both P and V are also given crs). At the end of the protocol execution, the verifier outputs a bit (representing his decision); we write $\langle \mathsf{P}(\mathsf{sk}), \mathsf{V} \rangle(\mathsf{crs}, \mathsf{pk})$ for the random variable corresponding to the verifier's verdict. Similarly, we write $\mathsf{P}(\mathsf{crs}, \mathsf{pk}, \mathsf{sk}) \rightleftarrows \mathsf{V}(\mathsf{crs}, \mathsf{pk})$ for the random variable corresponding to transcripts of honest protocol executions.

For applying the FS transform one is typically interested in so-called *canonical* ID schemes. Intuitively, canonical ID schemes are the counterpart of 3PC arguments (cf. Section 4.1). In particular, we can think of the prover algorithm as being split into two sub-algorithms $\mathsf{P} := (\mathsf{P}_0, \mathsf{P}_1)$, where P_0 takes as input a pair $(\mathsf{pk}, \mathsf{sk})$ and outputs the prover's first message α (the so-called commitment), and P_1 takes as input $(\mathsf{pk}, \mathsf{sk})$ as well as the verifier's challenge β to produce the prover's second message γ (the so-called response). In general P_0 and P_1 are allowed to share the same random tape, which we denote by $r \in \{0,1\}^*$. In a similar fashion we can think of the verifier's algorithm as split into two sub-algorithms $\mathsf{V} = (\mathsf{V}_0, \mathsf{V}_1)$, where V_0 outputs a uniformly random value $\beta \in \mathcal{B}$ and V_1 is deterministic and corresponds to the verifier's verdict (i.e., V_1 takes as input pk and a transcript (α, β, γ) and returns a decision bit $d \in \{0,1\}$).

A canonical ID scheme typically satisfies three properties known as completeness, soundness, and HVZK, which are analogues to the corresponding properties of 3PC argument systems. We provide an informal definition of completeness and soundness below (we discuss HVZK later in this section).

- **Completeness.** The honest prover P (holding sk) convinces the honest verifier V (holding pk) with overwhelming probability (over the randomness of $\mathsf{K}, \mathsf{P}, \mathsf{V}$).
- **Soundness.** For all PPT provers P^*, the probability that P^* convinces V on common input $(\mathsf{crs}, \mathsf{pk})$ is bounded by $s(\lambda) \in \mathsf{negl}(\lambda)$.[10]

Signature schemes. A signature scheme is a triple of PPT algorithms $(\mathsf{K}, \mathsf{S}, \mathsf{V})$. Algorithm K takes as input the security parameter 1^λ and outputs a pair of keys $(\mathsf{pk}, \mathsf{sk})$. Algorithm S takes as input a pair (sk, m), and outputs a signature σ on message $m \in \{0,1\}^*$. Algorithm V takes as input pk and a pair (m, σ) and returns a decision bit $d \in \{0,1\}$.

We say that a signature scheme satisfies completeness if for all $(\mathsf{pk}, \mathsf{sk})$ output by $\mathsf{K}(1^\lambda)$, and for all messages $m \in \{0,1\}^*$, we have that $\mathsf{V}(\mathsf{pk}, (m, \mathsf{S}(\mathsf{sk}, m))) = 1$ with overwhelming probability.

The standard security notion for signature schemes is called existential unforgeability against chosen-message attacks (EUF-CMA). Roughly speaking this notion demands that it should be hard to forge a signature without knowing the secret key, even when given access to an oracle, signing polynomially many messages of the adversary's choice. Here, we will instead stick to a strictly weaker notion called random-message unforgeability against random message attack (RUF-RMA). The difference is that now both signature queries and the final forgery are for messages that are not under the adversary's control (in particular they are uniformly random messages).

[10] The definition of soundness we consider is very weak, in that for instance, it is achieved by the naive protocol where P forwards sk to V. Such a protocol is not passively secure, since an adversary observing a single honest execution can later impersonate the prover. Note, however, that we additionally require a canonical ID scheme to satisfy HVZK, which implies passive security.

Definition 7.1 (RUF-RMA) *Let $\Pi = (\mathsf{K}, \mathsf{S}, \mathsf{V})$ be a signature scheme. We say that Π satisfies q-bounded random-message unforgeability against random-message attacks (q-bounded RUF-RMA) if for all PPT adversaries $\mathcal{A}$ asking at most q oracle queries the following holds:*

$$\Pr\left[\mathsf{V}(\mathsf{pk}, (m^*, \sigma^*)) = 1 : \ (\mathsf{pk}, \mathsf{sk}) \leftarrow_{\$} \mathsf{K}(1^\lambda); m^* \leftarrow_{\$} \{0,1\}^*; \sigma^* \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{sk})}(1^\lambda, \mathsf{pk}, m^*)\right] \leq \mathsf{negl}(\lambda)\,,$$

where oracle $\mathsf{Sign}(\mathsf{sk})$, upon each query, samples a fresh random message $m_i \leftarrow_{\$} \{0,1\}^*$ and returns (m_i, σ_i) with $\sigma_i \leftarrow_{\$} \mathsf{S}(\mathsf{sk}, m_i)$.

FS signatures. The FS transform allows to turn canonical ID schemes into signature schemes, as specified below. Let $\Pi = (\mathsf{K}, \mathsf{P}, \mathsf{V})$ be the initial canonical ID scheme. Additionally, consider a family of hash functions H consisting of algorithms $\mathsf{H.KGen}$, $\mathsf{H.kl}$, $\mathsf{H.Eval}$, $\mathsf{H.il}$ and $\mathsf{H.ol}$ (see Section 3.1); here $\mathsf{H.il}$ and $\mathsf{H.ol}$ correspond, respectively, to the bit lengths of messages $\alpha||m$ and β (as a function of the security parameter λ).

The FS collapse of Π using H is a triple of algorithms $\overline{\Pi}_{\mathrm{FS},\mathsf{H}} := (\mathsf{K}_{\mathrm{FS}}, \mathsf{S}_{\mathrm{FS}}, \mathsf{V}_{\mathrm{FS}})$ defined as follows.

- Algorithm K_{FS} takes as input the security parameter, samples $\mathsf{hk} \leftarrow_{\$} \mathsf{H.KGen}(1^\lambda)$, $(\mathsf{crs}, \mathsf{pk}, \mathsf{sk}) \leftarrow_{\$} \mathsf{K}(1^\lambda)$, and outputs $\overline{\mathsf{pk}} := (\mathsf{crs}, \mathsf{hk}, \mathsf{pk})$ and $\overline{\mathsf{sk}} := (\mathsf{crs}, \mathsf{hk}, \mathsf{pk}, \mathsf{sk})$.
- Algorithm S_{FS} takes as input $\overline{\mathsf{sk}} := (\mathsf{crs}, \mathsf{hk}, \mathsf{pk}, \mathsf{sk})$ and a message $m \in \{0,1\}^*$, and runs $\mathsf{P}_0(\mathsf{crs}, \mathsf{pk}, \mathsf{sk})$ in order to obtain the commitment $\alpha \in \{0,1\}^{\mathsf{H.il}(\lambda)-|m|}$; next P_{FS} defines the challenge as $\beta := \mathsf{H.Eval}(\mathsf{hk}, \alpha||m)$ and runs $\mathsf{P}_1(\mathsf{crs}, \mathsf{pk}, \mathsf{sk}, \beta)$ in order to obtain the response γ. Finally P_{FS} outputs $\sigma := (\alpha, \gamma)$.
- Algorithm V_{FS} takes as input $\overline{\mathsf{pk}} := (\mathsf{crs}, \mathsf{hk}, \mathsf{pk})$ and a pair (m, σ), and returns 1 if and only if verifier $\mathsf{V}_1(\mathsf{crs}, \mathsf{pk}, (\alpha, \beta, \gamma)) = 1$ where $\beta = \mathsf{H.Eval}(\mathsf{hk}, \alpha||m)$.

7.2 Proof of Random-Message Unforgeability

In a nutshell the result of Fiat and Shamir (for the case of signatures) says that whenever $\Pi = (\mathsf{P}, \mathsf{V})$ is a (standard-model) canonical ID scheme satisfying completeness, computational soundness, and HVZK (in addition to a basic requirement on the min-entropy of the prover's commitment), its FS collapse $\overline{\Pi}_{\mathrm{FS},\mathsf{H}}$ is an EUF-CMA signature scheme if H is modeled as a random oracle.

Here we show that the FS transform admits a very simple standard-model instantiation when starting from a special class of canonical ID schemes $\Pi = (\mathsf{K}, \mathsf{P}, \mathsf{V})$ satisfying three additional properties. The obtained signature scheme will satisfy q-bounded RUF-RMA. Since most of the technical details are identical to our standard-model instantiation for FS NIZK, we only highlight the main differences here. The properties we need are defined below:

P1′: The first property requires that the commitment α can be computed independently of $(\mathsf{pk}, \mathsf{sk})$, i.e. $\alpha \leftarrow_{\$} \mathsf{P}_0(\mathsf{crs})$. This property is analogous to "instance-independent commitments" for argument systems (cf. Definition 4.7).

P2′: The second property requires that the SEGR $\varrho(\lambda) := s(\lambda)/2^{-a(\lambda)}$ is negligible in the security parameter, where $s(\lambda)$ is the soundness parameter of the ID scheme and $a(\lambda) \in \mathbb{N}$ is the maximum bit-length of the commitment. The definition of SEGR is analogous to the definition of SEGR for argument systems (cf. Definition 4.5).

P3′: There exists a PPT simulator $\mathcal{S} := (\mathcal{S}', \mathcal{S}'')$ such that the following two distributions are computationally indistinguishable.

- Adversary $\mathcal{A}(\mathsf{crs}, \mathsf{pk})$ asking up-to q queries to an oracle returning transcripts of honest executions between $\mathsf{P}(\mathsf{crs}, \mathsf{pk}, \mathsf{sk})$ and $\mathsf{V}(\mathsf{crs}, \mathsf{pk})$, where $(\mathsf{crs}, \mathsf{pk}, \mathsf{sk})$ are generated by running algorithm K.
- Adversary $\mathcal{A}(\mathsf{crs}, \mathsf{pk})$ asking up-to q queries to $\mathcal{S}''$, where crs is generated by running $\mathcal{S}'$ (also holding a trapdoor tk) and $(\mathsf{pk}, \mathsf{sk})$ are generated using K. Here, $\mathcal{S}''$ computes a simulated transcript (α, β, γ) knowing only pk (and some trapdoor information); furthermore the simulator can compute (α, β) independently of pk.

This property is analogous to "q-bounded instance-independent HVZK" for argument systems (cf. Definition 4.8).

We will call canonical ID schemes satisfying properties $\mathbf{P1}'$-$\mathbf{P3}'$ above (besides completeness and soundness) *highly sound* canonical ID schemes.

Theorem 7.2 *Let $\Pi = (\mathsf{K}, \mathsf{P}, \mathsf{V})$ be a* highly sound *canonical ID scheme and H be a programmable q-wise independent hash function. Then, the FS collapse $\overline{\Pi}_{\mathrm{FS},\mathsf{H}}$ of Π using H yields a signature scheme satisfying q-RUF-RMA.*

Proof (sketch). Assume there exists a PPT adversary $\mathcal{A}$ and some polynomial $p(\cdot)$ such that for infinitely many values of $\lambda \in \mathbb{N}$ the following holds:

$$\Pr\left[\mathsf{V}_{\mathrm{FS}}(\overline{\mathsf{pk}}, (m^*, \sigma^*)) = 1 : \ (\overline{\mathsf{pk}}, \overline{\mathsf{sk}}) \leftarrow_{\$} \mathsf{K}_{\mathrm{FS}}(1^\lambda); m^* \leftarrow_{\$} \{0,1\}^*; \sigma^* \leftarrow \mathcal{A}^{\mathsf{Sign}_{\mathrm{FS}}(\overline{\mathsf{sk}})}(1^\lambda, \overline{\mathsf{pk}}, m^*)\right] \geq 1/p(\lambda),$$

where oracle $\mathsf{Sign}_{\mathrm{FS}}(\overline{\mathsf{sk}})$ internally runs the signing algorithm $\mathsf{S}_{\mathrm{FS}}(\overline{\mathsf{sk}}, \cdot)$.

We describe a sequence of intermediate hybrid experiments and argue indistinguishability via a standard hybrid argument.

$\mathsf{Game}_1(\lambda)$**:** This is identical to the RUF-RMA experiment, but now the randomness r_i used to generate the q signatures σ_i corresponding to $\mathcal{A}$'s signature queries, and the q messages $m_i \leftarrow_{\$} \{0,1\}^*$, are pre-sampled. Additionally, values $\alpha_i = \mathsf{P}_0(1^\lambda; r_i)$ are pre-computed—note that this is possible because of property $\mathbf{P1}'$—which allows us to also pre-compute values β_i as $\beta_i = \mathsf{H.Eval}(\mathsf{hk}, \alpha_i || m_i)$ where hk is the hash key. All of these values are stored in a trapdoor $\overline{\mathsf{tk}}$ which is given to the hybrid oracle answering signature queries.

We say that $\mathsf{Game}_1(\lambda) = 1$ if and only if the forgery returned by $\mathcal{A}$ is accepting. Observe that all of the above steps (pre-computing values) are clearly just syntactical changes, and thus $\Pr[\mathsf{Game}_1(\lambda) = 1] \geq 1/p(\lambda)$.

q-wise

$\mathsf{Game}_2(\lambda)$**:** In the second step we replace all values β_i with uniformly random values sampled from the range of hash function H and the key hk is chosen via programming the hash function. Down to the programmable q-wise independence property of H the distribution corresponding to $\mathsf{Game}_1(\lambda)$ and $\mathsf{Game}_2(\lambda)$ are identical, i.e., $\mathsf{Game}_2(\lambda) \equiv \mathsf{Game}_1(\lambda)$.

HVZK

$\mathsf{Game}_3(\lambda)$**:** In the third step we change the way signature queries are answered. In particular, instead of running $\mathsf{S}_{\mathrm{FS}}(\overline{\mathsf{sk}}, \cdot)$, we define a simulator Sim that answers signature queries only given as input $\overline{\mathsf{pk}}$ (and some trapdoor information). The simulator pre-computes all values and then relies on the HVZK simulator $\mathcal{S}$ (here is where we use property $\mathbf{P3}'$). The description of Sim is essentially identical (with minor modifications) to the simulator Sim defined in the proof of Lemma 4.9, and is therefore omitted.

Down to the q-bounded instance-independent HVZK property of Π, we can write:

$$|\Pr[\mathsf{Game}_3(\lambda) = 1] - \Pr[\mathsf{Game}_2(\lambda) = 1]| \in \mathsf{negl}(\lambda)\,.$$

Combining the above equations, we have that there exists a polynomial $\overline{p}(\cdot)$ such that, for infinitely many values of $\lambda \in \mathbb{N}$, $\Pr[\mathsf{Game}_3(\lambda) = 1] \geq 1/\overline{p}(\lambda)$.

We now use this fact to contradict the soundness of the underlying ID scheme. This last step of the proof is analogous to the proof of soundness for FS NIZK, and consists of two sub-steps that are outlined below:

- In the first step, we consider a selective variant of the FS transform, in a similar way as we did in Section 4.3 for the case of argument systems. In the selective variant, which we denote by $\Pi_{\text{sel-FS},\mathsf{H}}$, the verifier V sends a random hash key hk together with a random message $m^* \leftarrow\!\!\$\, \{0,1\}^*$ that is hashed by the prover together with the commitment α in order to generate the challenge β (and the corresponding answer γ).

  Note that the above described selective variant of the FS transform still yields a public-coin ID scheme. Furthermore, it is easy to prove that $\Pi_{\text{sel-FS},\mathsf{H}}$ satisfies both completeness and soundness, provided that the original ID scheme does. This proof is almost identical to the proof of Theorem 4.3, and is therefore omitted.

- In the second step, we use the adversary $\mathcal{A}$ (with non-negligible advantage in game $\mathsf{Game}_3(\lambda)$) to construct a prover P^* that breaks the soundness of $\Pi_{\text{sel-FS},\mathsf{H}}$ with non-negligible advantage.

  Prover P^* receives as input a pair $(\mathsf{crs}, \mathsf{pk})$, and initially picks a value α (uniformly at random from the set of all possible commitments) that it forwards to the verifier. The verifier replies with a random hash key hk and random message $m^* \in \{0,1\}^*$. At this point, P^* initializes the adversary $\mathcal{A}$ with a simulated public key $\overline{\mathsf{pk}} = (\mathsf{crs}', \mathsf{hk}, \mathsf{pk})$ (where crs' is the simulated CRS coming from Sim) and replies to $\mathcal{A}$'s signature queries as specified in $\mathsf{Game}_3(\lambda)$ (i.e., by running the simulator Sim). Finally, it forwards m^* to $\mathcal{A}$ obtaining a forgery $\sigma^* = (\alpha^*, \gamma^*)$. In case $\alpha^* = \alpha$, then P^* passes γ^* to the verifier. Else, P^* aborts.

  Clearly, the probability of P^* breaking soundness is lower bounded by the success probability of $\mathcal{A}$ times the probability that α^* is equal to α. It follows that, if $\Pi_{\text{sel-FS},\mathsf{H}}$ has soundness parameter $s(\lambda)$, the SEGR $\varrho(\lambda) := s(\lambda)/2^{-a(\lambda)} \geq 1/\overline{p}(\lambda)$, contradicting property $\mathbf{P2}'$.

The above two facts imply the statement of the theorem. □

7.3 Obtaining the Required Properties

It remains to construct a highly sound canonical ID scheme. As we briefly explain now, the latter can be done using similar techniques as the ones we used to construct highly sound 3PC arguments.

Let us start with any (standard-model) canonical ID scheme Π that satisfies completeness, soundness, HVZK, and moreover has instance-independent commitments (i.e., property $\mathbf{P1}'$). Many canonical ID schemes satisfy this requirement, including the ones by Schnorr [Sch91] and Guillou-Quisquater [GQ90]. Hence, we can apply the two compilers described in Section 5 and Section 6 in order to obtain a highly sound canonical ID scheme Π'' as follows:

- First, we transform Π into an ID scheme Π' that additionally satisfies $\mathbf{P3}'$ (i.e., the HVZK simulator is instance-independent). This is achieved by having the prover commit to the value α that would have been sent in Π using an equivocal commitment, in a similar fashion as we did in our compiler from Fig. 6.

- Second, we transform Π' into an ID scheme Π'' that additionally satisfies $\mathbf{P2}'$. This is achieved by providing a mechanism that allows to produce many challenges β given only a single commitment α, in a similar way as we did in the compiler from Fig. 4.

We remark that, for the case of FS signatures, the obtained signature scheme will be in the standard model even though the starting highly sound canonical ID scheme is in the CRS model.

Acknowledgments

We are grateful to Christina Brzuska for her active participation in this research. Her feedback and suggestions played an essential part in the development of this work.

We thank Nils Fleischhacker, Markulf Kohlweiss, Mihir Bellare, and Ivan Visconti for helpful comments on the presentation. We are grateful to an anonymous reviewer of TCC 2016 for pointing out that the constant hash function already suffices for obtaining a 1-bounded NIZK assuming properties **P1-P3**, and thereby inspiring using a q-wise independent hash-function as instantiation. Before, we used a more complicated construction based on indistinguishability obfuscation and puncturable PRFs. We also thank the same reviewer for pointing out the Blum–Lapidot–Shamir protocol, and Ivan Visconti for helpful discussions and clarifications about the protocol itself.

References

[AABN02] Michel Abdalla, Jee Hea An, Mihir Bellare, and Chanathip Namprempre. From identification to signatures via the Fiat-Shamir transform: Minimizing assumptions for security and forward-security. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 418–433, Amsterdam, The Netherlands, April 28 – May 2, 2002. Springer, Heidelberg, Germany. (Cited on page 4.)

[Bar01] Boaz Barak. How to go beyond the black-box simulation barrier. In *42nd Annual Symposium on Foundations of Computer Science*, pages 106–115, Las Vegas, NV, USA, October 14–17, 2001. IEEE Computer Society Press. (Cited on page 4.)

[BCP14] Elette Boyle, Kai-Min Chung, and Rafael Pass. On extractability obfuscation. In Yehuda Lindell, editor, *TCC 2014: 11th Theory of Cryptography Conference*, volume 8349 of *Lecture Notes in Computer Science*, pages 52–73, San Diego, CA, USA, February 24–26, 2014. Springer, Heidelberg, Germany. (Cited on pages 15, 30, and 31.)

[BDG+13] Nir Bitansky, Dana Dachman-Soled, Sanjam Garg, Abhishek Jain, Yael Tauman Kalai, Adriana López-Alt, and Daniel Wichs. Why "Fiat-Shamir for proofs" lacks a proof. In Amit Sahai, editor, *TCC 2013: 10th Theory of Cryptography Conference*, volume 7785 of *Lecture Notes in Computer Science*, pages 182–201, Tokyo, Japan, March 3–6, 2013. Springer, Heidelberg, Germany. (Cited on pages 4 and 5.)

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *20th Annual ACM Symposium on Theory of Computing*, pages 103–112, Chicago, IL, USA, May 2–4, 1988. ACM Press. (Cited on page 6.)

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 1–18, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Heidelberg, Germany. (Cited on page 14.)

[BGI+12] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):6:1–6:48, May 2012. (Cited on page 14.)

[BGW12] Nir Bitansky, Sanjam Garg, and Daniel Wichs. Why Fiat-Shamir for proofs lacks a proof. Cryptology ePrint Archive, Report 2012/705, 2012. `http://eprint.iacr.org/2012/705`. (Cited on pages 4 and 5.)

[Blu81] Manuel Blum. Coin flipping by telephone. In Allen Gersho, editor, *Advances in Cryptology – CRYPTO'81*, volume ECE Report 82-04, pages 11–15, Santa Barbara, CA, USA, 1981. U.C. Santa Barbara, Dept. of Elec. and Computer Eng. (Cited on pages 9, 10, 21, and 24.)

[BLV03] Boaz Barak, Yehuda Lindell, and Salil P. Vadhan. Lower bounds for non-black-box zero knowledge. In *44th Annual Symposium on Foundations of Computer Science*, pages 384–393, Cambridge, MA, USA, October 11–14, 2003. IEEE Computer Society Press. (Cited on page 4.)

[BPW12] David Bernhard, Olivier Pereira, and Bogdan Warinschi. How not to prove yourself: Pitfalls of the Fiat-Shamir heuristic and applications to Helios. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology – ASIACRYPT 2012*, volume 7658 of *Lecture Notes in Computer Science*, pages 626–643, Beijing, China, December 2–6, 2012. Springer, Heidelberg, Germany. (Cited on pages 4 and 6.)

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 62–73, Fairfax, Virginia, USA, November 3–5, 1993. ACM Press. (Cited on page 4.)

[BR06] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426, St. Petersburg, Russia, May 28 – June 1, 2006. Springer, Heidelberg, Germany. (Cited on page 30.)

[BS07] Mihir Bellare and Sarah Shoup. Two-tier signatures, strongly unforgeable signatures, and Fiat-Shamir without random oracles. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *PKC 2007: 10th International Conference on Theory and Practice of Public Key Cryptography*, volume 4450 of *Lecture Notes in Computer Science*, pages 201–216, Beijing, China, April 16–20, 2007. Springer, Heidelberg, Germany. (Cited on page 6.)

[BST14] Mihir Bellare, Igors Stepanovs, and Stefano Tessaro. Poly-many hardcore bits for any one-way function and a framework for differing-inputs obfuscation. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014, Part II*, volume 8874 of *Lecture Notes in Computer Science*, pages 102–121, Kaoshiung, Taiwan, R.O.C., December 7–11, 2014. Springer, Heidelberg, Germany. (Cited on page 14.)

[CCR16] Ran Canetti, Yilei Chen, and Leonid Reyzin. On the correlation intractability of obfuscated pseudorandom functions. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A: 13th Theory of Cryptography Conference, Part I*, volume 9562 of *Lecture Notes in Computer Science*, pages 389–415, Tel Aviv, Israel, January 10–13, 2016. Springer, Heidelberg, Germany. (Cited on page 4.)

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In *30th Annual ACM Symposium on Theory of Computing*, pages 209–218, Dallas, TX, USA, May 23–26, 1998. ACM Press. (Cited on page 4.)

[CHL05] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Compact e-cash. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 302–321, Aarhus, Denmark, May 22–26, 2005. Springer, Heidelberg, Germany. (Cited on page 3.)

[CL02] Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 61–76, Santa Barbara, CA, USA, August 18–22, 2002. Springer, Heidelberg, Germany. (Cited on page 3.)

[CL09] Kai-Min Chung and Feng-Hao Liu. Tight parallel repetition theorems for public-coin arguments. *Electronic Colloquium on Computational Complexity (ECCC)*, 16:109, 2009. (Cited on page 27.)

[CP15] Kai-Min Chung and Rafael Pass. Tight parallel repetition theorems for public-coin arguments using KL-divergence. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages 229–246, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany. (Cited on page 27.)

[CPS+16a] Michele Ciampi, Giuseppe Persiano, Alessandra Scafuro, Luisa Siniscalchi, and Ivan Visconti. Improved OR-composition of Sigma-protocols. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A: 13th Theory of Cryptography Conference, Part II,* volume 9563 of *Lecture Notes in Computer Science*, pages 112–141, Tel Aviv, Israel, January 10–13 2016. Springer, Berlin, Germany. (Cited on pages 8 and 21.)

[CPS+16b] Michele Ciampi, Giuseppe Persiano, Alessandra Scafuro, Luisa Siniscalchi, and Ivan Visconti. Online/offline OR composition of sigma protocols. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 63–92, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany. (Cited on pages 8 and 21.)

[CPS+16c] Michele Ciampi, Giuseppe Persiano, Alessandra Scafuro, Luisa Siniscalchi, and Ivan Visconti. A transform for NIZK almost as efficient and general as the Fiat-Shamir transform without programmable random oracles. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A: 13th Theory of Cryptography Conference, Part II,* volume 9563 of *Lecture Notes in Computer Science*, pages 83–111, Tel Aviv, Israel, January 10–13 2016. Springer, Berlin, Germany. (Cited on page 7.)

[CV05] Dario Catalano and Ivan Visconti. Hybrid trapdoor commitments and their applications. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *ICALP 2005: 32nd International Colloquium on Automata, Languages and Programming*, volume 3580 of *Lecture Notes in Computer Science*, pages 298–310, Lisbon, Portugal, July 11–15, 2005. Springer, Heidelberg, Germany. (Cited on pages 24 and 32.)

[CV07] Dario Catalano and Ivan Visconti. Hybrid commitments and their applications to zero-knowledge proof systems. *Theoretical computer science*, 374(1):229–260, 2007. (Cited on pages 24 and 32.)

[Dam00] Ivan Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 418–430, Bruges, Belgium, May 14–18, 2000. Springer, Heidelberg, Germany. (Cited on page 7.)

[DDN91] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography (extended abstract). In *23rd Annual ACM Symposium on Theory of Computing*, pages 542–552, New Orleans, LA, USA, May 6–8, 1991. ACM Press. (Cited on page 3.)

[DJKL12] Dana Dachman-Soled, Abhishek Jain, Yael Tauman Kalai, and Adriana López-Alt. On the (in)security of the Fiat-Shamir paradigm, revisited. *IACR Cryptology ePrint Archive*, 2012:706, 2012. (Cited on pages 4 and 5.)

[DNRS99] Cynthia Dwork, Moni Naor, Omer Reingold, and Larry J. Stockmeyer. Magic functions. In *40th Annual Symposium on Foundations of Computer Science*, pages 523–534, New York, NY, USA, October 17–19, 1999. IEEE Computer Society Press. (Cited on pages 4 and 5.)

[DRV12] Yevgeniy Dodis, Thomas Ristenpart, and Salil P. Vadhan. Randomness condensers for efficiently samplable, seed-dependent sources. In Ronald Cramer, editor, *TCC 2012: 9th Theory of Cryptography Conference*, volume 7194 of *Lecture Notes in Computer Science*, pages 618–635, Taormina, Sicily, Italy, March 19–21, 2012. Springer, Heidelberg, Germany. (Cited on page 4.)

[DV14] Özgür Dagdelen and Daniele Venturi. A second look at Fischlin's transformation. In David Pointcheval and Damien Vergnaud, editors, *AFRICACRYPT 14: 7th International Conference on Cryptology in Africa*, volume 8469 of *Lecture Notes in Computer Science*, pages 356–376, Marrakesh, Morocco, May 28–30, 2014. Springer, Heidelberg, Germany. (Cited on page 7.)

[EL04] Edith Elkind and Helger Lipmaa. Interleaving cryptography and mechanism design: The case of online auctions. In Ari Juels, editor, *FC 2004: 8th International Conference on Financial Cryptography*, volume 3110 of *Lecture Notes in Computer Science*, pages 117–131, Key West, USA, February 9–12, 2004. Springer, Heidelberg, Germany. (Cited on page 3.)

[FHN+12] Sebastian Faust, Carmit Hazay, Jesper Buus Nielsen, Peter Sebastian Nordholt, and Angela Zottarel. Signature schemes secure against hard-to-invert leakage. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology – ASIACRYPT 2012*, volume 7658 of *Lecture Notes in Computer Science*, pages 98–115, Beijing, China, December 2–6, 2012. Springer, Heidelberg, Germany. (Cited on page 10.)

[Fis05] Marc Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 152–168, Santa Barbara, CA, USA, August 14–18, 2005. Springer, Heidelberg, Germany. (Cited on pages 7 and 33.)

[FKMV12] Sebastian Faust, Markulf Kohlweiss, Giorgia Azzurra Marson, and Daniele Venturi. On the non-malleability of the Fiat-Shamir transform. In Steven D. Galbraith and Mridul Nandi, editors, *Progress in Cryptology - INDOCRYPT 2012: 13th International Conference in Cryptology in India*, volume 7668 of *Lecture Notes in Computer Science*, pages 60–79, Kolkata, India, December 9–12, 2012. Springer, Heidelberg, Germany. (Cited on page 4.)

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology – CRYPTO'86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194, Santa Barbara, CA, USA, August 1987. Springer, Heidelberg, Germany. (Cited on pages 3, 4, and 16.)

[GK03] Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the Fiat-Shamir paradigm. In *44th Annual Symposium on Foundations of Computer Science*, pages 102–115, Cambridge, MA, USA, October 11–14, 2003. IEEE Computer Society Press. (Cited on pages 4 and 5.)

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th Annual ACM Symposium on Theory of Computing*, pages 218–229, New York City, NY, USA, May 25–27, 1987. ACM Press. (Cited on page 3.)

[Gol01] Oded Goldreich. *Foundations of Cryptography: Basic Tools*, volume 1. Cambridge University Press, Cambridge, UK, 2001. (Cited on page 13.)

[GOSV14] Vipul Goyal, Rafail Ostrovsky, Alessandra Scafuro, and Ivan Visconti. Black-box non-black-box zero knowledge. In David B. Shmoys, editor, *46th Annual ACM Symposium on Theory of Computing*, pages 515–524, New York, NY, USA, May 31 – June 3, 2014. ACM Press. (Cited on page 5.)

[GQ90] Louis C. Guillou and Jean-Jacques Quisquater. A "paradoxical" indentity-based signature scheme resulting from zero-knowledge. In Shafi Goldwasser, editor, *Advances in Cryptology – CRYPTO'88*, volume 403 of *Lecture Notes in Computer Science*, pages 216–231, Santa Barbara, CA, USA, August 21–25, 1990. Springer, Heidelberg, Germany. (Cited on page 41.)

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 415–432, Istanbul, Turkey, April 13–17, 2008. Springer, Heidelberg, Germany. (Cited on page 6.)

[Hai09] Iftach Haitner. A parallel repetition theorem for any interactive argument. In *50th Annual Symposium on Foundations of Computer Science*, pages 241–250, Atlanta, GA, USA, October 25–27, 2009. IEEE Computer Society Press. (Cited on page 7.)

[HPWP10] Johan Håstad, Rafael Pass, Douglas Wikström, and Krzysztof Pietrzak. An efficient parallel repetition theorem. In Daniele Micciancio, editor, *TCC 2010: 7th Theory of Cryptography Conference*, volume 5978 of *Lecture Notes in Computer Science*, pages 1–18, Zurich, Switzerland, February 9–11, 2010. Springer, Heidelberg, Germany. (Cited on page 27.)

[HSW14] Susan Hohenberger, Amit Sahai, and Brent Waters. Replacing a random oracle: Full domain hash from indistinguishability obfuscation. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 201–220, Copenhagen, Denmark, May 11–15, 2014. Springer, Heidelberg, Germany. (Cited on page 6.)

[HV16] Carmit Hazay and Muthuramakrishnan Venkitasubramaniam. On the power of secure two-party computation. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 397–429, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany. (Cited on pages 8 and 21.)

[KRR16] Yael Tauman Kalai, Guy N. Rothblum, and Ron D. Rothblum. From obfuscation to the security of Fiat-Shamir for proofs. Cryptology ePrint Archive, Report 2016/303, 2016. `http://eprint.iacr.org/`. (Cited on page 7.)

[KZZ15] Aggelos Kiayias, Thomas Zacharias, and Bingsheng Zhang. End-to-end verifiable elections in the standard model. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 468–498, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany. (Cited on page 3.)

[Lin15] Yehuda Lindell. An efficient transform from sigma protocols to NIZK with a CRS and non-programmable random oracle. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part I*, volume 9014 of *Lecture Notes in Computer Science*, pages 93–109, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany. (Cited on pages 7, 24, and 32.)

[LS91] Dror Lapidot and Adi Shamir. Publicly verifiable non-interactive zero-knowledge proofs. In Alfred J. Menezes and Scott A. Vanstone, editors, *Advances in Cryptology – CRYPTO'90*, volume 537 of *Lecture Notes in Computer Science*, pages 353–365, Santa Barbara, CA, USA, August 11–15, 1991. Springer, Heidelberg, Germany. (Cited on pages 5, 9, 21, 22, and 24.)

[NVZ14] Jesper Buus Nielsen, Daniele Venturi, and Angela Zottarel. Leakage-resilient signatures with graceful degradation. In Hugo Krawczyk, editor, *PKC 2014: 17th International Conference on Theory and Practice of Public Key Cryptography*, volume 8383 of *Lecture Notes in Computer Science*, pages 362–379, Buenos Aires, Argentina, March 26–28, 2014. Springer, Heidelberg, Germany. (Cited on page 10.)

[Oka93] Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In Ernest F. Brickell, editor, *Advances in Cryptology – CRYPTO'92*, volume 740 of *Lecture Notes in Computer Science*, pages 31–53, Santa Barbara, CA, USA, August 16–20, 1993. Springer, Heidelberg, Germany. (Cited on page 4.)

[OV12] Rafail Ostrovsky and Ivan Visconti. Simultaneous resettability from collision resistance. *Electronic Colloquium on Computational Complexity (ECCC)*, 19:164, 2012. (Cited on page 9.)

[PS00] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000. (Cited on page 4.)

[PV07] Rafael Pass and Muthuramakrishnan Venkitasubramaniam. An efficient parallel repetition theorem for Arthur-Merlin games. In David S. Johnson and Uriel Feige, editors, *39th Annual ACM Symposium on Theory of Computing*, pages 420–429, San Diego, CA, USA, June 11–13, 2007. ACM Press. (Cited on page 27.)

[Sch91] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991. (Cited on page 41.)

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *46th Annual ACM Symposium on Theory of Computing*, pages 475–484, New York, NY, USA, May 31 – June 3, 2014. ACM Press. (Cited on pages 6 and 16.)

$\mathsf{Game}_1(\lambda)$ —iO (start puncturing τ-s)→ $\mathsf{Game}_2(\lambda)$ —pPRF (complete puncturing τ-s)→ $\mathsf{Game}_3(\lambda)$ —bad_{α^*} (choose α^* outside image)→

$\mathsf{K}^*(1^\lambda)$

	Game_1	Game_2	Game_3
1 :	$\mathsf{crs} \leftarrow\$ \mathsf{K}(1^\lambda)$	$\mathsf{crs} \leftarrow\$ \mathsf{K}(1^\lambda)$	$\mathsf{crs} \leftarrow\$ \mathsf{K}(1^\lambda)$
2 :	$\mathsf{k}_1 \leftarrow\$ \mathsf{F}_1.\mathsf{KGen}(1^\lambda)$	$\mathsf{k}_1 \leftarrow\$ \mathsf{F}_1.\mathsf{KGen}(1^\lambda)$	$\mathsf{k}_1 \leftarrow\$ \mathsf{F}_1.\mathsf{KGen}(1^\lambda)$
3 :	$\mathsf{k}_2 \leftarrow\$ \mathsf{F}_2.\mathsf{KGen}(1^\lambda)$	$\mathsf{k}_2 \leftarrow\$ \mathsf{F}_2.\mathsf{KGen}(1^\lambda)$	$\mathsf{k}_2 \leftarrow\$ \mathsf{F}_2.\mathsf{KGen}(1^\lambda)$
4 :		**for** $i = 1, \ldots, q$ **do**	**for** $i = 1, \ldots, q$ **do**
5 :		$\boldsymbol{\tau}[i] \leftarrow\$ \{0,1\}^{\mathsf{F}_1.\mathsf{il}(\lambda)}$	$\boldsymbol{\tau}[i] \leftarrow\$ \{0,1\}^{\mathsf{F}_1.\mathsf{il}(\lambda)}$
6 :		$\overline{\mathbf{p}}_\tau[i] \leftarrow \mathsf{owf}(\boldsymbol{\tau}[i])$	$\overline{\mathbf{p}}_\tau[i] \leftarrow \mathsf{owf}(\boldsymbol{\tau}[i])$
7 :		$\boldsymbol{\alpha}^*[i] \leftarrow \mathsf{F}_1.\mathsf{Eval}(\boldsymbol{\tau}[i])$	$\boldsymbol{\alpha}^*[i] \leftarrow\$ \{0,1\}^{\mathsf{F}_1.\mathsf{ol}(\lambda)}$
8 :			
9 :			
10 :		**if** bad_τ **then abort**	**if** bad_τ **then abort**
11 :		$\mathsf{k}_1^* \leftarrow\$ \mathsf{F}_1.\mathsf{Pntr}(\mathsf{k}_1, \{\boldsymbol{\tau}[i], \ldots, \boldsymbol{\tau}[q]\})$	$\mathsf{k}_1^* \leftarrow\$ \mathsf{F}_1.\mathsf{Pntr}(\mathsf{k}_1, \{\boldsymbol{\tau}[i], \ldots, \boldsymbol{\tau}[q]\})$
12 :	$\overline{C}_0 \leftarrow\$ \mathsf{iO}(C_0[\mathsf{k}_1])$	$\overline{C}_0 \leftarrow\$ \mathsf{iO}(C_0[\mathsf{k}_1^*, \overline{\mathbf{p}}_\tau, \boldsymbol{\alpha}^*])$	$\overline{C}_0 \leftarrow\$ \mathsf{iO}(C_0[\mathsf{k}_1^*, \overline{\mathbf{p}}_\tau, \boldsymbol{\alpha}^*])$
13 :	$\overline{C}_1 \leftarrow\$ \mathsf{iO}(C_1[\mathsf{k}_1, \mathsf{k}_2])$	$\overline{C}_1 \leftarrow\$ \mathsf{iO}(C_1[\mathsf{k}_1^*, \overline{\mathbf{p}}_\tau, \boldsymbol{\alpha}^*, \mathsf{k}_2])$	$\overline{C}_1 \leftarrow\$ \mathsf{iO}(C_1[\mathsf{k}_1^*, \overline{\mathbf{p}}_\tau, \boldsymbol{\alpha}^*, \mathsf{k}_2])$
14 :	**return** $(\mathsf{crs}, \overline{C}_0, \overline{C}_1)$	**return** $(\mathsf{crs}, \overline{C}_0, \overline{C}_1)$	**return** $(\mathsf{crs}, \overline{C}_0, \overline{C}_1)$

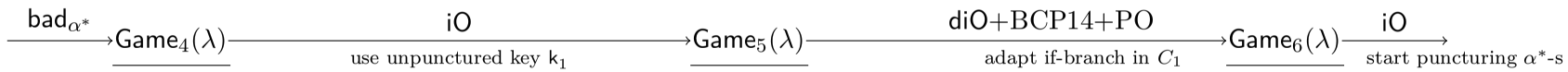
$C_0[\mathsf{k}_1](\tau)$

1 :		**if** $\exists i \in [q] : \overline{\mathbf{p}}_\tau[i](\tau) = 1$ **then**	**if** $\exists i \in [q] : \overline{\mathbf{p}}_\tau[i](\tau) = 1$ **then**
2 :		**return** $\boldsymbol{\alpha}^*[i]$	**return** $\boldsymbol{\alpha}^*[i]$
3 :	$\alpha^* \leftarrow \mathsf{F}_1.\mathsf{Eval}(\mathsf{k}_1, \tau)$	$\alpha^* \leftarrow \mathsf{F}_1.\mathsf{Eval}(\mathsf{k}_1^*, \tau)$	$\alpha^* \leftarrow \mathsf{F}_1.\mathsf{Eval}(\mathsf{k}_1^*, \tau)$
4 :	**return** α^*	**return** α^*	**return** α^*

$C_1[\mathsf{k}_1, \mathsf{k}_2](\alpha^*, \tau)$

1 :			
2 :			
3 :			
4 :	$r^* \leftarrow \mathsf{F}_2.\mathsf{Eval}(\mathsf{k}_2, \alpha^*)$	$r^* \leftarrow \mathsf{F}_2.\mathsf{Eval}(\mathsf{k}_2, \alpha^*)$	$r^* \leftarrow \mathsf{F}_2.\mathsf{Eval}(\mathsf{k}_2, \alpha^*)$
5 :	$\alpha \leftarrow \mathsf{P}_0(r^*)$	$\alpha \leftarrow \mathsf{P}_0(r^*)$	$\alpha \leftarrow \mathsf{P}_0(r^*)$
6 :	**if** $\alpha^* \neq \mathsf{F}_1.\mathsf{Eval}(\mathsf{k}_1, \tau)$ **then**	**if** $(\exists i \in [q] : \overline{\mathbf{p}}_\tau[i](\tau) = 1 \wedge \alpha^* \neq \boldsymbol{\alpha}^*[i])$ **or**	**if** $(\exists i \in [q] : \overline{\mathbf{p}}_\tau[i](\tau) = 1] \wedge \alpha^* \neq \boldsymbol{\alpha}^*[i])$ **or**
7 :		$(\forall i \in [q] : \overline{\mathbf{p}}_\tau[i](\tau) = 0 \wedge \alpha^* \neq \mathsf{F}_1.\mathsf{Eval}(\mathsf{k}_1^*, \tau))$ **then**	$(\forall i \in [q] : \overline{\mathbf{p}}_\tau[i](\tau) = 0 \wedge \alpha^* \neq \mathsf{F}_1.\mathsf{Eval}(\mathsf{k}_1^*, \tau))$ **then**
8 :	$r^* \leftarrow \perp$	$r^* \leftarrow \perp$	$r^* \leftarrow \perp$
9 :	**return** (α, r^*)	**return** (α, r^*)	**return** (α, r^*)

Figure 7: Pseudocode for proof of zero-knowledge for the compiled protocol from Section 5.



$\mathsf{K}^*(1^\lambda)$

$\mathsf{crs} \leftarrow\!\$\, \mathsf{K}(1^\lambda)$
$\mathsf{k}_1 \leftarrow\!\$\, \mathsf{F}_1.\mathsf{KGen}(1^\lambda)$
$\mathsf{k}_2 \leftarrow\!\$\, \mathsf{F}_2.\mathsf{KGen}(1^\lambda)$
for $i = 1, \ldots, q$ **do**
  $\boldsymbol{\tau}[i] \leftarrow\!\$\, \{0,1\}^{\mathsf{F}.\mathsf{il}(\lambda)}$
  $\overline{\mathbf{p}}_\tau[i] \leftarrow \mathsf{owf}(\boldsymbol{\tau}[i])$
  $\boldsymbol{\alpha}^*[i] \leftarrow\!\$\, \{0,1\}^{\mathsf{F}_1.\mathsf{ol}(\lambda)}$

if $\mathsf{bad}_\tau \vee \mathsf{bad}_{\alpha^*}$ **then abort**
$\mathsf{k}_1^* \leftarrow\!\$\, \mathsf{F}_1.\mathsf{Pntr}(\mathsf{k}_1, \{\boldsymbol{\tau}[i], \ldots, \boldsymbol{\tau}[q]\})$
$\overline{C}_0 \leftarrow\!\$\, \mathsf{iO}(C_0[\mathsf{k}_1^*, \overline{\mathbf{p}}_\tau, \boldsymbol{\alpha}^*])$
$\overline{C}_1 \leftarrow\!\$\, \mathsf{iO}(C_1[\mathsf{k}_1^*, \overline{\mathbf{p}}_\tau, \boldsymbol{\alpha}^*, \mathsf{k}_2])$
return $(\mathsf{crs}, \overline{C}_0, \overline{C}_1)$

$\mathsf{K}^*(1^\lambda)$

$\mathsf{crs} \leftarrow\!\$\, \mathsf{K}(1^\lambda)$
$\mathsf{k}_1 \leftarrow\!\$\, \mathsf{F}_1.\mathsf{KGen}(1^\lambda)$
$\mathsf{k}_2 \leftarrow\!\$\, \mathsf{F}_2.\mathsf{KGen}(1^\lambda)$
for $i = 1, \ldots, q$ **do**
  $\boldsymbol{\tau}[i] \leftarrow\!\$\, \{0,1\}^{\mathsf{F}_1.\mathsf{il}(\lambda)}$
  $\overline{\mathbf{p}}_\tau[i] \leftarrow \mathsf{owf}(\boldsymbol{\tau}[i])$
  $\boldsymbol{\alpha}^*[i] \leftarrow\!\$\, \{0,1\}^{\mathsf{F}_1.\mathsf{ol}(\lambda)}$

if $\mathsf{bad}_\tau \vee \mathsf{bad}_{\alpha^*}$ **then abort**

$\overline{C}_0 \leftarrow\!\$\, \mathsf{iO}(C_0[\mathsf{k}_1, \overline{\mathbf{p}}, \boldsymbol{\alpha}^*])$
$\overline{C}_1 \leftarrow\!\$\, \mathsf{iO}(C_1[\mathsf{k}_1, \overline{\mathbf{p}}, \boldsymbol{\alpha}^*, \mathsf{k}_2])$
return $(\mathsf{crs}, \overline{C}_0, \overline{C}_1)$

$\mathsf{K}^*(1^\lambda)$

$\mathsf{crs} \leftarrow\!\$\, \mathsf{K}(1^\lambda)$
$\mathsf{k}_1 \leftarrow\!\$\, \mathsf{F}_1.\mathsf{KGen}(1^\lambda)$
$\mathsf{k}_2 \leftarrow\!\$\, \mathsf{F}_2.\mathsf{KGen}(1^\lambda)$
for $i = 1, \ldots, q$ **do**
  $\boldsymbol{\tau}[i] \leftarrow\!\$\, \{0,1\}^{\mathsf{F}_1.\mathsf{il}(\lambda)}$
  $\overline{\mathbf{p}}_\tau[i] \leftarrow \mathsf{owf}(\boldsymbol{\tau}[i])$
  $\boldsymbol{\alpha}^*[i] \leftarrow\!\$\, \{0,1\}^{\mathsf{F}_1.\mathsf{ol}(\lambda)}$

if $\mathsf{bad}_\tau \vee \mathsf{bad}_{\alpha^*}$ **then abort**

$\overline{C}_0 \leftarrow\!\$\, \mathsf{iO}(C_0[\mathsf{k}_1, \overline{\mathbf{p}}, \boldsymbol{\alpha}^*])$
$\overline{C}_1 \leftarrow\!\$\, \mathsf{iO}(C_1[\mathsf{k}_1, \boldsymbol{\alpha}^*, \mathsf{k}_2])$
return $(\mathsf{crs}, \overline{C}_0, \overline{C}_1)$

$C_0[\mathsf{k}_1^*, \overline{\mathbf{p}}_\tau, \boldsymbol{\alpha}^*](\tau)$

if $\exists i \in [q] : \overline{\mathbf{p}}_\tau[i](\tau) = 1$ **then**
  return $\boldsymbol{\alpha}^*[i]$
$\alpha^* \leftarrow \mathsf{F}_1.\mathsf{Eval}(\mathsf{k}_1^*, \tau)$
return α^*

$C_0[\mathsf{k}_1, \overline{\mathbf{p}}, \boldsymbol{\alpha}^*](\tau)$

if $\exists i \in [q] : \overline{\mathbf{p}}[i] = \mathsf{owf}(\tau)$ **then**
  return $\boldsymbol{\alpha}^*[i]$
$\alpha^* \leftarrow \mathsf{F}_1.\mathsf{Eval}(\mathsf{k}_1, \tau)$
return α^*

$C_0[\mathsf{k}_1, \overline{\mathbf{p}}, \boldsymbol{\alpha}^*](\tau)$

if $\exists i \in [q] : \overline{\mathbf{p}}[i] = \mathsf{owf}(\tau)$ **then**
  return $\boldsymbol{\alpha}^*[i]$
$\alpha^* \leftarrow \mathsf{F}_1.\mathsf{Eval}(\mathsf{k}_1, \tau)$
return α^*

$C_1[\mathsf{k}_1^*, \overline{\mathbf{p}}_\tau, \boldsymbol{\alpha}^*, \mathsf{k}_2](\alpha^*, \tau)$

$r^* \leftarrow \mathsf{F}_2.\mathsf{Eval}(\mathsf{k}_2, \alpha^*)$
$\alpha \leftarrow \mathsf{P}_0(r^*)$
if $(\exists i \in [q] : \overline{\mathbf{p}}_\tau[i](\tau) = 1 \wedge \alpha^* \neq \boldsymbol{\alpha}^*[i])$ **or**
  $(\forall i \in [q] : \overline{\mathbf{p}}_\tau[i](\tau) = 0 \wedge \alpha^* \neq \mathsf{F}_1.\mathsf{Eval}(\mathsf{k}_1^*, \tau))$ **then**
  $r^* \leftarrow \bot$
return (α, r^*)

$C_1[\mathsf{k}_1, \overline{\mathbf{p}}, \boldsymbol{\alpha}^*, \mathsf{k}_2](\alpha^*, \tau)$

$r^* \leftarrow \mathsf{F}_2.\mathsf{Eval}(\mathsf{k}_2, \alpha^*)$
$\alpha \leftarrow \mathsf{P}_0(r^*)$
if $(\exists i \in [q] : \overline{\mathbf{p}}[i] = \mathsf{owf}(\tau) \wedge \alpha^* \neq \boldsymbol{\alpha}^*[i])$ **or**
  $(\forall i \in [q] : \overline{\mathbf{p}}[i] \neq \mathsf{owf}(\tau) \wedge \alpha^* \neq \mathsf{F}_1.\mathsf{Eval}(\mathsf{k}_1, \tau))$ **then**
  $r^* \leftarrow \bot$
return (α, r^*)

$C_1[\mathsf{k}_1, \boldsymbol{\alpha}^*, \mathsf{k}_2](\alpha^*, \tau)$

$r^* \leftarrow \mathsf{F}_2.\mathsf{Eval}(\mathsf{k}_2, \alpha^*)$
$\alpha \leftarrow \mathsf{P}_0(r^*)$
if $\alpha^* \neq \mathsf{F}_1.\mathsf{Eval}(\mathsf{k}_1, \tau)$ **then**
  $r^* \leftarrow \bot$
return (α, r^*)

Figure 8: Pseudocode for proof of zero-knowledge for the compiled protocol from Section 5.

$\xrightarrow{\mathsf{iO}} \mathsf{Game}_7(\lambda) \xrightarrow[\text{complete puncturing } \alpha^*\text{-s}]{\mathsf{pPRF}} \mathsf{Game}_8(\lambda) \xrightarrow[\text{presample commitments } \alpha]{\mathsf{iO}} \mathsf{Game}_9(\lambda) \xrightarrow[\text{use simulator to presample commitments}]{\mathsf{hvzk}} \mathsf{Game}_{10}(\lambda)$

$\mathsf{Game}_7(\lambda)$

$\mathsf{K}^*(1^\lambda)$

$\mathsf{crs} \leftarrow\$ \mathsf{K}(1^\lambda)$
$\mathsf{k}_1 \leftarrow\$ \mathsf{F}_1.\mathsf{KGen}(1^\lambda)$
$\mathsf{k}_2 \leftarrow\$ \mathsf{F}_2.\mathsf{KGen}(1^\lambda)$
for $i = 1, \ldots, q$ **do**
  $\boldsymbol{\tau}[i] \leftarrow\$ \{0,1\}^{\mathsf{F}_1.\mathsf{il}(\lambda)}$
  $\overline{\mathbf{p}}_\tau[i] \leftarrow \mathsf{owf}(\boldsymbol{\tau}[i])$
  $\boldsymbol{\alpha}^*[i] \leftarrow\$ \{0,1\}^{\mathsf{F}_1.\mathsf{ol}(\lambda)}$
  $\mathbf{r}^*[i] \leftarrow \mathsf{F}_2.\mathsf{Eval}(\mathsf{k}_2, \boldsymbol{\alpha}^*[i])$
if $\mathsf{bad}_\tau \vee \mathsf{bad}_{\alpha^*}$ **then abort**
$\mathsf{k}_2^* \leftarrow\$ \mathsf{F}_2.\mathsf{Pntr}(\mathsf{k}_2, \{\boldsymbol{\alpha}^*[1], \ldots, \boldsymbol{\alpha}^*[q]\})$
$\overline{C}_0 \leftarrow\$ \mathsf{iO}(C_0[\mathsf{k}_1, \overline{\mathbf{p}}, \boldsymbol{\alpha}^*])$
$\overline{C}_1 \leftarrow\$ \mathsf{iO}(C_1[\mathsf{k}_1, \boldsymbol{\alpha}^*, \mathsf{k}_2^*, \mathbf{r}^*])$
return $(\mathsf{crs}, \overline{C}_0, \overline{C}_1)$

$C_0[\mathsf{k}_1, \overline{\mathbf{p}}, \boldsymbol{\alpha}^*](\tau)$

if $\exists i \in [q] : \overline{\mathbf{p}}[i] = \mathsf{owf}(\tau)$ **then**
  return $\boldsymbol{\alpha}^*[i]$
$\alpha^* \leftarrow \mathsf{F}_1.\mathsf{Eval}(\mathsf{k}_1, \tau)$
return α^*

$C_1[\mathsf{k}_1, \boldsymbol{\alpha}^*, \mathsf{k}_2^*, \mathbf{r}^*](\alpha^*, \tau)$

if $\exists i \in [q] : \alpha^* = \boldsymbol{\alpha}^*[i]$
  $\alpha \leftarrow \mathsf{P}_0(\mathbf{r}^*[i])$
else
  $r^* \leftarrow \mathsf{F}_2.\mathsf{Eval}(\mathsf{k}_2^*, \alpha^*)$
  $\alpha \leftarrow \mathsf{P}_0(r^*)$
if $\alpha^* \neq \mathsf{F}_1.\mathsf{Eval}(\mathsf{k}_1, \tau)$ **then**
  $r^* \leftarrow \bot$
return (α, r^*)

$\mathsf{Game}_8(\lambda)$

$\mathsf{K}^*(1^\lambda)$

$\mathsf{crs} \leftarrow\$ \mathsf{K}(1^\lambda)$
$\mathsf{k}_1 \leftarrow\$ \mathsf{F}_1.\mathsf{KGen}(1^\lambda)$
$\mathsf{k}_2 \leftarrow\$ \mathsf{F}_2.\mathsf{KGen}(1^\lambda)$
for $i = 1, \ldots, q$ **do**
  $\boldsymbol{\tau}[i] \leftarrow\$ \{0,1\}^{\mathsf{F}_1.\mathsf{il}(\lambda)}$
  $\overline{\mathbf{p}}_\tau[i] \leftarrow \mathsf{owf}(\boldsymbol{\tau}[i])$
  $\boldsymbol{\alpha}^*[i] \leftarrow\$ \{0,1\}^{\mathsf{F}_1.\mathsf{ol}(\lambda)}$
  $\mathbf{r}^*[i] \leftarrow\$ \{0,1\}^{\mathsf{F}_2.\mathsf{ol}(\lambda)}$
if $\mathsf{bad}_\tau \vee \mathsf{bad}_{\alpha^*}$ **then abort**
$\mathsf{k}_2^* \leftarrow\$ \mathsf{F}_2.\mathsf{Pntr}(\mathsf{k}_2, \{\boldsymbol{\alpha}^*[1], \ldots, \boldsymbol{\alpha}^*[q]\})$
$\overline{C}_0 \leftarrow\$ \mathsf{iO}(C_0[\mathsf{k}_1, \overline{\mathbf{p}}, \boldsymbol{\alpha}^*])$
$\overline{C}_1 \leftarrow\$ \mathsf{iO}(C_1[\mathsf{k}_1, \boldsymbol{\alpha}^*, \mathsf{k}_2^*, \mathbf{r}^*])$
return $(\mathsf{crs}, \overline{C}_0, \overline{C}_1)$

$C_0[\mathsf{k}_1, \overline{\mathbf{p}}, \boldsymbol{\alpha}^*](\tau)$

if $\exists i \in [q] : \overline{\mathbf{p}}[i] = \mathsf{owf}(\tau)$ **then**
  return $\boldsymbol{\alpha}^*[i]$
$\alpha^* \leftarrow \mathsf{F}_1.\mathsf{Eval}(\mathsf{k}_1, \tau)$
return α^*

$C_1[\mathsf{k}_1, \boldsymbol{\alpha}^*, \mathsf{k}_2^*, \mathbf{r}^*](\alpha^*, \tau)$

if $\exists i \in [q] : \alpha^* = \boldsymbol{\alpha}^*[i]$
  $\alpha \leftarrow \mathsf{P}_0(\mathbf{r}^*[i])$
else
  $r^* \leftarrow \mathsf{F}_2.\mathsf{Eval}(\mathsf{k}_2^*, \alpha^*)$
  $\alpha \leftarrow \mathsf{P}_0(r^*)$
if $\alpha^* \neq \mathsf{F}_1.\mathsf{Eval}(\mathsf{k}_1, \tau)$ **then**
  $r^* \leftarrow \bot$
return (α, r^*)

$\mathsf{Game}_9(\lambda)$

$\mathsf{K}^*(1^\lambda)$

$\mathsf{crs} \leftarrow\$ \mathsf{K}(1^\lambda)$
$\mathsf{k}_1 \leftarrow\$ \mathsf{F}_1.\mathsf{KGen}(1^\lambda)$
$\mathsf{k}_2 \leftarrow\$ \mathsf{F}_2.\mathsf{KGen}(1^\lambda)$
for $i = 1, \ldots, q$ **do**
  $\boldsymbol{\tau}[i] \leftarrow\$ \{0,1\}^{\mathsf{F}_1.\mathsf{il}(\lambda)}$
  $\overline{\mathbf{p}}_\tau[i] \leftarrow \mathsf{owf}(\boldsymbol{\tau}[i])$
  $\boldsymbol{\alpha}^*[i] \leftarrow\$ \{0,1\}^{\mathsf{F}_1.\mathsf{ol}(\lambda)}$
  $\mathbf{r}^*[i] \leftarrow\$ \{0,1\}^{\mathsf{F}_2.\mathsf{ol}(\lambda)}$
  $\boldsymbol{\alpha}[i] \leftarrow\$ \mathsf{P}_0(\mathbf{r}^*[i])$
if $\mathsf{bad}_\tau \vee \mathsf{bad}_{\alpha^*}$ **then abort**
$\mathsf{k}_2^* \leftarrow\$ \mathsf{F}_2.\mathsf{Pntr}(\mathsf{k}_2, \{\boldsymbol{\alpha}^*[1], \ldots, \boldsymbol{\alpha}^*[q]\})$
$\overline{C}_0 \leftarrow\$ \mathsf{iO}(C_0[\mathsf{k}_1, \overline{\mathbf{p}}, \boldsymbol{\alpha}^*])$
$\overline{C}_1 \leftarrow\$ \mathsf{iO}(C_1[\mathsf{k}_1, \boldsymbol{\alpha}^*, \mathsf{k}_2^*, \boldsymbol{\alpha}])$
return $(\mathsf{crs}, \overline{C}_0, \overline{C}_1)$

$C_0[\mathsf{k}_1, \overline{\mathbf{p}}, \boldsymbol{\alpha}^*](\tau)$

if $\exists i \in [q] : \overline{\mathbf{p}}[i] = \mathsf{owf}(\tau)$ **then**
  return $\boldsymbol{\alpha}^*[i]$
$\alpha^* \leftarrow \mathsf{F}_1.\mathsf{Eval}(\mathsf{k}_1, \tau)$
return α^*

$C_1[\mathsf{k}_1, \boldsymbol{\alpha}^*, \mathsf{k}_2^*, \boldsymbol{\alpha}](\alpha^*, \tau)$

if $\exists i \in [q] : \alpha^* = \boldsymbol{\alpha}^*[i]$
  $\alpha \leftarrow \boldsymbol{\alpha}[i]$
else
  $r^* \leftarrow \mathsf{F}_2.\mathsf{Eval}(\mathsf{k}_2^*, \alpha^*)$
  $\alpha \leftarrow \mathsf{P}_0(r^*)$
if $\alpha^* \neq \mathsf{F}_1.\mathsf{Eval}(\mathsf{k}_1, \tau)$ **then**
  $r^* \leftarrow \bot$
return (α, r^*)

$\mathsf{Game}_{10}(\lambda)$

$\mathsf{K}^*(1^\lambda)$

$(\mathsf{crs}, \mathsf{tk}) \leftarrow\$ \mathcal{S}'(1^\lambda)$
$\mathsf{k}_1 \leftarrow\$ \mathsf{F}_1.\mathsf{KGen}(1^\lambda)$
$\mathsf{k}_2 \leftarrow\$ \mathsf{F}_2.\mathsf{KGen}(1^\lambda)$
for $i = 1, \ldots, q$ **do**
  $\boldsymbol{\tau}[i] \leftarrow\$ \{0,1\}^{\mathsf{F}_1.\mathsf{il}(\lambda)}$
  $\overline{\mathbf{p}}_\tau[i] \leftarrow \mathsf{owf}(\boldsymbol{\tau}[i])$
  $\boldsymbol{\alpha}^*[i] \leftarrow\$ \{0,1\}^{\mathsf{F}_1.\mathsf{ol}(\lambda)}$
  $\mathbf{r}^*[i] \leftarrow\$ \{0,1\}^{\mathsf{F}_2.\mathsf{ol}(\lambda)}$
  $(\boldsymbol{\alpha}[i], \beta_{\mathsf{tmp}}) \leftarrow\$ \mathcal{S}_0''(\mathsf{crs}, \mathsf{tk}; \mathbf{r}^*[i])$
if $\mathsf{bad}_\tau \vee \mathsf{bad}_{\alpha^*}$ **then abort**
$\mathsf{k}_2^* \leftarrow\$ \mathsf{F}_2.\mathsf{Pntr}(\mathsf{k}_2, \{\boldsymbol{\alpha}^*[1], \ldots, \boldsymbol{\alpha}^*[q]\})$
$\overline{C}_0 \leftarrow\$ \mathsf{iO}(C_0[\mathsf{k}_1, \overline{\mathbf{p}}, \boldsymbol{\alpha}^*])$
$\overline{C}_1 \leftarrow\$ \mathsf{iO}(C_1[\mathsf{k}_1, \boldsymbol{\alpha}^*, \mathsf{k}_2^*, \boldsymbol{\alpha}])$
return $(\mathsf{crs}, \overline{C}_0, \overline{C}_1)$

$C_0[\mathsf{k}_1, \overline{\mathbf{p}}, \boldsymbol{\alpha}^*](\tau)$

if $\exists i \in [q] : \overline{\mathbf{p}}[i] = \mathsf{owf}(\tau)$ **then**
  return $\boldsymbol{\alpha}^*[i]$
$\alpha^* \leftarrow \mathsf{F}_1.\mathsf{Eval}(\mathsf{k}_1, \tau)$
return α^*

$C_1[\mathsf{k}_1, \overline{\mathbf{p}}, \boldsymbol{\alpha}^*, \mathsf{k}_2^*, \boldsymbol{\alpha}](\alpha^*, \tau)$

if $\exists i \in [q] : \alpha^* = \boldsymbol{\alpha}^*[i]$
  $\alpha \leftarrow \boldsymbol{\alpha}[i]$
else
  $r^* \leftarrow \mathsf{F}_2.\mathsf{Eval}(\mathsf{k}_2^*, \alpha^*)$
  $\alpha \leftarrow \mathsf{P}_0(r^*)$
if $\alpha^* \neq \mathsf{F}_1.\mathsf{Eval}(\mathsf{k}_1, \tau)$ **then**
  $r^* \leftarrow \bot$
return (α, r^*)

Figure 9: Pseudocode for proof of zero-knowledge for the compiled protocol from Section 5.