Adaptive UC NIZK for Practical Applications

Anna Lysyanskaya[1] and Leah Namisa Rosenbloom[2]

[1] Brown University, Providence RI 02906, USA
anna_lysyanskaya@brown.edu
[2] Northeastern University, Boston MA 02115, USA
l.rosenbloom@northeastern.edu

Abstract. Non-interactive zero-knowledge proofs of knowledge (NIZKPoK) serve as a key building block in many important cryptographic protocols. In practice, NIZKPoK are often constructed from Σ-protocols based on the discrete logarithm assumption, and proven secure in the random-oracle model (ROM). Recent work and standardization initiatives have highlighted the importance of concurrent security of NIZKPoK against adversaries who are capable of adaptive corruption.

In this paper, we focus explicitly on Σ-protocols from linear function families, which include those for proving discrete-logarithm based relations. We define a natural adaptivity property for this class of Σ-protocols and show that, given any straight-line compiler which preserves the adaptivity property, it is possible to transform these protocols into adaptive, universally composable (UC) NIZKPoK in the global ROM. We then prove that the most efficient known construction of a straight-line compiler, Kondi and shelat's randomized version of Fischlin's transform [41,34], preserves the adaptivity property. Our construction avoids introducing any additional cryptographic primitives, and, following Kondi and shelat, requires only a polylogarithmic number of repetitions of the underlying Σ-protocol.

Table of Contents

1 Introduction **3**

2 Preliminaries **7**
2.1 The Global Random Oracle Model(s) 7
2.2 The $\mathcal{G}_{\mathtt{roRO}}$-$\mathcal{F}_{\mathtt{CRS}}$-hybrid Model 8
2.3 UC Security with Adaptive Corruptions 8

3 Adaptive Σ-protocols and Friends **9**
3.1 Adaptive Σ-protocols 9
3.2 OR-protocols 12
3.3 Adaptive Witness Indistinguishability 14
3.4 Simple Adaptive Σ-protocol Abstraction 16

4 Adaptive Straight-Line Compilers **17**
4.1 Definition of an Adaptive Straight-Line Compiler 18
4.2 The Randomized Fischlin Transform is an Adaptive SLC 19

5 Constructing Adaptive, Efficient UC NIZKPoK **26**
5.1 The Adaptive NIZKPoK Ideal Functionality 26
5.2 Adaptive UC NIZKPoK are adNIM-SHVZK and adNI-SSS 27
5.3 Adaptive UC NIZKPoK in the Programmable Global ROM 27
5.4 Adaptive UC NIZKPoK in the Global RO-CRS Hybrid Model . . 28

A Appendix **36**
A.1 Notation 36
A.2 Global Random Oracles 36
A.3 $\mathcal{G}_{\mathtt{roRO}}$-$\mathcal{F}_{\mathtt{CRS}}$-hybrid Model Requirements 37
A.4 Adaptive Corruptions in the UC Model 37
A.5 Adaptive Σ-protocol Template 38
A.6 Standard Completeness and Two-Special Soundness 38
A.7 adSHVZK and RE Imply Adaptive WI Full Proof 39
A.8 The Standard Random Oracle Model 41
A.9 Non-Interactive Special Soundness 41
A.10 aUC NIZKPoK Required Properties Full Proof 42
A.11 aUC NIZKPoK in the $\mathcal{G}_{\mathtt{rpoRO}}$-hybrid Model Full Proof 44
A.12 Requirements of the Randomized Fischlin Transform 46
A.13 Discussion of Modules 47
A.14 $\Sigma_{\mathtt{simple}}$ is Adaptive Full Proof 47
A.15 PoK DL is an Adaptive Σ-protocol Proof 48
A.16 PoK EqRep is an Adaptive Σ-protocol Proof 48
A.17 Efficiency of the Randomized Fischlin Transform 50

1 Introduction

Cryptographic protocols that maintain security under concurrent composition and against adversaries who can corrupt participants adaptively are vital to the formation of cryptographic standards that capture real-world threat models. Our foremost consideration is achieving concurrent, adaptive security in non-interactive zero-knowledge proofs of knowledge (NIZKPoK) on the verge of adoption within larger cryptographic systems [50,27]. NIZKPoK are widely employed in group [17,8], blind [38], threshold [4], aggregate [41], and multi- [4,30] signatures, cryptographic shuffles [54,43] and accumulators [2,13], anonymous networks [28], credentials [12,14,15,51], e-cash [11], e-token [10], and voting [46,43,49], distributed ledgers [46], verifiable secret sharing [49] and encryption schemes [8,16], and in bootstrapping multi-party protocols secure in the "honest-but-curious" model (in which participants are guaranteed to act according to the protocol) into those secure in the more realistic "malicious" model (in which the adversary can adaptively corrupt and execute arbitrary code).

Achieving concurrent and adaptive security for Σ-protocol-based NIZKPoK deployed in the above systems is notoriously difficult [53,33,6,44]. Yet, Σ-protocols made non-interactive in the ROM are being embedded into real-world systems [27]. Constructing universally composable (UC) NIZKPoK that can withstand adaptive corruptions in the universal composition with global subroutines (UCGS) framework [19,1] (where the global subroutine is a random oracle) is therefore a natural and well-motivated problem.

Canetti, Sarkar, and Wang [23] developed a compiler that leverages UC non-interactive commitments and a correlation intractable hash function to obtain adaptive Σ-protocol-based UC NIZKPoK from standard assumptions. They consider Σ-protocols in the $\mathcal{F}_{\mathsf{NICOM}}$-CRS-hybrid model, where the first message of the Σ-protocol is transformed into a UC non-interactive commitment using equivocable commitments, CCA-2-secure public key encryption with oblivious ciphertext sampling, and a (non-programmable) common reference string (CRS). Non-interactivity and extractability are achieved using the correlation intractable hash function. These additional cryptographic building blocks are difficult to realize efficiently, so it is of theoretical interest only.

In this paper, we ask the question: can we achieve adaptively secure UC NIZKPoK in the global ROM from Σ-protocols efficiently? Specifically, existing constructions that are considered for standardization rely on Σ-protocols as a building block. Can we achieve a simple compiler that would make this building block into adaptively secure UC NIZKPoK in the global ROM, without paying a high price in efficiency?

Our main results. For Σ-protocols that can be described as linear function families [38], we show that our new adaptively-secure version of Kondi and shelat's randomized Fischlin transform [41,35] gives us such a compiler. Efficiency wise, our compiler inherits the same overhead as the standard version, thereby outperforming other known techniques (which we discuss further below).

In the programmable global ROM, we achieve adaptive UC NIZKPoK using our compiler alone; in the non-programmable global ROM, we modify Lysyanskaya and Rosenbloom's OR-protocol transformation [42] for the adaptive corruption setting. Along the way, we define formally the natural adaptivity property of Σ-protocols from linear function families, and also extend the definition to adaptive witness indistinguishability for OR-protocols. We show that proofs of knowledge of a discrete logarithm [48] and proofs of knowledge and equality of representations of a set of group element in a set of bases [18,16,32,50] are Σ-protocols from linear function families, and can therefore be efficiently transformed using our techniques into adaptive UC NIZKPoK in both global ROMs. Since the latter Σ-protocols are the main building block in numerous constructions in the area of privacy-preserving authentication (just to mention a few, see [14,15,7,3,51,39]), our main result will enable the realization of these constructions in a composable way in the adaptive corruption model.

Other Related Work. Prior work has shown that straight-line extraction is necessary to obtain UC NIZKPoK [42]. The overhead of the standard, randomized, and now our adaptive version of the Fischlin transform is—in terms of prover computation and communication complexity—polylogarithmic ($O(\texttt{polylog}(\lambda))$ in the security parameter λ. The cut-and-choose technique due to Pass [47] and adopted by Unruh [52] incurs an overhead of λ^2 (see Kondi and shelat [41, p. 3] for discussion), while the commit-and-open technique due to Camenisch and Damgård [8] and adopted by Canetti, Sarkar, and Wang [23] is in the CRS model rather than the ROM, and incurs an overhead of $O(\lambda)$.

Other proposals for adaptive UC NIZKPoK such as those based on the construction of Groth, Ostrovsky, and Sahai [37], are not optimized to transform Σ-protocols in the ROM. Rather, they are in the CRS model and involve representing the NP statement to be proven as a circuit C, encrypting (and more) each bit of the witness making C satisfiable, and doing a NIZK proof that the computations are correct with respect to the witness and some randomness. Groth, Ostrovsky, and Sahai's original construction incurred an overhead of $O(\lambda)$; more recent constructions such as the one due to Cohen, shelat, and Wichs [26] use new techniques to increase the efficiency, however the overhead is still substantial (in Cohen, shelat, and Wichs' construction, the proof size is $|w| \cdot \texttt{poly}(\lambda, d)$, where $|w|$ is the size of the witness and d is the circuit depth).

Garay, MacKenzie, and Yang [36] developed an efficient compiler for creating adaptive UC *interactive* ZK PoK from Σ-protocols. Lysyanskaya and Rosenbloom showed how to create UC NIZKPoK in the global ROM, but did not consider adaptive security [42]. Crites, Komlo, and Maller [31] achieved threshold signatures from Schnorr's Σ-protocol with strong notions of adaptive and concurrent security in the algebraic group model with random oracles; this work considers unforgeability and does not employ "full-strength" NIZKPoK.

Similarly, many of the protocols undergoing implementation and standardization focus on signatures from Σ-protocols in the ROM, and circumvent the need for NIZKPoK. Part of this seems to be because (as in the work of Crites, Komlo, and Maller) NIZKPoK are "overly strong" in some cases [31, p. 8]. However, the

difficulty of achieving strong security notions for NIZKPoK from Σ-protocols in the ROM has also been a significant limiting factor. In the FROST-Interactive protocol, for example, Komlo and Goldberg opted for one-time verifiable random functions "because of the difficulty of simulating [NIZKPoK] in a concurrent setting" [40, p.28]. Our work not only demonstrates that Σ-protocols can be transformed into universally composable, adaptively-secure NIZKPoK in the global ROM, but in such a way that is "plug-and-play" with existing infrastructure (i.e. by repeating various steps of widely-implemented Σ-protocols rather than by introducing new cryptographic primitives). We hope to set the stage for readily deployable, "full-strength" NIZKPoK for new cryptographic applications on the horizon. In the remainder of the introduction, we provide necessary high-level context for understanding our results.

Adaptive Σ-protocols. A standard Σ-protocol [32,50] is a three-move, public coin zero-knowledge proof system over a binary NP relation R, where statements x are proven to be in the language L_R using a "witness" w such that $(x, w) \in R$. V accepts or rejects the proof transcript $(x, \texttt{com}, \texttt{chl}, \texttt{res})$. This proof system must be *complete*, *special honest-verifier zero-knowledge* (SHVZK), and *special sound* (SS). Completeness says that if P forms a proof using the three-move form on input $(x, w) \in R$, then V accepts. SHVZK requires a *simulator* algorithm `SimProve` that, on input $(x, \texttt{chl})$ (but no witness) produces a proof transcript that looks indistinguishable from that of a "real" prover P. SS implies an *extractor* algorithm `Extract` that can compute a witness w from any two valid proofs $(\texttt{com}, \texttt{chl}, \texttt{res})$, $(\texttt{com}', \texttt{chl}', \texttt{res}')$ of a statement x with the same first message $\texttt{com} = \texttt{com}'$ but different verifier challenges $\texttt{chl} \neq \texttt{chl}'$. In an OR-type Σ-protocol [32,29], P proves that either it knows a witness w_0 to a statement x_0, or else it knows a witness w_1 to a statement x_1. The proof of the chosen witness w_b is computed according to the "real" prove algorithm of Σ_b, while the proof of $w_{\overline{b}}$ is computed according to the `SimProve` algorithm of $\Sigma_{\overline{b}}$. OR-protocols have a property called witness indistinguishability (WI), which states that the adversary cannot tell from seeing a proof whether the prover used w_b or $w_{\overline{b}}$.

In the adaptive corruption setting, the adversary can corrupt P after P has already issued proofs, revealing the entirety of P's random tape. We say that a Σ-protocol is adaptive SVHZK (adSHVZK) if it has an additional simulator algorithm, `SimRand`, that generates randomness that P could have used to produce a given (simulated) proof on input a given witness. Adaptive WI (adWI) additionally requires an algorithm `RealToSim`, which allows the simulator to "equivocate" the randomness used in a "real" proof of a witness w_b to make it look simulated (while `SimRand` is used on the simulated proof of $w_{\overline{b}}$ to make it look "real"). We show that popular Σ-protocols for proving discrete-logarithm-based relations are adaptive according to these definitions.

Universal Composition with Global Subroutines. The security experiment in the UC framework [19,1] tests a session of a cryptographic protocol Π in the presence of arbitrary concurrent protocols, which are modeled using an adversarial "environment" machine $\mathcal{Z}$. $\mathcal{Z}$ controls the corrupted protocol participants through an adversary machine, and can also send inputs to honest protocol participants

and observe their outputs. In the "real-world" security experiment, the honest parties form their outputs according to the protocol Π, and interact with a malicious adversary $\mathscr{A}$ that can be an arbitrary polynomial-time algorithm. In the "ideal-world" experiment, honest parties are just placeholder "dummies" who pass all of their inputs to an ideal functionality $\mathcal{F}$ (in our case, the ideal functionality $\mathcal{F}_{\texttt{aNIZK}}$ for adaptive NIZKPoK), which acts according to an "ideal" version of the protocol. The adversary $\mathscr{A}$ is replaced with an "ideal adversary" $\mathcal{S}$, who acts like $\mathscr{A}$ when talking to $\mathcal{Z}$, but also communicates with $\mathcal{F}$ and simulates the view of the corrupted parties. When $\mathcal{Z}$ wishes to adaptively corrupt a protocol participant P, $\mathcal{F}$ hands over any relevant information about P to $\mathcal{S}$, and all future communications with P are handled through $\mathcal{S}$. If $\mathcal{Z}$ cannot distinguish its interaction with "real" parties running Π in the presence of the "real" adversary $\mathscr{A}$ from its interaction with "ideal" (dummy) parties running $\mathcal{F}$ in the presence of the "ideal" adversary, then Π is said to "UC-realize" (or be UC-secure with respect to) $\mathcal{F}$ in the presence of $\mathcal{Z}$.

The universal composition with global subroutines (UCGS) model developed by Badertscher, Canetti, Hesse, Tackmann, and Zikas [1] accomodates a *global subroutine protocol* $\mathcal{G}$, which is wrapped into a joint body with the test protocol session π by a "shell" protocol $\texttt{M}$. The global subroutines of interest to our work are the restricted programmable observable global RO $\mathcal{G}_{\texttt{rpoRO}}$ of Camenisch, Drijvers, Gagliardoni, Lehmann, and Neven [9] and the restricted observable (non-programmable) global RO $\mathcal{G}_{\texttt{roRO}}$ of Canetti, Jain, and Scafuro [20].

Adaptive Straight-Line Compilers. A straight-line compiler (SLC) [42] is an algorithm $\texttt{SLC}$ that takes as input any Σ-protocol Σ_R for relation R (as described above) and produces as output a *non-interactive, straight-line extractable* (NISLE) proof system $\Pi_R^{\texttt{SLC}}$ for R in the ROM. A proof system is straight-line extractable if the challenger in the security experiment can obtain the two proofs needed to run the $\texttt{Extract}$ algorithm (and thereby compute a witness for the statement x) immediately after an adversarial prover issues a single proof, i.e. without any further interaction with the prover. In the ROM, the security experiment has access to the adversary's queries to the RO; therefore, as long as an adversarial prover is forced to query the RO on two proof transcripts with the same first message but different challenges, the $\texttt{Extract}$ algorithm can immediately compute a witness for the prover's statement. The randomized Fischlin transform [35,41] sets the proof repetition parameters such that the prover is guaranteed to issue two such transcripts to the RO.

We define an *adaptive* SLC as an SLC that preserves the adaptive security properties of the underlying Σ-protocol—that is, the resulting proof system $\Pi_R^{\texttt{SLC}}$ is also secure against adaptive corruptions. Specifically, we define new adaptive versions of the non-interactive multiple SHVZK (NIM-SHVZK) and special simulation-soundness (NI-SSS) properties guaranteed by the regular SLC, and show these properties (including completeness, Definitions 9-11) are both *necessary* and *sufficient* to obtain adaptive UC NIZKPoK in the global ROM.

Putting it all together. Consider the Σ-protocol for proving knowledge and equality of discrete logarithm representations in prime-order groups. It is used often in

applied cryptography constructions and was put forth for standardization [50]. Our results give a recipe for obtaining an adaptively secure UC NIZKPoK in the global ROM for equality of discrete logarithm representations, as follows: First, we show that this Σ-protocol is already an adaptively secure one, i.e. we give the algorithm `SimRand` (Section 3). Next, we show that compiling it with an adaptive SLC such as the adaptive randomized Fischlin transform (Section 4) yields an adaptively secure UC NIZKPoK in the programmable $\mathcal{G}_{\texttt{rpoRO}}$-hybrid model; we also show that in the non-programmable $\mathcal{G}_{\texttt{roRO}}$-$\mathcal{F}_{\texttt{CRS}}$-hybrid model, we obtain an adaptively secure UC NIZKPoK by compiling a slightly modified version of the Σ-protocol with an adaptive SLC (Section 5).

2 Preliminaries

In this section, we review the global RO, RO-CRS hybrid, and UCGS models. We will later demonstrate how to obtain adaptive UC NIZKPoK in the UCGS model with respect to both oracles. We use standard notation, available for review in Appendix A.1.

2.1 The Global Random Oracle Model(s)

We will demonstrate how to obtain adaptive UC NIZKPoK in two global random oracle models: the restricted *observable* global ROM of Canetti et al. [22] and the restricted *programmable* observable global ROM of Camenisch et al. [9]. The global ROs in both models have a traditional random function interface, `Query`, which takes an any-length string as input and returns a uniformly random ℓ-bit string as output [5]. The "global" designation refers to the fact that there exists a single instance of the oracle for all sessions of a protocol (as opposed to one functionality per protocol session, as in the standard UC model [19]). In order to be considered a global subroutine in the UCGS model [1], the global RO must be *subroutine respecting* and *regular* with respect to the challenge protocol. Both global ROs are subroutine respecting since the "extended instance" of each includes querying parties in any session, and no oracle subroutines interact directly with the querent. They are also regular with respect to the challenge protocol, since they neither invoke new NIZKPoK protocol participants nor run them as subroutines.

The observable-only RO $\mathcal{G}_{\texttt{roRO}}$ [22,9] records all "illegitimate" queries made for a session s by parties with an $\texttt{sid} \neq s$, which captures all of the environment's direct queries (since the environment is external to all legitimate protocol sessions by definition). The observability property is captured by the interface `Observe`, which takes a session s as input and produces a list of illegitimate queries $\mathcal{Q}_s$ for s as output. Since the only queries in $\mathcal{Q}_s$ are adversarial, we model $\mathcal{Q}_s$ as completely public—$\mathcal{G}_{\texttt{roRO}}$ can release it to anyone who asks [9].

The *programmable* version $\mathcal{G}_{\texttt{rpoRO}}$ [9] builds on the observable-only functionality with an `IsProgrammed` interface, which reveals the programmed entries

for a session s to any parties in the same session. Since the environment is external to all legitimate protocol sessions by definition, it will not be able to query the `IsProgrammed` interface directly, and must instead ask through the corrupted session participants (in the ideal world, the simulator can always return "false" to corrupted session participants' `IsProgrammed` queries). Unlike the "illegitimate queries" of the observation interface, the list of programmed queries cannot be made public to everyone (or the environment could trivially distinguish the real from ideal experiments). We include formalizations of both definitions in Appendix A.2.

2.2 The $\mathcal{G}_{\texttt{roRO}}$-$\mathcal{F}_{\texttt{CRS}}$-hybrid Model

In the observable-only $\mathcal{G}_{\texttt{roRO}}$-hybrid model, the `SimProve` algorithm has no additional power over a regular prover, since it cannot program $\mathcal{G}_{\texttt{roRO}}$. Thus, NIZKPoK in the plain $\mathcal{G}_{\texttt{roRO}}$-hybrid model are impossible [22,9,47,21,20]. To work around this impossibility result and construct UC NIZKPoK while avoiding the session-localized `IsProgrammed` interface of $\mathcal{G}_{\texttt{rpoRO}}$, Lysyanskaya and Rosenbloom introduce the $\mathcal{G}_{\texttt{roRO}}$-$\mathcal{F}_{\texttt{CRS}}$-hybrid model [42], where $\mathcal{F}_{\texttt{CRS}}$ is the (local) ideal common reference string (CRS) functionality.

In the real-world execution of the $\mathcal{G}_{\texttt{roRO}}$-$\mathcal{F}_{\texttt{CRS}}$-hybrid model, $\mathcal{F}_{\texttt{CRS}}$ has one interface, `Query`, that simply returns a consistent CRS to the participants of a particular session s. In the ideal-world experiment, the simulator plays the role of $\mathcal{F}_{\texttt{CRS}}$, and can generate $\texttt{CRS}_s$ for each session s with a secret trapdoor $\texttt{trap}_s$. Provers in the $\mathcal{G}_{\texttt{roRO}}$-$\mathcal{F}_{\texttt{CRS}}$-hybrid model prove via an OR-protocol [29,32] that *either* they know a real witness to a statement, *or* they know the trapdoor $\texttt{trap}_s$ to $\texttt{CRS}_s$. This allows the simulator in the ideal-world experiment to "simulate" proofs of statements without witnesses using its extra power—the CRS trapdoor—which a real-world prover will never have.

In order to make the CRS statement compatible with the definition of Σ-protocols, it must be drawn from a *samplable-hard relation* S—that is, generating the CRS must be efficient, and, given any $\texttt{CRS}_s \in L_S$, it must be overwhelmingly difficult to generate a $\texttt{trap}_s$ such that $S(\texttt{CRS}_s, \texttt{trap}_s) = 1$ [42]. The relation S must additionally be *Σ-friendly*: it must have a corresponding *efficient*—and in our case *adaptive* and *witness equivocable* (according to the definitions given in Section 3)—Σ-protocol Σ_S. Proofs in the $\mathcal{G}_{\texttt{roRO}}$-$\mathcal{F}_{\texttt{CRS}}$-hybrid model are therefore well-specified as OR-protocols $\Sigma_{R \vee S}$, where R is the relation of the original Σ-protocol Σ_R, and S is the relation underlying Σ_S. Formal definitions of $\mathcal{F}_{\texttt{CRS}}$ and context-friendly properties are given in Appendix A.3.

2.3 UC Security with Adaptive Corruptions

Informally, a protocol Π qualifies as adaptive UC with respect to an ideal functionality $\mathcal{F}$ (i.e. Π "aUC-emulates" $\mathcal{F}$) in the global ROM if for all efficient players in the security experiment, no adaptively-corrupting environment can distinguish the real-world experiment (with Π and $\mathscr{A}$) from the ideal-world experiment (with $\mathcal{F}_{\texttt{aNIZK}}$ and $\mathcal{S}$). In other words, adaptive UC is just standard

UC with an adaptive corruption flag set, but to maintain the clarity of our results we call any protocol that satisfies the UC security definition with adaptive corruptions aUC for short. We review the specifics of adaptive corruption in the UC experiment in Appendix A.4. Briefly, $\mathcal{Z}$ is allowed to issue a command $(\textbf{Corrupt}, P)$ during the protocol execution which reveals a full internal view of a party P.

In the UC with global subroutines (UCGS) model, Π UC-emulates $\mathcal{F}$ *in the presence of a global subroutine* $\mathcal{G}$ if $\mathtt{M}[\Pi, \mathcal{G}]$ UC-emulates $\mathtt{M}[\mathcal{F}, \mathcal{G}]$, where $\mathtt{M}$ is the "shell" wrapper mentioned in the introduction. Badertscher et al. [1] show that as long as the global subroutine is subroutine respecting and regular with respect to the challenge protocol, and that the challenge protocol is subroutine respecting *except in its interactions with* $\mathcal{G}$ (i.e. it is $\mathcal{G}$-subroutine respecting), then $\mathtt{M}[\Pi, \mathcal{G}]$ (resp. $\mathtt{M}[\mathcal{F}, \mathcal{G}]$) are subroutine respecting and behave just like Π and $\mathcal{G}$ (resp. $\mathcal{F}$ and $\mathcal{G}$) would behave as individual entities. We argued in earlier in this section that both $\mathcal{G}_{\mathtt{rpoRO}}$ and $\mathcal{G}_{\mathtt{roRO}}$ qualify as global subroutines, and as $\mathcal{G}_{\mathtt{rpoRO}}$ and $\mathcal{G}_{\mathtt{roRO}}$ are the only global subroutines in our experiments, Π and $\mathcal{F}_{\mathtt{aNIZK}}$ are $\mathcal{G}_{\mathtt{rpoRO}}$- and $\mathcal{G}_{\mathtt{roRO}}$-subroutine respecting. Therefore, without loss of generality, we consider our UC experiments "in the presence of global subroutines" $\mathcal{G}_{\mathtt{rpoRO}}$ and $\mathcal{G}_{\mathtt{roRO}}$—or in the $\mathcal{G}_{\mathtt{rpoRO}}$- and $\mathcal{G}_{\mathtt{roRO}}$-hybrid models—and omit the notation $\mathtt{M}$.

We review the standard UC-security definition with respect to a generic global subroutine $\mathcal{G}$, and instantiate the individual versions (i.e. with the generic global RO $\mathcal{G}_{\mathtt{RO}}$, $\mathcal{G}_{\mathtt{rpoRO}}$, $\mathcal{G}_{\mathtt{roRO}}$, and $\mathcal{F}_{\mathtt{CRS}}$) as needed throughout the paper.

Definition 1 (aUC Protocols in the $\mathcal{G}$-hybrid Model). *A protocol Π with security parameter λ* aUC-realizes the ideal functionality $\mathcal{F}$ with adaptive corruptions in the $\mathcal{G}$-hybrid model *if for all efficient $\mathscr{A}$, there exists an ideal adversary $\mathcal{S}$ efficient in expectation such that for all efficient environments $\mathcal{Z}$ that can issue adaptive corruptions,* $\mathsf{IDEAL}^{\mathcal{G}}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(1^\lambda, \mathtt{aux}) \approx_c \mathsf{REAL}^{\mathcal{G},*}_{\Pi,\mathscr{A},\mathcal{Z}}(1^\lambda, \mathtt{aux})$, *where $\mathcal{G}$ is a global subroutine,* $\mathtt{aux}$ *is any auxiliary information provided to the environment, and $*$ represents any additional local functionality included in the real-world experiment.*

3 Adaptive Σ-protocols and Friends

In this section, we formalize notions of adaptive Σ-protocols, OR-protocols, and adaptive witness indistinguishability. We then identify a common class of Σ-protocols and show they are adaptive according to our definition.

3.1 Adaptive Σ-protocols

Recall from the introduction that a Σ-protocol for a binary NP relation R is a three-move public-coin protocol between a prover P and a verifier V, after which V is convinced that P has a witness w for some statement x such that $R(x, w) = 1$. P is assumed to be a *probabilistic* (polynomial-time) Turing Machine (TM)—that is, P is assumed to have a random tape, from which it can

sample (polynomially-many) random bits. We make this random tape explicit, denoting it as a random string r of length λ_r.

The basic three-move form of a Σ-protocol is generally described as an interactive "protocol template" τ [32,42]. The algorithmic version of the protocol template definition consists of the following algorithms: τ.`Setup`, τ.`Commit`, τ.`Challenge`, τ.`Respond`, and τ.`Decision` [42]. We modify the τ.`Setup` algorithm such that the public parameters contain the prover's *randomness security parameter* λ_P, derived from the overall security parameter λ, that specifies the amount of randomness necessary to compute the first message `com`. In particular, we assume the randomness r that is given as input to τ.`Commit` is sampled uniformly at random from $\{0,1\}^{\lambda_P}$, representing a λ_P-length section of P's random tape. The algorithms τ.`Challenge`, τ.`Respond`, and τ.`Decision` are unchanged. We include a formal description of the adaptive Σ-protocol template in Appendix A.5.

In addition to the protocol template, Σ-protocols are defined with respect to three fundamental properties: completeness, special honest-verifier zero-knowledge (SHVZK), and special soundness (SS). In the adaptive corruption setting, the zero-knowledge simulator must additionally be able to produce a view of the prover's randomness that is consistent with the proofs generated by the `SimProve` algorithm. We therefore introduce a new algorithm—`SimRand`—which, given its own section of random tape $r_{\texttt{Sim}}$ of size λ_S, a statement, a proof, some auxiliary information produced by `SimSetup` and `SimProve`, and *the witness that was supposedly used to compute the proof*, outputs some simulated randomness r. This algorithm captures the intuition that an adaptive Σ-protocol simulator must be able to generate convincing randomness for the prover's random tape *after the proof has already been generated* and the prover is corrupted.

Definition 2 (Adaptive Σ-protocol). *An* adaptive Σ-protocol for a relation R *based on adaptive protocol template* τ *(Appendix A.5) is a tuple of efficient procedures* $\Sigma_{R,\tau} = (\texttt{Setup}, \texttt{Prove}, \texttt{Verify}, \texttt{SimSetup}, \texttt{SimProve}, \texttt{SimRand}, \texttt{Extract})$*, defined as follows.*

- $\texttt{ppm} \leftarrow \texttt{Setup}(1^\lambda)$*: Given a security parameter* 1^λ*, invoke* τ.`Setup`(1^λ) *to obtain the public parameters* `ppm`*, which include randomness security parameters* λ_P *and* $\lambda_{\texttt{Sim}}$ *for the prover and simulator, respectively.*
- $\pi \leftarrow \texttt{Prove}((\texttt{ppm}, x, w, r), (\texttt{ppm}, x, \texttt{chl}))$*: Let the first (resp. second) argument to* `Prove` *be the input to* P *(resp.* V*), where both parties get* `ppm` *and the statement* x*, but only* P *gets witness* w *and randomness* $r \leftarrow_\$ \{0,1\}^{\lambda_P}$*.* P *and* V *run* τ.`Commit`*,* τ.`Challenge`*, and* τ.`Respond`*. Output* $\pi = (\texttt{com}, \texttt{chl}, \texttt{res})$*.*
- $\{0,1\} \leftarrow \texttt{Verify}(\texttt{ppm}, x, \pi)$*: Given a proof* π *for statement* x*, parse* π *as* $(\texttt{com}, \texttt{chl}, \texttt{res})$ *and output the result of running* τ.`Decision` *on input* $(x, \texttt{com}, \texttt{chl}, \texttt{res})$*.* `Verify` *must satisfy the* completeness property *(Appendix A.6).*
- $(\texttt{ppm}, z_{\texttt{Sim}}) \leftarrow \texttt{SimSetup}(1^\lambda)$*: Generate* `ppm` *and simulator state information* $z_{\texttt{Sim}}$*. Together,* `SimSetup`*,* `SimProve`*, and* `SimRand` *must satisfy the* adaptive special honest-verifier zero-knowledge property *(Appendix A.6).*

- $(\pi, z_\pi) \leftarrow \mathtt{SimProve}(\mathtt{ppm}, z_{\mathtt{Sim}}, x, \mathtt{chl}, r_{\mathtt{Sim}})$: *Given public parameters* $\mathtt{ppm}$*, state* $z_{\mathtt{Sim}}$*, statement* x*, challenge* $\mathtt{chl}$*, and randomness* $r_{\mathtt{Sim}} \leftarrow \{0,1\}^{\lambda_{\mathtt{Sim}}}$*, produce a proof* $\pi = (\mathtt{com}, \mathtt{chl}, \mathtt{res})$ *and proof state information* z_π*.*
- $r \leftarrow \mathtt{SimRand}(\mathtt{ppm}, z_{\mathtt{Sim}}, z_\pi, x, \pi, w, r_{\mathtt{Sim}})$: *Given public parameters* $\mathtt{ppm}$*, simulator state* $z_{\mathtt{Sim}}$*, proof state* z_π*, proof* π *for statement* x*, a witness* w *such that* $R(x, w) = 1$*, and randomness* $r_{\mathtt{Sim}}$*, produce randomness* r*.*
- $w \leftarrow \mathtt{Extract}(\mathtt{ppm}, x, \pi, \pi')$: *Given two proofs* $\pi = (\mathtt{com}, \mathtt{chl}, \mathtt{res})$ *and* $\pi' = (\mathtt{com}, \mathtt{chl}', \mathtt{res}')$ *for a statement* x*, output a witness* w*.* $\mathtt{Extract}$ *must satisfy the* two-special soundness property *(Appendix A.6).*

For convenience and when the meaning is clear, we use Σ_R *to represent* $\Sigma_{R,\tau}$ *and omit* $\mathtt{ppm}$ *from the input of the algorithms.*

Standard definitions of completeness and two-special soundness are given in Appendix A.6; our new definition of adaptive SHVZK is given below.

Definition 3 (Adaptive Special Honest-Verifier Zero-Knowledge). *A* Σ*-protocol* Σ_R *for relation* R *is statistical (resp. computational)* adaptive special honest-verifier zero-knowledge (adaptive SHVZK) *if there exist algorithms* $\mathtt{SimSetup}$*,* $\mathtt{SimProve}$*, and* $\mathtt{SimRand}$ *such that for any security parameter* λ*, any adversary (resp. any* PPT *adversary)* $\mathscr{A}$*, and a bit* $b \leftarrow_\$ \{0,1\}$*, there exists some negligible function* negl *such that* $\Pr[b' = b] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$*, where* b' *is the result of running the game* $\mathsf{adSHVZK}_{\mathscr{A},\Sigma_R}(1^\lambda, b)$ *from Figure 1. We say* $\mathscr{A}$ *wins the* $\mathsf{adSHVZK}$ *game if* $\Pr[b' = b] > \frac{1}{2} + \mathsf{negl}(\lambda)$*.*

$\mathsf{adSHVZK}_{\mathscr{A},\Sigma_R}(1^\lambda, 0)$: Real	$\mathsf{adSHVZK}_{\mathscr{A},\Sigma_R}(1^\lambda, 1)$: Ideal
$\mathtt{ppm} \leftarrow \Sigma_R.\mathtt{Setup}(1^\lambda)$	$(\mathtt{ppm}, z_{\mathtt{Sim}}) \leftarrow \Sigma_R.\mathtt{SimSetup}(1^\lambda)$
$(x, w, \mathtt{st}) \leftarrow \mathscr{A}(1^\lambda, \mathtt{ppm})$	$(x, w, \mathtt{st}) \leftarrow \mathscr{A}(1^\lambda, \mathtt{ppm})$
if $R(x, w) = 1$:	**if** $R(x, w) = 1$:
$\quad r \leftarrow_\$ \{0,1\}^{\lambda_P}$	$\quad r_{\mathtt{Sim}} \leftarrow_\$ \{0,1\}^{\lambda_{\mathtt{Sim}}}$
$\quad \mathtt{chl} \leftarrow \{0,1\}^\ell$	$\quad \mathtt{chl} \leftarrow \{0,1\}^\ell$
$\quad \pi \leftarrow \Sigma_R.\mathtt{Prove}((x, w, r), (x, \mathtt{chl}))$	$\quad (\pi, z_\pi) \leftarrow \Sigma_R.\mathtt{SimProve}(z_{\mathtt{Sim}}, x, \mathtt{chl}, r_{\mathtt{Sim}})$
	$\quad r \leftarrow \Sigma_R.\mathtt{SimRand}(z_{\mathtt{Sim}}, z_\pi, x, \pi, w, r_{\mathtt{Sim}})$
else :	**else** :
$\quad \pi \leftarrow \perp$	$\quad \pi \leftarrow \perp$
$b' \leftarrow \mathscr{A}(\mathtt{st}, \pi, r)$	$b' \leftarrow \mathscr{A}(\mathtt{st}, \pi, r)$
return b'	**return** b'

Fig. 1. Adaptive Special Honest-Verifier Zero-Knowledge (adSHVZK) Game.

3.2 OR-protocols

An OR-protocol [32,42] is a Σ-protocol over a disjunctive relation: the prover shows that *either* it knows a witness w_0 for a statement x_0 such that $R_0(x_0, w_0) = 1$, *or* it knows a witness w_1 for a statement x_1 such that $R_1(x_1, w_1) = 1$. The structure of the proof produced by an OR-protocol prover is $\Phi = (\pi_0, \pi_1, \texttt{CHL})$, where one of the proofs π_b is computed according to the $\Sigma_b.\texttt{Prove}$ algorithm and the other is according to the $\Sigma_{\overline{b}}.\texttt{SimProve}$ algorithm.

Recall from the introduction that an OR-protocol Σ_{OR} must have a property called *witness indistinguishability*, which (in the non-adaptive setting) follows directly from the special honest-verifier zero-knowledge property of Σ_0 and Σ_1 [32]. We introduce an *adaptive* OR-protocol formalization, which we will need for the proof of Theorem 8. Since $\Sigma_{OR}.\texttt{SimProve}$ computes both halves of the proof according to $\Sigma_b.\texttt{SimProve}$ and $\Sigma_{\overline{b}}.\ \texttt{SimProve}$, respectively, in order to make the simulated random tape look "real," $\Sigma_{OR}.\texttt{SimRand}$ must produce real-looking randomness for the proof π_b corresponding to the witness w_b that was supposedly used to run $\Sigma_{OR}.\texttt{Prove}$.

This is not enough, however: we know from Damgård that the zero knowledge property for Σ-protocols implies the witness indistinguishability property for OR-type Σ-protocols [32]. Informally, our *adaptive* witness indistinguishability property states that $\mathscr{A}$ cannot tell whether a prover used w_b or $w_{\overline{b}}$, even upon learning either the prover's true random tape, or an alternative random tape that the prover might have had if it were using the other witness. Conversely, if our adaptive Σ-protocols do not afford witness indistinguishability when composed in an OR-protocol, they cannot satisfy zero knowledge. Therefore, we introduce an algorithm to capture the idea that the simulator, after issuing a *real proof* for the witness w_b, must be able to reveal its random coins as if it used $w_{\overline{b}}$.

Let $\texttt{RealToSim}$ be an algorithm that takes as input a proof π for statement x computed according to $\Sigma.\texttt{Prove}$ using witness w and randomness r, and outputs simulated random coins $r_{\texttt{Sim}}$. In order to express that the simulator is able to effectively equivocate the prover's random tape in either direction (such that the proof could have been computed according to w_b or $w_{\overline{b}}$), we say that a Σ-protocol is *randomness equivocable* if there does not exist an $\mathscr{A}$ which can distinguish a simulated proof accompanied by the simulator's actual random tape $r_{\texttt{Sim}}$ and a real proof accompanied by the output of $\texttt{RealToSim}$ with non-negligible advantage. For clarity we keep the $\texttt{SimRand}$ and $\texttt{RealToSim}$ algorithms separate for OR-protocols, but intuitively if one considers the OR-protocol as any other Σ-protocol, one can think of wrapping both of these algorithms together into an overarching $\Sigma_{OR}.\texttt{SimRand}'$ algorithm (as specified in Definition 2) which produces the contents of the prover's random tape given either scenario (i.e. making a fully-simulated proof look real, or equivocating the witness used).

Definition 4 (Randomness Equivocability). *A Σ-protocol Σ_R for relation R is* randomness equivocable (RE) *if there exist algorithms* $\texttt{SimSetup}$, $\texttt{SimProve}$, $\texttt{SimRand}$, *and* $\texttt{RealToSim}$ *such that Σ_R is adaptive special honest-verifier zero-knowledge (Definition 3) and for any security parameter λ, any adversary (resp.*

any PPT *adversary)* $\mathscr{A}$*, and a bit* $b \leftarrow_{\$} \{0,1\}$*, there exists some negligible function* negl *such that* $\Pr[b' = b] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$*, where* b' *is the result of running the game* $\mathsf{RE}_{\mathscr{A},\Sigma_R}(1^\lambda, b)$ *from Figure 2. We say* $\mathscr{A}$ *wins the* RE *game if* $\Pr[b' = b] > \frac{1}{2} + \mathsf{negl}(\lambda)$.

$\mathsf{RE}_{\mathscr{A},\Sigma_R}(1^\lambda, 0)$	$\mathsf{RE}_{\mathscr{A},\Sigma_R}(1^\lambda, 1)$
$(\mathtt{ppm}, z_{\mathtt{Sim}}) \leftarrow \Sigma_R.\mathtt{SimSetup}(1^\lambda)$	$(\mathtt{ppm}, z_{\mathtt{Sim}}) \leftarrow \Sigma_R.\mathtt{SimSetup}(1^\lambda)$
$(\mathtt{Prove}, x, w, \mathtt{st}) \leftarrow \mathscr{A}(1^\lambda, \mathtt{ppm})$	$(\mathtt{Prove}, x, w, \mathtt{st}) \leftarrow \mathscr{A}(1^\lambda, \mathtt{ppm})$
if $R(x, w) = 1$:	**if** $R(x, w) = 1$:
$\quad r \leftarrow_{\$} \{0,1\}^{\lambda_P}$	$\quad r_{\mathtt{Sim}} \leftarrow \{0,1\}^{\lambda_{\mathtt{Sim}}}$
$\quad \mathtt{chl} \leftarrow \{0,1\}^{\ell}$	$\quad \mathtt{chl} \leftarrow \{0,1\}^{\ell}$
$\quad \pi \leftarrow \Sigma_R.\mathtt{Prove}((x, w, r), (x, \mathtt{chl}))$	$\quad (\pi, z_\pi) \leftarrow \Sigma_R.\mathtt{SimProve}(z_{\mathtt{Sim}}, x, \mathtt{chl}, r_{\mathtt{Sim}})$
$\quad r_{\mathtt{Sim}} \leftarrow \Sigma_R.\mathtt{RealToSim}(\mathtt{ppm}, z_{\mathtt{Sim}}, x, \pi, w, r)$	
else :	**else** :
$\quad \pi \leftarrow \perp$	$\quad \pi \leftarrow \perp$
$b' \leftarrow \mathscr{A}(\mathtt{st}, \pi, r_{\mathtt{Sim}})$	$b' \leftarrow \mathscr{A}(\mathtt{st}, \pi, r_{\mathtt{Sim}})$
return b'	**return** b'

Fig. 2. Randomness Equivocability (RE) Game.

When plugged into an OR-protocol, the `SimRand` and `RealToSim` algorithms allow the simulator to equivocate the witness used to compute a real proof from w_b to $w_{\bar{b}}$ by fabricating a "real" prover's random tape for the previously simulated proof $\pi_{\bar{b}}$ (according to $\Sigma_{\bar{b}}.\mathtt{SimRand}$) *and* the simulator's random tape for the previously "real" proof π_b (according to $\Sigma_b.\mathtt{RealToSim}$). We call this OR-protocol algorithm `EquivRand`.

Definition 5 (Adaptive OR-Protocol). *An* adaptive OR-protocol *for a relation* $R_{OR} = R_0 \vee R_1$ *based on adaptive* Σ*-protocols* Σ_{R_0,τ_0} *and* Σ_{R_1,τ_1} *(Definition 2) is a tuple of procedures* $\Sigma_{OR} = (\mathtt{Setup}, \mathtt{Prove}, \mathtt{Verify}, \mathtt{SimSetup}, \mathtt{SimProve}, \mathtt{SimRand}, \mathtt{Extract})$*, defined as follows.*

- $\mathtt{PPM} \leftarrow \mathtt{Setup}(1^\lambda)$*: Given a security parameter* 1^λ*, run* $\Sigma_{R_0}.\mathtt{Setup}(1^\lambda)$ *to obtain* $\mathtt{ppm}_0$ *and* $\Sigma_{R_1}.\mathtt{Setup}(1^\lambda)$ *to obtain* $\mathtt{ppm}_1$*. Let* $\lambda_P = \lambda_{\mathtt{Sim}} = \lambda_{r_0} + \lambda_{\mathtt{Sim}_0} + \lambda_{r_1} + \lambda_{\mathtt{Sim}_1}$*. Output* $\mathtt{PPM} = (\mathtt{ppm}_0, \mathtt{ppm}_1, \lambda_P, \lambda_{\mathtt{Sim}})$.
- $\Phi \leftarrow \mathtt{Prove}((\mathtt{PPM}, X, W, T), (\mathtt{PPM}, X, \mathtt{CHL}))$*: Parse* $X = (x_0, x_1)$*,* $W = (w, b)$*, and* $T = (r_b, r_{\mathtt{Sim}_b}, r_{\bar{b}}, r_{\mathtt{Sim}_{\bar{b}}})$*, and let* b *be the bit such that* $(x_b, w) \in R_b$*.* P *computes* $\mathtt{com}_b$ *according to* $\tau_b.\mathtt{Commit}(x_b, w, r_b)$*.* P *chooses* $\mathtt{chl}_{\bar{b}}$ *at random and generates* $(\mathtt{com}_{\bar{b}}, \mathtt{chl}_{\bar{b}}, \mathtt{res}_{\bar{b}})$ *by running* $\Sigma_{R_{\bar{b}}}.\mathtt{SimProve}(x_{\bar{b}}, z_{\bar{b}}, \mathtt{chl}_{\bar{b}}, r_{\mathtt{Sim}_{\bar{b}}})$*.* P *sends* V $\mathtt{Com} = (\mathtt{com}_0, \mathtt{com}_1)$ *and* V *sends* P *a random* ℓ*-bit string* $\mathtt{CHL}$*.* P

sets $\mathtt{chl}_b = \mathtt{CHL} \oplus \mathtt{chl}_{\bar{b}}$ *and computes* $\mathtt{res}_b$ *according to* $\tau_b.\mathtt{Respond}(x_b, w, \mathtt{com}_b, \mathtt{chl}_b)$. *$P$ sends* $(\mathtt{Chl}, \mathtt{Res}) = (\mathtt{chl}_0, \mathtt{chl}_1, \mathtt{res}_0, \mathtt{res}_1)$ *to V.*

- $\{0,1\} \leftarrow \mathtt{Verify}(\mathtt{PPM}, X, \Phi)$*: Parse* Φ *as* $(\pi_0, \pi_1, \mathtt{CHL})$*, where* $\pi_b = (\mathtt{com}_b, \mathtt{chl}_b, \mathtt{res}_b)$ *for* $b \in \{0,1\}$*. If* $\tau_0.\mathtt{Decision}(x_0, \mathtt{com}_0, \mathtt{chl}_0, \mathtt{res}_0) = \tau_1.\mathtt{Decision}(x_1, \mathtt{com}_1, \mathtt{chl}_1, \mathtt{res}_1) = 1$ *and* $\mathtt{chl}_0 \oplus \mathtt{chl}_1 = \mathtt{CHL}$*, return 1 (accept). Otherwise, return 0 (reject).*
- $(\mathtt{PPM}, Z_{\mathtt{Sim}}) \leftarrow \mathtt{SimSetup}(1^\lambda)$*: Compute* $(\mathtt{ppm}_0, z_{\mathtt{Sim}_0}) \leftarrow \Sigma_{R_0}.\mathtt{SimSetup}(1^\lambda)$ *and* $(\mathtt{ppm}_1, z_{\mathtt{Sim}_1}) \leftarrow \Sigma_{R_1}.\mathtt{SimSetup}(1^\lambda)$*. Return* $(\mathtt{PPM}, Z)$ *where* $\mathtt{PPM} = (\mathtt{ppm}_0, \mathtt{ppm}_1)$ *and* $Z_{\mathtt{Sim}} = (z_{\mathtt{Sim}_0}, z_{\mathtt{Sim}_1})$.
- $(\Phi, Z_\Phi) \leftarrow \mathtt{SimProve}(\mathtt{PPM}, X, Z_{\mathtt{Sim}}, \mathtt{CHL}, T_{\mathtt{Sim}})$: *Parse* $\mathtt{PPM} = (\mathtt{ppm}_0, \mathtt{ppm}_1)$*,* $X = (x_0, x_1)$*,* $Z_{\mathtt{Sim}} = (z_{\mathtt{Sim}_0}, z_{\mathtt{Sim}_1})$*, and* $T_{\mathtt{Sim}} = (r_{\mathtt{Sim}_0}, r_{\mathtt{Sim}_1})$*. Generate* $\mathtt{chl}_0$ *uniformly at random and set* $\mathtt{chl}_1 = \mathtt{chl}_0 \oplus \mathtt{CHL}$*. Compute* $(\pi_0, z_{\pi_0}) \leftarrow \Sigma_{R_0}.\mathtt{SimProve}(x_0, \mathtt{chl}_0, r_{\mathtt{Sim}_0})$ *and* $(\pi_1, z_{\pi_1}) \leftarrow \Sigma_{R_1}.\mathtt{SimProve}(x_1, \mathtt{chl}_1, r_{\mathtt{Sim}_1})$*. Return* (Φ, Z_Φ) *where* $\Phi = (\pi_0, \pi_1, \mathtt{CHL})$ *and* $Z_\Phi = (z_{\pi_0}, z_{\pi_1})$.
- $T' \leftarrow \mathtt{SimRand}(\mathtt{PPM}, Z_{\mathtt{Sim}}, X, \Phi, Z_\Phi, W, T_{\mathtt{Sim}})$: *Parse* $\mathtt{PPM} = (\mathtt{ppm}_0, \mathtt{ppm}_1)$*,* $Z_{\mathtt{Sim}} = (z_{\mathtt{Sim}_0}, z_{\mathtt{Sim}_1})$*,* $X = (x_0, x_1)$*,* $\Phi = (\pi_0, \pi_1, \mathtt{CHL})$*,* $Z_\Phi = (z_{\pi_0}, z_{\pi_1})$*,* $W = (w, b)$*, and* $T_{\mathtt{Sim}} = (r_{\mathtt{Sim}_0}, r_{\mathtt{Sim}_1})$*. Compute* $r_b \leftarrow \Sigma_{R_b}.\mathtt{SimRand}(\mathtt{ppm}_b, z_{\mathtt{Sim}_b}, x_b, z_{\pi_b}, w, r_{\mathtt{Sim}_b})$ *and let* $r_{\bar{b}} = r_{\mathtt{Sim}_{\bar{b}}}$*. Return* $T = (r_0, r_1)$.
- $T^* \leftarrow \mathtt{EquivRand}(\mathtt{PPM}, Z_{\mathtt{Sim}}, X, \Phi, Z_\Phi, W, T)$: *Parse* $\mathtt{PPM} = (\mathtt{ppm}_0, \mathtt{ppm}_1)$*,* $Z_{\mathtt{Sim}} = (z_{\mathtt{Sim}_0}, z_{\mathtt{Sim}_1})$*,* $X = (x_0, x_1)$*,* $\Phi = (\pi_0, \pi_1, \mathtt{CHL})$*,* $Z_\Phi = (z_{\pi_0}, z_{\pi_1})$*,* $W = (w, b)$*, and* $T = (r_b, r_{\mathtt{Sim}_b}, r_{\bar{b}}, r_{\mathtt{Sim}_{\bar{b}}})$*. Compute* $r'_{\bar{b}} \leftarrow \Sigma_{R_b}.\mathtt{RealToSim}(\mathtt{ppm}_b, x_b, \pi_b, w, r_b)$ *and* $r'_b \leftarrow \Sigma_{R_{\bar{b}}}.\mathtt{SimRand}(\mathtt{ppm}_b, z_{\mathtt{Sim}_{\bar{b}}}, x_b, \pi_b, z_{\pi_{\bar{b}}}, w, r_{\mathtt{Sim}_{\bar{b}}})$*. Return* $T = (r'_0, r'_1)$.
- $W \leftarrow \mathtt{Extract}(\mathtt{PPM}, X, \Phi, \Phi')$*: Parse* $X = (x_0, x_1)$*,* $\Phi = (\pi_0, \pi_1)$*, and* $\Phi' = (\pi'_0, \pi'_1)$*. Compute* $w_0 \leftarrow \Sigma_{R_0}.\mathtt{Extract}(x_0, \pi_0, \pi'_0)$ *and* $w_1 \leftarrow \Sigma_{R_1}.\mathtt{Extract}(x_1, \pi_1, \pi'_1)$*. Return* $W = (w_0, w_1)$.

3.3 Adaptive Witness Indistinguishability

Finally, we formalize the definition of adaptive witness indistinguishability for OR-protocols and prove that as long as the underlying Σ-protocols Σ_0 and Σ_1 are both adaptive special honest-verifier zero-knowledge *and* randomness equivocable, the OR-protocol based on Σ_0 and Σ_1 is adaptive witness indistinguishable.

Definition 6 (Adaptive Witness Indistinguishability). *An adaptive OR-protocol* $\Sigma_{OR} = (\mathtt{Setup}, \mathtt{Prove}, \mathtt{Verify}, \mathtt{SimSetup}, \mathtt{SimProve}, \mathtt{SimRand}, \mathtt{Extract})$ *(Definition 5) based on two adaptive Σ-protocols Σ_{R_0} and Σ_{R_1} (Definition 2) is* adaptive witness indistinguishable (adWI) *if there exists an algorithm* EquivRand *such that for all security parameters* 1^λ*, all PPT $\mathscr{A}$, and a bit* $b \leftarrow_\$ \{0,1\}$*, there exists some negligible function* negl *such that* $\Pr[b' = b] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$*, where b' is the result of running the game* $\mathsf{adWI}_{\mathscr{A}, \Sigma_{OR}}(1^\lambda, b)$ *from Figure 3. We say $\mathscr{A}$ wins the* adWI *game if* $\Pr[b' = b] > \frac{1}{2} + \mathsf{negl}(\lambda)$.

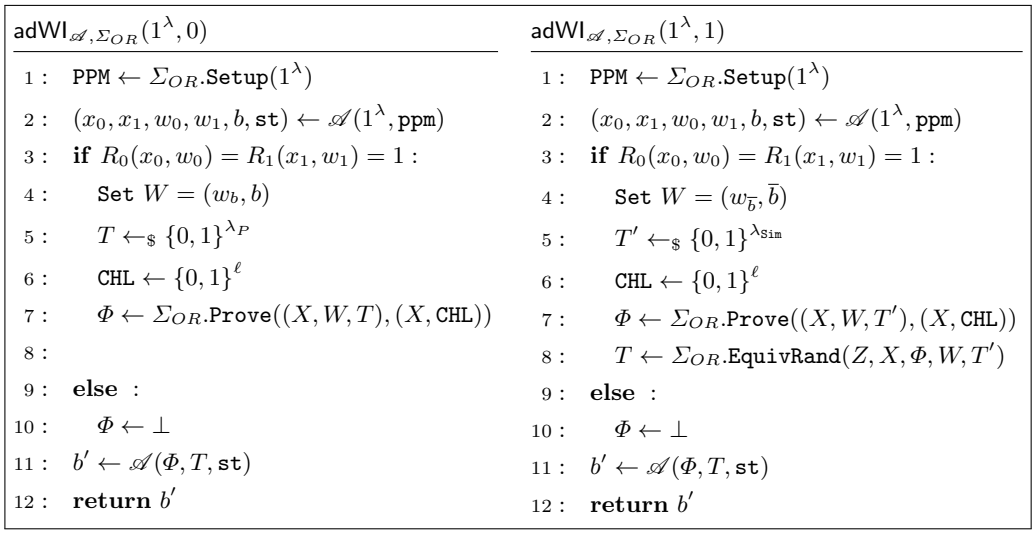


Fig. 3. Adaptive Witness Indistinguishability (adWI) Game.

Theorem 1 (adSHVZK and RE Imply Adaptive WI). *Let Σ_{OR} be an OR-protocol (Definition 5) based on two Σ-protocols Σ_{R_0,τ_0} and Σ_{R_1,τ_1} (Definition 2). If Σ_{OR} is adaptive special honest-verifier zero-knowledge (Definition 3) and randomness equivocable (Definition 4), then Σ_{OR} is adaptive witness indistinguishable (Definition 6).*

Proof. We wish to show that as long as Σ_{OR} is adaptive special honest-verifier zero-knowledge (adSHVZK) and randomness equivocable (RE), $\mathscr{A}$'s advantage in distinguishing the adaptive witness indistinguishability (adWI) experiments is negligible. Without loss of generality, we consider equivalent formulations of the definitions in which the statement $\Pr[b' = b] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$ is replaced with $|\Pr[b' = 0 \mid b = 0] - \Pr[b' = 0 \mid b = 1]| \leq \mathsf{negl}(\lambda)$, where b' represents $\mathscr{A}$'s output in either the $b = 0$ or $b = 1$ experiments.

Let the 0-bit adWI experiment be denoted RS for "real-sim," since the adversary's chosen witness w_b is used as input to $\Sigma_{OR}.\mathtt{Prove}$, i.e. π_b and r_b (resp. $\pi_{\overline{b}}$ and $r_{\overline{b}}$) are computed according to $\Sigma_b.\mathtt{Prove}$ and the prover's randomness(resp. $\Sigma_{\overline{b}}.\mathtt{SimProve}$ and the simulator's randomness). Let the 1-bit adWI experiment be denoted SR for the above scenario's complement, "sim-real," where the only change is that the random tape is equivocated (according to $\Sigma_b.\mathtt{SimRand}$ and $\Sigma_{\overline{b}}.\mathtt{RealToSim}$). We fix $\mathscr{A}$'s probability of outputting 0 in the RS (resp. SR) experiment to be $p^{\mathscr{A}}_{\mathtt{RS}}(\lambda)$ (resp. $p^{\mathscr{A}}_{\mathtt{SR}}(\lambda)$). We will show via a hybrid experiment SS, or "sim-sim"—in which both halves of the proof are according to $\mathtt{SimProve}$—that for all $\mathscr{A}$, as long as Σ_{OR} is adSHVZK and RE, $|p^{\mathscr{A}}_{\mathtt{RS}}(\lambda) - p^{\mathscr{A}}_{\mathtt{SR}}(\lambda)| \leq \mathsf{negl}(\lambda)$.

In the proof of the first lemma, we show that, as long as both Σ_0 and Σ_1 are adSHVZK, $|p^{\mathscr{A}}_{\mathtt{RS}}(\lambda) - p^{\mathscr{A}}_{\mathtt{SS}}(\lambda)| \leq \mathsf{negl}(\lambda)$. Our reduction proceeds by using an $\mathscr{A}$ such that $|p^{\mathscr{A}}_{\mathtt{RS}}(\lambda) - p^{\mathscr{A}}_{\mathtt{SS}}(\lambda)| > \mathsf{negl}(\lambda)$ as a black-box, forwarding all queries to the

adSHVZK challenger for proofs π_b of $\mathscr{A}$'s chosen Σ_b and computing the other half $\pi_{\overline{b}}$ using $\Sigma_{\overline{b}}.\mathtt{SimProve}$. In the second lemma we use a similar $\mathscr{A}$ and proof strategy, this time to show that, as long as both Σ_0 and Σ_1 are randomness equivocable, $|p^{\mathscr{A}}_{\mathtt{RS}}(\lambda) - p^{\mathscr{A}}_{\mathtt{SS}}(\lambda)| \leq \mathsf{negl}(\lambda)$. By these two lemmas and the triangle inequality, we conclude that $|p^{\mathscr{A}}_{\mathtt{RS}}(\lambda) - p^{\mathscr{A}}_{\mathtt{SR}}(\lambda)| = |(p^{\mathscr{A}}_{\mathtt{RS}}(\lambda) - p^{\mathscr{A}}_{\mathtt{SS}}(\lambda)) + (p^{\mathscr{A}}_{\mathtt{SS}}(\lambda) - p^{\mathscr{A}}_{\mathtt{SR}}(\lambda))| \leq \mathsf{negl}(\lambda)$. The full details of the proof are available in Appendix A.7. □

3.4 Simple Adaptive Σ-protocol Abstraction

We use the abstraction of Hauck, Kiltz, and Loss [38], and show how it captures simple Σ-protocols. We then give two instantiations: proofs of knowledge of a discrete logarithm and n equivalent representations of m witnesses. As proven in Section 5, any Σ-protocol that qualifies as adaptive can be efficiently bootstrapped into an adaptive UC NIZKPoK in both the programmable and non-programmable global ROMs.

Hauck et al.'s construction of canonical identification schemes from pseudo-modules and linear function families [38] describes an abstract three-move identification scheme that bears close resemblance to the three-move form of a Σ-protocol. We remodel their construction as a Σ-protocol template and add new adaptive Σ-protocol-specific abstractions for the `SimProve`, `SimRand`, `RealToSim`, and `Extract` algorithms. We narrow the focus of our abstraction to modules, and discuss the potential implications of module properties for Σ-protocols in Appendix A.13. We highlight the properties of linear function families that are critical to understanding our results as part of the proof below.

Definition 7 (Simple Adaptive Σ-protocol Candidate). *The* simple adaptive Σ-protocol candidate $\Sigma_{\mathtt{simple}}$ *for relation R is a tuple of efficient procedures* $\Sigma_{\mathtt{simple}} = (\mathtt{Setup}, \mathtt{Prove}, \mathtt{Verify}, \mathtt{SimSetup}, \mathtt{SimProve}, \mathtt{SimRand}, \mathtt{RealToSim}, \mathtt{Extract})$ *that adhere to the following format:*

- $\mathtt{ppm} \leftarrow \mathtt{Setup}(1^\lambda)$*: Define the linear function family* $\mathtt{LF} = (\mathtt{PGen}, \mathtt{F})$ *(Definition 3.1 in [38]) such that* $\mathtt{ppm} \leftarrow \mathtt{PGen}(1^\lambda)$ *fixes a statement space $\mathcal{X}$, witness space $\mathcal{W}$, first message space $\mathcal{A}$, challenge space $\mathcal{C}$, response space $\mathcal{B}$, and randomness space $\mathcal{R}$ such that $\mathcal{W} = \mathcal{R} = \mathcal{B}$, $\mathcal{X} = \mathcal{A}$, $\mathcal{W}$ and $\mathcal{X}$ form* modules *over $\mathcal{C}$, $|\mathcal{X}| \geq |\mathcal{C}| \geq 2^{2\lambda}$, and the evaluation function $\mathtt{F} : \mathcal{W} \to \mathcal{X}$.*
- $\pi \leftarrow \mathtt{Prove}((\mathtt{ppm}, x, w, r), (\mathtt{ppm}, x, \mathtt{chl}))$*: P sends V a first message* $\mathtt{com} = \mathtt{F}(r)$*. V replies with a challenge* $\mathtt{chl} \leftarrow_{\$} \mathcal{C}$*. P responds with* $\mathtt{res} = \mathtt{chl} \cdot w + r$*.*
- $\{0,1\} \leftarrow \mathtt{Verify}(\mathtt{ppm}, x, \pi)$*: Parse π as* $(\mathtt{com}, \mathtt{chl}, \mathtt{res})$*. If* $\mathtt{F}(\mathtt{res}) = \mathtt{chl} \cdot x + \mathtt{com}$*, output 1 (accept). Otherwise, output 0 (reject).*
- $\mathtt{ppm} \leftarrow \mathtt{SimSetup}(1^\lambda)$*: Invoke* $\mathtt{Setup}(1^\lambda)$ *and return* $\mathtt{ppm}$*.*
- $\pi \leftarrow \mathtt{SimProve}(\mathtt{ppm}, x, \mathtt{chl}, r')$ *: Parse* $r' = \mathtt{res}$ *and compute* $\mathtt{com} = F(\mathtt{res} - \mathtt{chl} \cdot w)$*. Return* $\pi = (\mathtt{com}, \mathtt{chl}, \mathtt{res})$*.*
- $r \leftarrow \mathtt{SimRand}(\mathtt{ppm}, x, \pi, w)$ *: Parse* $\pi = (\mathtt{com}, \mathtt{chl}, \mathtt{res})$*. Compute and output* $r = \mathtt{res} - (\mathtt{chl} \cdot w)$*.*
- $r' \leftarrow \mathtt{RealToSim}(\mathtt{ppm}, \pi)$ *: Parse* $\pi = (\mathtt{com}, \mathtt{chl}, \mathtt{res})$ *and output* $r' = \mathtt{res}$*.*

– $w \leftarrow \texttt{Extract}(\texttt{ppm}, x, \pi, \pi')$: *Given any two proofs* $\pi = (\texttt{com}, \texttt{chl}, \texttt{res})$ *and* $\pi' = (\texttt{com}, \texttt{chl}', \texttt{res}')$ *such that* $\texttt{Verify}(x, \pi) = \texttt{Verify}(x, \pi') = 1$ *and* $\texttt{chl} \neq \texttt{chl}'$*, compute and output* $w = (\texttt{res} - \texttt{res}') \cdot (\texttt{chl} - \texttt{chl}')^{-1}$.

Theorem 2 ($\Sigma_{\texttt{simple}}$ is Adaptive). *The simple adaptive Σ-protocol candidate given above (Definition 7) is an adaptive Σ-protocol (Definition 3) with randomness equivocability (Definition 4).*

Proof. The proof relies on two core properties of the evaluation function $\texttt{F}$, described below (for details in context, see Definition 4.1 [38]). First, the *pseudo-module homomorphism* (PMH) property states that for all $y, z \in \mathcal{W}$, $\texttt{F}(\texttt{chl} \cdot y + z) = \texttt{chl} \cdot \texttt{F}(y) + \texttt{F}(z)$. Second, the *smoothness property* guarantees that for all $y \in \mathcal{W}$, $\texttt{F}(y)$ is uniformly distributed over $\mathcal{X}$. (Recall these properties hold as well for $\mathcal{W} = \mathcal{R} = \mathcal{B}$ and $\mathcal{X} = \mathcal{A}$.) Verifying that the simple Σ-protocol format above satisfies the necessary completeness, adaptive special honest-verifier zero-knowledge, special soundness, and randomness equivocability properties is straight-forward; we defer the details to Appendix A.14.

Adaptive Proof of Knowledge of Discrete Logarithm. Proofs of knowledge of discrete logarithm (PoK DL), also known as Schnorr proofs [48], are the backbone of many practical constructions of group [17,8], threshold [4], blind [38], and multi- [4,30] signatures. See Appendix A.15 for the proof of:

Theorem 3 ($\Sigma_{\texttt{DL}}$ is Simple Adaptive). *A proof of knowledge of discrete logarithm can be described as a simple adaptive Σ-protocol with randomness equivocability (Definition 7).*

Adaptive Proof of Knowledge of Equality of n Representations. Proofs of knowledge of equality of n representations of m witnesses (PoK EqRep) are essential building blocks in the construction of cryptographic shuffles [54,43], accumulators [2,13], and anonymous credentials [12], which form the basis of anonymous networks, voting, payment, and identification systems. Note also that PoK EqRep is a generalization of other common Σ-protocols such as standard PoK of representation [18,16] and Okamoto-Schnorr [45], and that the following result holds for those narrower instantiations as well. See Appendix A.16 for the proof of:

Theorem 4 ($\Sigma_{\texttt{EqRep}}$ is Simple Adaptive). *A proof of knowledge of equality of n representations can be described as a simple adaptive Σ-protocol with randomness equivocability (Definition 7).*

4 Adaptive Straight-Line Compilers

Recall from the introduction that an adaptive straight-line compiler is a straight-line compiler [42] that preserves the adaptive security properties of the Σ-protocol under transformation. A regular straight-line compiler $\texttt{SLC}$ takes a Σ-protocol Σ_R for a relation R as input and produces a new proof system $\Pi_R^{\texttt{SLC}}$ that

is a tuple of (non-interactive) *algorithms* $(\texttt{Setup}^H, \texttt{Prove}^H, \texttt{Verify}^H, \texttt{SimSetup}, \texttt{SimProve}, \texttt{Extract})$, where H is a traditional RO. In order to be considered a straight-line compiler, $\Pi_R^{\texttt{SLC}}$ must have the following properties: overwhelming completeness (i.e. a negligibly small completeness error is allowed), non-interactive multiple special honest-verifier zero-knowledge (NIM-SHVZK), and non-interactive special simulation soundness (NI-SSS) [42].

4.1 Definition of an Adaptive Straight-Line Compiler

Our *adaptive* SLC, denoted $\texttt{aSLC}$, says that $\Pi_R^{\texttt{aSLC}}$ must have *adaptive* NIM-SHVZK and *adaptive* NI-SSS properties—that is, NIM-SHVZK and NI-SSS must hold even when the adversary gets to compare the prover's internal tapes with the output of $\texttt{SimRand}$.

Definition 8 (Adaptive Straight-Line Compiler). *An algorithm* $\texttt{SLC}$ *is an* adaptive straight-line compiler (adaptive SLC) *in the random-oracle model if given any adaptive* Σ*-protocol* Σ_R *for relation* R *(Definition 2) as input, it outputs a tuple of* algorithms $\Pi_R^{\texttt{SLC}} = (\texttt{Setup}^H, \texttt{Prove}^H, \texttt{Verify}^H, \texttt{SimSetup}, \texttt{SimProve}, \texttt{SimRand}, \texttt{Extract})$ *based on* Σ_R *that satisfy the following properties: overwhelming completeness (Definition 9), adaptive non-interactive multiple special honest-verifier zero-knowledge (Definition 10), and adaptive non-interactive special simulation soundness (Definition 11).*

We refer to $\Pi_R^{\texttt{aSLC}} \leftarrow \texttt{aSLC}(\Sigma_R)$ *as an adaptive and non-interactive straight-line extractable (adaptive NISLE) proof system for* R*, and proofs generated by* $\Pi_R^{\texttt{aSLC}}$ *as adaptive and non-interactive straight-line extractable zero-knowledge proofs of knowledge (adaptive NISLE ZKPoK).*

Definition 9 (Overwhelming Completeness). *An adaptive NISLE proof system* $\Pi_R^{\texttt{aSLC}}$ *for relation* R *in the random-oracle model has the* overwhelming completeness *property if for any security parameter* λ*, any* $(x, w) \in R$*, and any proof* $\pi \leftarrow \Pi_R^{\texttt{aSLC}}.\texttt{Prove}^H(x, w)$, $\Pr[\Pi_R^{\texttt{aSLC}}.\texttt{Verify}^H(x, \pi) = 1] \geq 1 - \mathsf{negl}(\lambda)$.

To capture the difference between a standard random oracle (modeled as a truly random function) and its programmable version (modeled as a data structure that can be edited by the security experiment), the RO H_f is parameterized by a function $f \leftarrow_\$ F$ selected from random function family F, while the RO H_L is parameterized by an initially empty list L that the challenger in the security experiment can program via an interface $\texttt{Prog}_L$. When programming does not occur, the list oracle simply selects a return value uniformly at random from the output space. In order to maintain indistinguishability between the experiments and satisfy the (adaptive) NIM-SHVZK property, the ideal-world challenger must program the RO imperceptibly.[3] For convenience the interface $\texttt{Prog}$ is included in the definitions of NIM-SHVZK and NI-SSS below; in non-programmable models, one can replace $\texttt{Prog}_L$ with H_L.

[3] To satisfy NIM-SHVZK when $\Sigma_R.\texttt{SimProve}$ involves programming, the first message $\texttt{com}$ will need entropy that is superlogarithmic in the security parameter [35,42].

Definition 10 (Adaptive Non-Interactive Multiple SHVZK). *An adaptive NISLE proof system* Π_R^{aSLC} *for relation* R *in the random-oracle model has the* adaptive non-interactive multiple special honest-verifier zero-knowledge (adNIM-SHVZK) *property if there exist algorithms* $\Pi_R^{\mathsf{SLC}}.\mathtt{SimSetup}$, $\Pi_R^{\mathsf{SLC}}.\mathtt{SimProve}$, *and* $\Pi_R^{\mathsf{SLC}}.\mathtt{SimRand}$ *such that for any security parameter* λ, *any* PPT *adversary* $\mathscr{A}$, *and a bit* $b \leftarrow_{\$} \{0,1\}$, *there exists some negligible function* negl *such that* $\Pr[b' = b] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$, *where* b' *is the result of running the game* $\mathsf{adNIM\text{–}SHVZK}^{H_*,*}_{\mathscr{A},\Pi_R^{\mathsf{SLC}}}(1^\lambda, b)$ *from Figure 4.*

$\mathsf{adNIM\text{–}SHVZK}^{H_*,F}_{\mathscr{A},\Pi_R^{\mathsf{SLC}}}(1^\lambda, 0)$: Real	$\mathsf{adNIM\text{–}SHVZK}^{H_*,\mathtt{Prog}}_{\mathscr{A},\Pi_R^{\mathsf{SLC}}}(1^\lambda, 1)$: Ideal
$f \leftarrow_{\$} F$	$L \leftarrow \bot$
$\mathtt{ppm} \leftarrow \Pi_R^{\mathsf{SLC}}.\mathtt{Setup}^{H_f}(1^\lambda)$	$(\mathtt{ppm}, z_{\mathtt{Sim}}) \leftarrow \Pi_R^{\mathsf{SLC}}.\mathtt{SimSetup}^{\mathtt{Prog}_L}(1^\lambda)$
$\mathtt{st} \leftarrow \mathscr{A}^{H_f}(1^\lambda, \mathtt{ppm})$	$\mathtt{st} \leftarrow \mathscr{A}^{H_L}(1^\lambda, \mathtt{ppm})$
while $\mathtt{st} \notin \{0,1\}$:	**while** $\mathtt{st} \notin \{0,1\}$:
$(\mathtt{Prove}, x, w, \mathtt{st}) \leftarrow \mathscr{A}^{H_f}(\mathtt{st})$	$(\mathtt{Prove}, x, w, \mathtt{st}) \leftarrow \mathscr{A}^{H_L}(\mathtt{st})$
if $R(x, w) = 1$:	**if** $R(x, w) = 1$:
$r \leftarrow_{\$} \{0,1\}^{\lambda_r}$	$(\pi, z_\pi) \leftarrow \Pi_R^{\mathsf{SLC}}.\mathtt{SimProve}^{\mathtt{Prog}_L}(z_s, x)$
$\pi \leftarrow \Pi_R^{\mathsf{SLC}}.\mathtt{Prove}^{H_f}(x, w, r)$	$r \leftarrow \Pi_R^{\mathsf{SLC}}.\mathtt{SimRand}^{\mathtt{Prog}_L}(z_{\mathtt{Sim}}, z_\pi, x, \pi, w)$
else :	**else** :
$\pi, r \leftarrow \bot$	$\pi, r \leftarrow \bot$
$\mathtt{st} \leftarrow \mathscr{A}^{H_f}(\mathtt{st}, \pi, r)$	$\mathtt{st} \leftarrow \mathscr{A}^{H_L}(\mathtt{st}, \pi, r)$
return st	**return** st

Fig. 4. Adaptive Non-Interactive Multiple SHVZK (adNIM-SHVZK) Game.

Definition 11 (Adaptive Non-Interactive SSS). *An adaptive NISLE proof system* Π_R^{SLC} *for relation* R *in the random-oracle model has the* adaptive non-interactive special simulation sound (adNI-SSS) *property if there exists an algorithm* $\Pi_R^{\mathsf{SLC}}.\mathtt{Extract}$ *such that for any security parameter* λ *and any* PPT *adversary* $\mathscr{A}$, $\Pr[\mathtt{Fail} \leftarrow \mathsf{adNI\text{–}SSS}^{H}_{\mathscr{A},\Pi_R^{\mathsf{SLC}}}(1^\lambda)] \leq \mathsf{negl}(\lambda)$, *where* H *is any random oracle and* $\mathsf{adNI\text{–}SSS}$ *is the game described in Figure 5.*

4.2 The Randomized Fischlin Transform is an Adaptive SLC

The randomized Fischlin transform [35,41] $\mathtt{rFis}$ is a (standard) straight-line compiler that transforms any Σ-protocol with certain general properties into a non-interactive, straight-line extractable proof system $\Pi_R^{\mathtt{aFis}}$ in the ROM [42].

$\mathsf{adNI\text{–}SSS}^{H_L,\mathtt{Prog}_L}_{\mathscr{A},\Pi^{\mathtt{SLC}}_R}(1^\lambda)$

$L, \mathtt{pflist}, \mathtt{Response}, \mathtt{st} \leftarrow \bot;$
$(\mathtt{ppm}, z_{\mathtt{Sim}}) \leftarrow \Pi^{\mathtt{SLC}}_R.\mathtt{SimSetup}^{\mathtt{Prog}_L}(1^\lambda)$
while $(\mathtt{Query}, \mathcal{Q}_{\mathscr{A}}, \mathtt{st}) \leftarrow \mathscr{A}^{H_L}(1^\lambda, \mathtt{ppm}, \mathtt{st}) \neq \mathtt{halt}$:
  if $\mathtt{Query} = (\mathtt{Prove}, x, w) \wedge R(x, w) = 1$:
    $\mathtt{chl} \leftarrow_\$ \{0,1\}^\ell; r_{\mathtt{Sim}} \leftarrow_\$ \{0,1\}^{\lambda_{\mathtt{Sim}}}$
    $(\pi, z_\pi) \leftarrow \Pi^{\mathtt{SLC}}_R.\mathtt{SimProve}^{\mathtt{Prog}_L}(z_{\mathtt{Sim}}, x, \mathtt{chl}, r_{\mathtt{Sim}})$
    $\mathtt{pflist}.\mathbf{append}(x, \pi)$
    $r \leftarrow \Pi^{\mathtt{SLC}}_R.\mathtt{SimRand}^{\mathtt{Prog}_L}(z, z_\pi, x, \pi, w, r_{\mathtt{Sim}})$
    $\mathtt{Response} \leftarrow (x, \pi, r)$
  elseif $\mathtt{Query} = (\mathtt{Challenge}, x, \pi)$
    if $\Pi^{\mathtt{SLC}}_R.\mathtt{Verify}^{H_L}(x, \pi) = 1 \wedge (x, \pi) \notin \mathtt{pflist}$:
      $w \leftarrow \Pi^{\mathtt{SLC}}_R.\mathtt{Extract}(x, \pi, \mathcal{Q}_{\mathscr{A}})$
      if $R(x, w) = 0$: **return** $\mathtt{Fail}$
  $\mathtt{st} \leftarrow \mathscr{A}^{H_L}(\mathtt{st}, \mathtt{Response})$
return $\mathtt{Success}$

Fig. 5. Adaptive Non-Interactive Special Simulation Soundness (adNI-SSS) Game.

The prover in $\mathtt{rFis}$ essentially rewinds itself, computing proofs on repeated commitments (but different challenges) until it is guaranteed with overwhelmingly probability that there are at least two transcripts queried to the RO with the same commitment but different challenges. The $\mathtt{rFis}$ extractor takes these proofs as input (via the adversary's oracle query history $\mathcal{Q}_{\mathscr{A}}$), and can therefore extract witnesses for adversarially-created proofs without rewinding the prover.

In the remainder of this section, we extend $\mathtt{rFis}$ for the adaptive setting, and prove that our construction satisfies the definition of an adaptive straight-line compiler. In particular, we will show that the *adaptive* randomized Fischlin transform $\mathtt{aFis}$, which extends $\mathtt{rFis}$ with a $\mathtt{SimRand}$ functionality, preserves the adaptive security property of the underlying Σ-protocol: as long as Σ_R is adaptive and conforms to the standard requirements of $\mathtt{rFis}$ (reviewed in Appendix A.12), $\mathtt{aFis}$ is an adaptive SLC. The $\Pi^{\mathtt{aFis}}_R.\mathtt{SimRand}$ functionality, which is in charge of producing a convincing version Q of the prover's random tape, works as follows.

In order to reconstruct the random "first-message" section of Q (i.e. the section used to run $\tau.\mathtt{Commit}$), the $\Pi^{\mathtt{aFis}}_R.\mathtt{SimRand}$ algorithm runs $\Sigma_{R,\tau}.\mathtt{SimRand}$ using auxiliary output from $\Pi^{\mathtt{aFis}}_R.\mathtt{SimProve}$. To simulate the random "challenge-selection" section of Q (i.e. the section used to generate the prover's random selection of challenges), $\Pi^{\mathtt{aFis}}_R.\mathtt{SimRand}$ first gathers the random coins corresponding to all of the challenges sampled by $\Pi^{\mathtt{aFis}}_R.\mathtt{SimProve}$. Note that in order

for (not only the challenges in the proof tuple) but *all* of the challenges included in Q to agree with the output of the RO, $\Pi_R^{\mathtt{aFis}}.\mathtt{SimRand}$ must be able to simulate the prover's entire sequence of challenges. In the adaptive setting, generating a convincing version of the prover's entire challenge sequence is non-trivial, and in particular requires a simulation technique other than the dynamic programming strategy of the original and randomized Fischlin transforms [35,41].

Fortunately, Kondi and shelat's randomization of the challenge space allows the $\Pi_R^{\mathtt{aFis}}.\mathtt{SimRand}$ algorithm to finish the challenge section of the prover's random tape by computing transcripts using the real witness, sampling a fresh challenge-to-output mapping, and inserting the challenge used to compute the simulated transcript as the first input to produce the all-zero string. We assume that the challenge space ℓ is exponential, or on the order of 2^λ, with respect to security parameter λ. This is true of many Σ-protocols, and in particular discrete logarithm-based protocols that are the primary focus of this work. For those that do not have this property a priori, the security parameter can be sufficiently increased using parallel repetition of the proof process, trading linear multiplicative overhead for adaptive security. We predict that this restriction can be relaxed, but leave the exploration of those techniques to future work.

Construction 1 (Adaptive Randomized Fischlin Transform). Let $\Sigma_{R,\tau}$ be any adaptive Σ-protocol for relation R (Definition 2) based on protocol template τ (Appendix A.5) with the required properties for $\mathtt{rFis}$ (Appendix A.12). Let H be any random oracle and $\mathtt{trunc}_b$ be a truncation algorithm which returns the first b bits of $H(x)$ for some input x. Then the *adaptive randomized Fischlin transform* of $\Sigma_{R,\tau}$, denoted $\mathtt{aFis}$, is an algorithm that takes $\Sigma_{R,\tau}$ as input and creates a tuple of algorithms $\Pi_R^{\mathtt{aFis}} = (\mathtt{Setup}^H, \mathtt{Prove}^H, \mathtt{Verify}^H, \mathtt{SimSetup}, \mathtt{SimProve}, \mathtt{SimRand}, \mathtt{Extract})$, defined as follows.

- $\mathtt{ppm} \leftarrow \mathtt{Setup}^H(1^\lambda)$: H is fixed. Let b, n, t be set according to the randomized Fischlin transform [41,35]: $bn = \omega(\log\lambda)$, $2^{t-b} = \omega(\log\lambda)$, $b, n, t = O(\log\lambda)$ and $b \leq t$, where we use n in place of r repetitions to avoid confusion with the notation for randomness. Let $\ell = O(2^\lambda)$. The compound randomness security parameters are $\lambda_Q = \lceil n(\lambda_r + 2^t\ell)\rceil$ and $\lambda_{Q'} = \lceil n(\lambda'_r + \ell) + n(\lambda_r + 2^t\ell)\rceil$, where λ_r and $\lambda_{r'}$ are the randomness security parameters of $\Sigma_{R,\tau}$. Run $\mathtt{ppm}_\Sigma \leftarrow \tau.\mathtt{Setup}(1^\lambda)$. Output the public parameters $\mathtt{ppm} = (\mathtt{ppm}_\Sigma, b, n, t, \ell, \lambda_Q)$.
- $(x, \Phi) \leftarrow \mathtt{Prove}^H(x, w, Q)$: Compute the vector of n first messages $\overline{\mathtt{com}} = \langle \mathtt{com}_1, \ldots, \mathtt{com}_n \rangle$ by running $\tau.\mathtt{Commit}(x, w, r_i)$ for $1 \leq i \leq n$, where each r_i consumes λ_r bits of the random tape Q. To compute each proof π_i, sample challenges $\mathtt{chl}_j$ from Q and compute $\mathtt{res}_j \leftarrow \tau.\mathtt{Respond}(x, w, \mathtt{com}_i, \mathtt{chl}_j)$ until $\mathtt{trunc}(H(x, \overline{\mathtt{com}}, i, \mathtt{chl}_j, \mathtt{res}_j)) = 0^b$, then set $\pi_i = (\mathtt{com}_i, \mathtt{chl}_j, \mathtt{res}_j)$. Finally, return (x, Φ), where $\Phi = (\pi_1, \ldots, \pi_n)$.
- $\{0,1\} \leftarrow \mathtt{Verify}^H(x, \Phi)$: Parse $\Phi = (\pi_1, \ldots, \pi_n)$. Output 1 (accept) if and only if $\Sigma_R.\mathtt{Verify}(x, \pi_i) = 1$ and $\sum_{i=1}^n \mathtt{trunc}(H(x, \overline{\mathtt{com}}, i, \mathtt{chl}_i, \mathtt{res}_i)) = 0^b$ for $1 \leq i \leq n$. Otherwise, output 0 (reject).

- $(\mathtt{ppm}, z_s) \leftarrow \mathtt{SimSetup}(1^\lambda)$: Fix H and generate b, n, t the same as in $\mathtt{Setup}$. Generate $\mathtt{ppm}_\Sigma$ and simulator state information z_s by running $\Sigma_{R,\tau}.\mathtt{SimSetup}$. Set $\mathtt{ppm} = (\mathtt{ppm}_\Sigma, b, n, t, \lambda_R)$ and return $(\mathtt{ppm}, z_s)$.
- $(x, \Phi, Z_\Phi) \leftarrow \mathtt{SimProve}(x, z_s, Q')$: For each proof $1 \leq i \leq n$, sample a random r_i and a random challenge $\mathtt{chl}_i$ from Q'. Then run $\Sigma_{R,\tau}.\mathtt{SimProve}(x, z_s, \mathtt{chl}_i, r_i)$ to obtain $\pi_i = (\mathtt{com}_i, \mathtt{chl}_i, \mathtt{res}_i)$ and z_{π_i}. For each proof, program the truncated output of H on input $(x, \overline{\mathtt{com}}, i, \mathtt{chl}_i, \mathtt{res}_i)$ to be 0^b. Finally, output the proof tuple (x, Φ) and auxiliary information Z_Φ, where $\Phi = (\Phi_1, \ldots, \Phi_n)$ and $Z_\Phi = (z_{\pi_1}, \ldots, z_{\pi_n})$.
- $Q \leftarrow \mathtt{SimRand}(\mathtt{ppm}, z_s, Z_\Phi, x, \Phi, w, Q')$: Parse $\Phi = (\pi_1, \ldots, \pi_n)$ and $Z_\Phi = (z_{\pi_1}, \ldots, z_{\pi_n})$. For each proof $\pi_i = (\mathtt{com}_i, \mathtt{chl}_i, \mathtt{res}_i)$, run $\Sigma_{R,\tau}.\mathtt{SimRand}(\mathtt{ppm}, z_s, z_{\pi_i}, x, \pi_i, w)$ to obtain r_i, and concatenate it to Q. Then, sample a fresh random challenge $\mathtt{chl}_j$ from Q' and compute $\mathtt{res}_j \leftarrow \tau.\mathtt{Respond}(\mathtt{ppm}, x, w, \mathtt{com}_j, \mathtt{chl}_j)$. If $\mathtt{trunc}(H(x, \overline{\mathtt{com}}, j, \mathtt{chl}_j, \mathtt{res}_j)) = 0^b$, concatenate $\mathtt{chl}_i$ to Q and move on to the next proof π_{i+1}; otherwise, concatenate $\mathtt{chl}_j$ to Q and continue sampling challenges until the condition is met. Finally, return Q.
- $w \leftarrow \mathtt{Extract}(X, \Phi, \mathcal{Q}_\mathscr{A})$: Parse $\Phi = (\pi_1, \ldots, \pi_n)$ and each $\pi_i = (\mathtt{com}_i, \mathtt{chl}_i, \mathtt{res}_i)$. Given a list $\mathcal{Q}_\mathscr{A}$ the adversary's queries to H, search for two queries $(x, \overline{\mathtt{com}}, i, \mathtt{chl}_i, \mathtt{res}_i)$ and $(x, \overline{\mathtt{com}}, i, \mathtt{chl}'_i, \mathtt{res}'_i)$ such that $(\mathtt{chl}_i, \mathtt{res}_i) \neq (\mathtt{chl}'_i, \mathtt{res}'_i)$ and $\Sigma_R.\mathtt{Verify}(x, \pi_i) = \Sigma_R.\mathtt{Verify}(x, \pi'_i) = 1$. If no such queries exist, output $\mathtt{Fail}$. Otherwise, obtain w by running $\Sigma_R.\mathtt{Extract}(x, \pi, \pi')$.

We now prove that as long as the Σ-protocol used as input to the transform is adaptive special honest-verifier zero-knowledge (adSHVZK), $\mathtt{aFis}$ is an adaptive SLC. We will show in Section 5 that any adaptive SLC is sufficient to bootstrap adaptive Σ-protocols into adaptive UC NIZKPoK in the global ROM(s).

Theorem 5 ($\mathtt{aFis}$ is an Adaptive SLC). *Let $\Sigma_{R,\tau}$ be an adaptive Σ-protocol based on protocol template τ for relation R (Definition 2) with the required properties for $\mathtt{rFis}$ (Appendix ??), and $H : \{0,1\}^* \to \{0,1\}^b$ be any random oracle. Then the adaptive randomized Fischlin transform $\mathtt{aFis}$ (Definition 1) is an adaptive straight-line compiler for $\Sigma_{R,\tau}$ (Definition 8) in the ROM.*

Proof. Recall that an adaptive straight-line compiler according to our definition must create protocols that are overwhelmingly complete, adaptive non-interactive multiple special honest-verifier zero-knowledge (adNIM-SHVZK), and adaptive non-interactive special simulation-sound (adNI-SSS).

First, note that since the specification of $\Pi_R^{\mathtt{aFis}}.\mathtt{Prove}$ does not functionally change between $\mathtt{rFis}$ and $\mathtt{aFis}$ (as our explicit treatment of the prover's randomness is syntactic), $\mathtt{aFis}$ is as complete as $\mathtt{rFis}$ [35,41]. We first construct a series of hybrid arguments to show that if $\Pi_R^{\mathtt{aFis}}$ is not adNIM-SHVZK, then Σ_R cannot be regular adaptive special honest-verifier zero-knowledge (adSHVZK), contradicting the assumption in the theorem statement.

Hybrid 0. This is the "real-world" experiment, where honest parties compute proofs using $\Pi_R^{\mathtt{aFis}}.\mathtt{Prove}$, verify proofs using $\Pi_R^{\mathtt{aFis}}.\mathtt{Verify}$, reveal their random

tapes upon corruption, and the random oracle is setup in "random-function" mode (as in the $b = 0$ experiment of the adNIM-SHVZK game in Figure 4).

Hybrid 1. The random oracle is setup by an experiment "challenger" in "list" mode, where all queries are recorded in a list. The challenger draws the oracle's response to new queries uniformly at random from the response space, then "programs" them (this is still mostly the $b = 0$ experiment in the adNIM-SHVZK from Figure 4, but rather than using function f the security experiment is using list L, and the resulting oracle is denoted H_L). Hybrids 0 and 1 are statistically indistinguishable as the challenger has essentially replaced the "real-world" oracle with a statistically identical process of choosing random values and storing them in a function table.

We now wish to construct a reduction $\mathscr{B}$ that uses an algorithm $\mathscr{A}$ that can win the adNIM-SHVZK game (Definition 10) parameterized over $\Pi_R^{\mathtt{aFis}}$ with non-negligible advantage as a black-box in order to win the adSHVZK game (Definition 3) parameterized over Σ_R with non-negligible advantage. Note that the adSHVZK challenger is expecting exactly one `Prove` query (i.e. it is not *multi*-adaptive), and will issue only one proof in return (as opposed to the tuple of proofs returned by $\Pi_R^{\mathtt{aFis}}.\mathtt{Prove}$). We therefore modify $\mathscr{A}$ to distinguish two hybrids, $(1, i, j)$ and $(1, i, j + 1)$.

Hybrid $(1, i, j)$. For $0 < i < \mathtt{max}$ and any $0 \leq j \leq n$ where $\mathtt{max} = \mathsf{poly}(\lambda)$ is the maximum number of `Prove` queries, the first $i - 1$ responses to $\mathscr{A}$'s `Prove` queries are all formed using the $\Pi_R^{\mathtt{aFis}}.\mathtt{Prove}$ algorithm, and starting with the $(i + 1)^{st}$ query, $\Pi_R^{\mathtt{aFis}}.\mathtt{SimProve}$ is used (along with the setup parameters from the $\Pi_R^{\mathtt{aFis}}.\mathtt{SimSetup}$ algorithm). The corresponding randomness of these proofs are computed using the $\Pi_R^{\mathtt{aFis}}.\mathtt{SimRand}$ algorithm. For the i^{th} `Prove` query, the response is a hybrid in which the first j proofs are computed using the $\Sigma.\mathtt{Prove}$ algorithm, while the rest are computed using the $\Sigma.\mathtt{SimProve}$ algorithm (again with setup from $\Sigma.\mathtt{SimProve}$ and randomness according to $\Sigma.\mathtt{SimRand}$). Hybrid $(1, \mathtt{max}, n)$ is identical to Hybrid 1 and Hybrid $(1, 0, 0)$ is identical to the $b = 1$ adNIM-SHVZK experiment from Figure 4.

Note that as part of the simulation algorithms for both Σ and $\Pi_R^{\mathtt{aFis}}$, $\mathscr{B}$ may need to program the RO H using the interface $\mathtt{Prog}_L$. Because H is set up in "list" mode, the security experiment allows programming from authorized parties, which in this case is the adSHVZK challenger and also $\mathscr{B}$ after the $(i, j+1^{st})$ proof is fixed. Without loss of generality we can imagine the reduction returns each completed proof in the tuple sequentially, as each one is computed and fixed independently once the commitment vector is generated. Note also that since all programmed points are generated independently, the ability of $\mathscr{B}$ to program at this stage does not give $\mathscr{B}$ any insight into what has been programmed previously (i.e. $\mathscr{B}$ does not have an advantage in determining whether the challenger programmed the oracle as part of its answer to the $j + 1^{st}$ prove query).

We argue that Hybrid $(1, i, j)$ is computationally indistinguishable from Hybrid $(1, i, j + 1)$ as long as Σ is adSHVZK. Assume for a contradiction that it is not. The reduction $\mathscr{B}$ proceeds as follows.

At the beginning of the experiment, $\mathscr{B}$ receives 1^λ and $\texttt{ppm}$ from the adSHVZK challenger. Note that the $\texttt{ppm}$ associated with Σ (i.e. other than the $\texttt{aFis}$ parameters (b, n, t)) are exactly the same between $\Sigma.\texttt{Setup}$, $\Sigma.\texttt{SimSetup}$, $\Pi_R^{\texttt{aFis}}.\texttt{Setup}$, and $\Pi_R^{\texttt{aFis}}.\texttt{SimSetup}$; the only difference is that the challenger in the adSHVZK and adNIM-SHVZK experiments might additionally generate some initial state information $z_{\texttt{Sim}}$ that it might use while simulating future proofs. The fact that the challenger has or has not generated $z_{\texttt{Sim}}$ is obviously not revealed to $\mathscr{B}$, but note that the value is deterministic with respect to the security parameter—$\mathscr{B}$ will generate the same $\texttt{ppm}, z_{\texttt{Sim}}$ upon running $\texttt{SimSetup}$ itself. The way the game is defined states that the adversary receives 1^λ and $\texttt{ppm}$ from the challenger, but equivalently, you could think of the adversary receiving 1^λ only, and generating the same $\texttt{ppm}$ itself.

Up to the i^{th} query $(\texttt{Prove}, x, w)$ from $\mathscr{A}$, the reduction runs the $\Pi_R^{\texttt{aFis}}.\texttt{Setup}$ algorithm itself to obtain $\texttt{ppm}$ and computes proofs entirely on its own: $\mathscr{B}$ samples randomness $Q \leftarrow \{0,1\}^{\lambda_Q}$, runs $\Pi_R^{\texttt{aFis}}.\texttt{Prove}(x, w, Q)$, and returns (Φ, Q) to $\mathscr{A}$. On $\mathscr{A}$'s i^{th} query $(\texttt{Prove}, x^*, w^*)$, $\mathscr{B}$ passes $(\texttt{Prove}, x^*, w^*)$ to the adSHVZK challenger and receives (π^*, r^*) for $\pi^* = (\texttt{com}^*, \texttt{chl}^*, \texttt{res}^*)$ that is either the result of running $\Sigma_R.\texttt{Prove}$ on randomness r^* or the result of running $\Sigma_R.\texttt{SimProve}$ and $\Sigma_R.\texttt{SimRand}$. Recall that that $\mathscr{A}$ is expecting more proofs than just π^*, as the output of $\Pi_R^{\texttt{aFis}}.\texttt{Prove}$ and $\Pi_R^{\texttt{aFis}}.\texttt{SimProve}$ should actually be a tuple of proofs $\Phi_i = \pi_1, \ldots, \pi_n$. Moreover, since each proof in the tuple $\pi_j \in \Phi_i$ requires the prover to make $2^t - 1$ secret queries to the RO, $\mathscr{A}$ is expecting more randomness than just r^*. Without loss of generality, we define each chunk Q_j of the prover's random tape Q to be a λ_r-length segment corresponding to the first-message randomness for π_j, followed by a 2^t-length segment corresponding to the random challenges for each RO query, where t is the total number of queries the prover makes to the RO in the process of producing π_j. $\mathscr{B}$ constructs the hybrid proof tuple Φ_i from the challenge query as follows.

Let the j^{th} commitment $\texttt{com}_j = \texttt{com}^*$, and $\pi_j = \pi^* = (\texttt{com}^*, \texttt{chl}^*, \texttt{res}^*)$. Since it must have the entire tuple $\texttt{com}_1, \ldots, \texttt{com}_n$ of commitments before running $\Pi_R^{\texttt{aFis}}.\texttt{Prove}$, $\mathscr{B}$ starts by running the steps of $\Pi_R^{\texttt{aFis}}.\texttt{SimProve}$ $n-j$ times to obtain $\pi_{j+1}, \ldots, \pi_n$. $\mathscr{B}$ then samples each $r_1, \ldots r_{j-1}$ from Q' and runs $\tau.\texttt{Commit}$ $j-1$ times to obtain $\texttt{com}_1, \ldots, \texttt{com}_{j-1}$. It then finalizes the commitment vector $\overline{\texttt{com}} = \texttt{com}_1, \ldots, \texttt{com}_n$ and programs the truncated output of H to output 0^b on input $(x, \overline{\texttt{com}}, k, \texttt{chl}_k, \texttt{res}_k)$ for each simulated proof π_k, where $k \in [j+1, n]$. $\mathscr{B}$ computes each proof $\pi_1, \ldots, \pi_{j-1}$ by running the steps of $\Pi_R^{\texttt{aFis}}.\texttt{Prove}$ starting right after the generation of the commitment vector. $\mathscr{B}$ generates the prover's hybrid random tape Q for the proof tuple Φ as follows. For each of the first chunks $Q_m \in Q_1, \ldots, Q_{j-1}$, $\mathscr{B}$ concatenates the first-message randomness r_m and challenge randomness $\texttt{chl}_1, \ldots, \texttt{chl}_t$ it used in the execution of $\Pi_R^{\texttt{aFis}}.\texttt{Prove}$. The j^{th} chunk, which must contain the bits corresponding to the challenge proof $\pi^* = \pi_j$, are generated as follows. $\mathscr{B}$ sets the first λ_r bits to be the value r^* returned by the adSHVZK challenger. $\mathscr{B}$ then samples a fresh random challenge $\texttt{chl}_m$, computes $\texttt{res}_m \leftarrow \tau.\texttt{Respond}(\texttt{ppm}, x, w, \texttt{com}^*, \texttt{chl}_m)$, and checks whether $\texttt{trunc}(H(x, \overline{\texttt{com}}, j, \texttt{chl}_m, \texttt{res}_m)) = 0^b$. If it does, $\mathscr{B}$ concatenates $\texttt{chl}^*$ to Q and

moves on to the next step. Otherwise, $\mathscr{B}$ concatenates $\texttt{chl}_m$ to Q and continues sampling challenges until the condition is met. Finally for the remaining chunks $Q_m \in Q_{j+1}, \ldots, Q_n$, $\mathscr{B}$ concatenates the first-message randomness r_m and challenge randomness $\texttt{chl}_1, \ldots, \texttt{chl}_t$ returned by executing $\Pi_R^{\texttt{aFis}}.\texttt{SimRand}$ on input the proofs $\pi_{j+1}, \ldots, \pi_n$ and auxiliary information $z_{\pi_{j+1}}, \ldots, z_{\pi_n}$ output by its execution of $\texttt{SimProve}$. $\mathscr{B}$ then returns Φ and Q to $\mathscr{A}$. From the i^{th} $\texttt{Prove}$ query onward, $\mathscr{B}$ computes proofs according to $\Pi_R^{\texttt{aFis}}.\texttt{SimProve}$ (with setup according to $\Pi_R^{\texttt{aFis}}.\texttt{SimSetup}$) and returns randomness according to $\Pi_R^{\texttt{aFis}}.\texttt{SimRand}$.

For the analysis, consider that $\mathscr{B}$'s execution of the proofs $\pi_1, \ldots, \pi_j$ and $\pi_{j+2}, \ldots, \pi_n$ and corresponding randomness is exactly what $\mathscr{A}$ expects in either hybrid. It remains to show that the challenge proof π_{j+1} and corresponding randomness is also formatted according to what $\mathscr{A}$ expects. If the adSHVZK challenger is playing with bit $b = 0$ and generated (π^*, r^*) according to $\Pi_R^{\texttt{aFis}}.\texttt{Prove}$ and the prover's random tape, then π_{j+1} and the first-message section of Q_{j+1} corresponding to r_{j+1} (i.e. the randomness used to generate $\texttt{com}^* = \texttt{com}_{j+1}$) will be exactly what $\mathscr{A}$ expects from Hybrid $(1, i, j+1)$. If π^*, r^* were generated according to $\Pi_R^{\texttt{aFis}}.\texttt{SimProve}$ and $\Pi_R^{\texttt{aFis}}.\texttt{SimRand}$ respectively, then $\mathscr{A}$'s view of π_{j+1} and r_{j+1} in Q_j will be according to Hybrid $(1, i, j)$.

For the challenge section of Q_{j+1}, note that the challenge selection process of $\Pi_R^{\texttt{aFis}}.\texttt{Prove}$ is identical to that of $\Pi_R^{\texttt{aFis}}.\texttt{SimRand}$ up to the challenge in the t^{th} transcript—the $\texttt{Prove}$ and $\texttt{SimRand}$ algorithms sample challenges and compute responses using the real witness until they find a transcript that causes the RO to return a truncated output of 0^b. When this condition occurs, the $\texttt{Prove}$ algorithm returns the accepting transcript computed using the challenge it sampled, while the $\texttt{SimRand}$ algorithm inserts the challenge and transcript generated by $\texttt{SimProve}$. For the challenge section of Q_{j+1}, $\mathscr{B}$ samples challenges according to either algorithm and when it finds a transcript that returns 0^b, it inserts the challenge $\texttt{chl}^*$ from the adSHVZK challenger. $\mathscr{A}$'s view of the challenge selection section in the case that $b = 0$ is therefore identical to Hybrid $(1, i, j+1)$, while if $b = 1$, its view is identical to Hybrid $(1, i, j)$. Therefore, if $\mathscr{B}$ outputs 0 when $\mathscr{A}$ outputs "Hybrid $(1, i, j+1)$" and 1 when $\mathscr{A}$ outputs "Hybrid $(1, i, j)$," it succeeds with the same probability as $\mathscr{A}$. Because i is bounded by the number of prove queries and j is bounded by the parameter n, the number of hybrids is polynomial in the security parameter, and $\mathscr{B}$'s summed advantage over all of the hybrids is negligible, completing the proof that $\Pi_R^{\texttt{aFis}}$ must be adNIM-SHVZK.

We now use the fact that $\Pi_R^{\texttt{aFis}}$ is adNIM-SHVZK to argue that $\Pi_R^{\texttt{aFis}}$ must also be adaptive non-interactive special simulation-sound (adNI-SSS). Consider a new reduction $\mathscr{B}$ that uses an adversary $\mathscr{A}$ that can win the adNI-SSS game (Definition 11), as well as a second adversary $\mathscr{A}'$ that can win the regular non-interactive special soundness (NI-SS) game (a weaker version of adNI-SSS where $\mathscr{A}$ cannot query the simulator; see Appendix A.9 for details) with non-negligible advantage as black boxes to win the adNIM-SHVZK game (Definition 10). $\mathscr{B}$ forwards all of $\mathscr{A}$'s $\texttt{Prove}$ queries to and from the adNIM-SHVZK challenger, and all of $\mathscr{A}$'s random oracle queries to and from H. Similarly, $\mathscr{B}$ forwards all of $\mathscr{A}'$'s oracle queries to and from H. Whenever $\mathscr{A}$ produces a fresh

(non-simulated) proof (x, Φ) such that $\Pi_R^{\texttt{aFis}}.\texttt{Verify}(x, \Phi) = 1$, $\mathscr{B}$ computes $w \leftarrow \Pi_R^{\texttt{aFis}}.\texttt{Extract}(x, \Phi, \mathcal{Q}_{\mathscr{A}})$ and checks whether $R(x, w) = 0$, which happens with non-negligible probability by assumption. $\mathscr{B}$ does the same for $\mathscr{A}''$'s proofs (x', Φ'). (The argument that there must be sufficient queries in $\mathcal{Q}_{\mathscr{A}}$ or $\mathcal{Q}_{\mathscr{A}'}$ to invoke $\Pi_R^{\texttt{aFis}}.\texttt{Extract}$ in the first place is identical to the arguments given by Kondi and shelat [41]; in brief, the strong special soundness property stops $\mathscr{A}$ from being able to "tweak" an old proof to create a forgery.)

If the adNIM-SVHZK challenger is playing with bit $b = 1$ and the proofs and randomness being passed to $\mathscr{A}$ are according to $\Pi_R^{\texttt{aFis}}.\texttt{SimProve}$ and $\Pi_R^{\texttt{aFis}}.\texttt{SimRand}$, this is exactly what $\mathscr{A}$ expects from the adNI-SSS game, and $\mathscr{B}$'s advantage is the same as $\mathscr{A}$'s. If the adNIM-SHVZK challenger is playing with bit $b = 0$, then $\mathscr{A}$'s advantage reduces to $\mathscr{A}''$'s advantage in the standard NI-SS game (since $\mathscr{A}'$ can always generate honest proofs and randomness according to $\Pi_R^{\texttt{aFis}}.\texttt{Prove}$ itself). Assume for a contradiction that there is a non-negligible difference between the extraction failures produced by $\mathscr{A}$ when $b = 1$ and $\mathscr{A}/\mathscr{A}'$ when $b = 0$. If $\mathscr{B}$ observes a difference in output between $\mathscr{A}$ and $\mathscr{A}'$, $\mathscr{B}$ knows $\mathscr{A}$ has an extra advantage due to seeing simulated proofs and randomness and outputs 1; otherwise if there is no difference, $\mathscr{B}$ outputs 0. Therefore, the difference must be negligible: as long as $\Pi_R^{\texttt{aFis}}$ is adNIM-SHVZK (as proven in the preceding paragraphs) and NI-SS (which follows directly from the special soundness of $\Sigma_{R,\tau}$ and the randomness of H), $\Pi_R^{\texttt{aFis}}$ must be adNI-SSS. □

5 Constructing Adaptive, Efficient UC NIZKPoK

In this section, we show that the adaptive non-interactive multiple special honest-verifier zero-knowledge (adNIM-SHVZK) and adaptive non-interactive special simulation-soundness (adNI-SSS) properties afforded by an adaptive SLC are *necessary* to achieve adaptive UC NIZKPoK in any global ROM, and that they are *sufficient* to transform any Σ-protocol into an adaptive UC NIZKPoK in the $\mathcal{G}_{\texttt{rpoRO}}$-hybrid model. For our construction in the $\mathcal{G}_{\texttt{roRO}}$-$\mathcal{F}_{\texttt{CRS}}$-hybrid model, we adapt Lysyanskaya and Rosenbloom's OR-protocol compiler [42].

5.1 The Adaptive NIZKPoK Ideal Functionality

Recall from the introduction that ideal functionalities $\mathcal{F}$ in the UC model operate on behalf of the honest "dummy parties" in the ideal-world experiment [19]. Upon receiving instructions from the ideal adversary $\mathcal{S}$ and setting up any necessary parameters, $\mathcal{F}_{\texttt{aNIZK}}$ proves statements for honest parties using the $\texttt{SimSetup}$ and $\texttt{SimProve}$ algorithms (which do not take witnesses as input) and verifies proofs using the $\texttt{Extract}$ algorithm. If the algorithms from $\mathcal{S}$ do not function as $\mathcal{F}_{\texttt{aNIZK}}$ intends, for instance if $\texttt{SimProve}$ produces proofs that do not verify or $\texttt{Extract}$ outputs invalid witnesses, $\mathcal{F}_{\texttt{aNIZK}}$ outputs $\texttt{Fail}$. During its processing, $\mathcal{F}_{\texttt{aNIZK}}$ stores information about the proofs it has computed on each dummy party's behalf. $\mathcal{S}$ retrieves this information via $\mathcal{F}_{\texttt{aNIZK}}$'s **Corrupt** interface whenever an honest party is corrupted by the environment. The style of our definition follows the standard introduced by Chase and Lysyanskaya [24].

Definition 12 (Adaptive NIZKPoK Ideal Functionality). *The ideal functionality* $\mathcal{F}_{\texttt{aNIZK}}$ *of an* adaptive non-interactive zero-knowledge proof of knowledge (adaptive NIZKPoK) *for a particular session* s *is defined as follows, where all outputs are returned to the calling party.*

Setup: *Initialize an empty list* C *of corrupted parties. Upon receiving a request* $(\texttt{Setup}, P)$ *from a party* $P = (\texttt{pid}, \texttt{sid})$*, check whether* $\texttt{sid} = s$ *and* $P \notin C$*. If either check fails, output* $\perp$*. Otherwise, if this is the first time a request* $(\texttt{Setup}, P)$ *was received from an uncorrupted party with* $\texttt{sid} = s$*, do as follows: pass* $(\texttt{Setup}, s)$ *to the ideal adversary* $\mathcal{S}$*, receive and store* $(\texttt{Algorithms}, \texttt{Setup}, \texttt{Prove}, \texttt{Verify}, \texttt{SimSetup}, \texttt{SimProve}, \texttt{Extract})$*. Then run* `SimSetup` *and store* $(\texttt{ppm}, z_{\texttt{Sim}})$ *where* `ppm` *are public parameters and* $z_{\texttt{Sim}}$ *is any auxiliary output of* `SimSetup`*, and output* `Success`*. Otherwise, output* $\perp$*.*

Prove: *Upon receiving a request* $(\texttt{Prove}, x, w)$ *from a party* $P = (\texttt{pid}, \texttt{sid})$*, check that* $\texttt{sid} = s$*,* $P \notin C$*, and* $R(x, w) = 1$*. If any check fails, output* $\perp$*. Otherwise, set* w *aside, compute* π *according to the* `SimProve` *algorithm, and check that* $\texttt{Verify}(x, \pi) = 1$*. If it doesn't, output* `Fail`*. Otherwise, record the tuple* $(\texttt{Proof}, P, x, w, \pi, z_{\texttt{Sim}}, z_\pi)$*, where* z_π *is any auxiliary output of the* `SimProve` *algorithm. Output* $(\texttt{Proof}, P, x, \pi)$*.*

Verify: *Upon receiving a request* $(\texttt{Verify}, x, \pi)$ *from a party* $P = (\texttt{pid}, \texttt{sid})$*, first check that* $\texttt{sid} = s$ *and* $P \notin C$*. If either check fails, output* $\perp$*. If there exists a tuple* $(\texttt{Proof}, P, x, w, \pi, z_{\texttt{Sim}}, z_\pi)$ *already stored, output* $(\texttt{Verification}, P, x, \pi, 1)$*. Else if* $\texttt{Verify}(x, \pi) = 0$*, output* $(\texttt{Verification}, P, x, \pi, 0)$*. Otherwise, compute* w *according to the* `Extract` *algorithm. If* $R(x, w) = 1$*, output* $(\texttt{Verification}, P, x, \pi, 1)$ *for a successful extraction. Else if* $R(x, w) = 0$*, output* `Fail`*.*

Corrupt: *Upon receiving a request* $(\textbf{Corrupt}, P)$ *from* $\mathcal{S}$*, add* P *to* C *and return all of the stored tuples* $(\texttt{Proof}, P, *)$*, if they exist. Otherwise, output* $\perp$*.*

5.2 Adaptive UC NIZKPoK are adNIM-SHVZK and adNI-SSS

We begin by showing that any adaptive UC NIZKPoK must be adaptive NIM-SHVZK and adaptive NI-SSS. Because this result holds for any choice of global RO with the minimal `Query` functionality (as described in Section 2), we use the generic $\mathcal{G}_{\texttt{RO}}$ to represent the global RO. See Appendix A.10 for the proof of:

Theorem 6 (Required Properties for aUC NIZKPoK). *Let* Π *be a protocol that aUC-realizes* $\mathcal{F}_{\texttt{aNIZK}}$ *in the* $\mathcal{G}_{\texttt{RO}}$*-hybrid model (Definition 1 where* $\mathcal{G}$ *is replaced with* $\mathcal{G}_{\texttt{RO}}$*) with adaptive corruptions. Then* Π *must be overwhelmingly complete (Definition 9), adaptive NIM-SHVZK (Definition 10), and adaptive NI-SSS (Definition 11).*

5.3 Adaptive UC NIZKPoK in the Programmable Global ROM

Next, we show that running any adaptive Σ-protocol Σ_R for relation R through any adaptive straight-line compiler (adaptive SLC) results in adaptive UC NIZKPoK for relation R in the (programmable) $\mathcal{G}_{\texttt{rpoRO}}$-hybrid model. Our proof techniques are covered in Theorem 8; we therefore leave details to Appendix A.11.

Theorem 7 (aUC NIZKPoK in the $\mathcal{G}_{\texttt{rpoRO}}$-hybrid Model). *Let Σ_R be any adaptive Σ-protocol for relation R (Definition 2), $\mathcal{G}_{\texttt{rpoRO}}$ be the restricted programmable observable global random oracle (Appendix A.2) and* SLC *be any adaptive straight-line compiler (Definition 8). Then the NISLE proof system $\Pi_R^{\texttt{SLC}} \leftarrow \texttt{SLC}(\Sigma_R)$ aUC-realizes $\mathcal{F}_{\texttt{aNIZK}}$ in the $\mathcal{G}_{\texttt{rpoRO}}$-hybrid model (Definition 1 where $\mathcal{G}$ is replaced with $\mathcal{G}_{\texttt{rpoRO}}$).*

5.4 Adaptive UC NIZKPoK in the Global RO-CRS Hybrid Model

Our final theorem shows that any adaptive SLC in conjunction with our adaptive OR-protocol compiler is sufficient to create adaptive UC NIZKPoK in the $\mathcal{G}_{\texttt{rpoRO}}$-$\mathcal{F}_{\texttt{CRS}}$-hybrid model. Recall from Section 2 that $\mathcal{F}_{\texttt{CRS}}$ must be parameterized over a samplable-hard, adaptive Σ-friendly relation S.

Construction 2 (Candidate Compiler). Let Σ_R be any adaptive Σ-protocol for relation R (Definition 2), $\mathcal{G}_{\texttt{roRO}}$ be the restricted observable global random oracle, Σ_S be an efficient Σ-protocol for samplable-hard relation S, $\mathcal{F}_{\texttt{CRS}}$ be the ideal CRS functionality (Appendix A.3) where $\texttt{GenCRS} := \kappa_S$, and aSLC be any adaptive straight-line compiler (Definition 8). Then our *candidate compiler* auc is an algorithm that, on input Σ_R and aSLC, produces a tuple of algorithms $\Pi_{\texttt{R}\vee\texttt{S}}^{\texttt{auc}} = (\texttt{Setup}^{\mathcal{G}_{\texttt{roRO}}}, \texttt{Prove}^{\mathcal{G}_{\texttt{roRO}},\mathcal{F}_{\texttt{CRS}}}, \texttt{Verify}^{\mathcal{G}_{\texttt{roRO}},\mathcal{F}_{\texttt{CRS}}}, \texttt{SimSetup}, \texttt{SimProve}, \texttt{SimRand}, \texttt{Extract})$, defined in Figure 6.

Theorem 8 (aUC NIZKPoK in the $\mathcal{G}_{\texttt{roRO}}$-$\mathcal{F}_{\texttt{CRS}}$-hybrid Model). *Let Σ_R be any adaptive Σ-protocol for relation R (Definition 2) with randomness equivocability (Definition 4), $\mathcal{G}_{\texttt{roRO}}$ be the restricted observable global random oracle (Appendix A.2), Σ_S be an efficient Σ-protocol for adaptive Σ-friendly relation S (Appendix A.3), $\mathcal{F}_{\texttt{CRS}}$ be the ideal CRS functionality (Appendix A.3),* aSLC *be any adaptive straight-line compiler (Definition 8), and* auc *be the candidate compiler (Construction 2). Then $\Pi_{\texttt{R}\vee\texttt{S}}^{\texttt{auc}} \leftarrow \texttt{auc}(\Sigma_R, \texttt{aSLC})$ aUC-realizes $\mathcal{F}_{\texttt{aNIZK}}$ in the $\mathcal{G}_{\texttt{roRO}}$-$\mathcal{F}_{\texttt{CRS}}$-hybrid model (Definition 1 where $\mathcal{G} := \mathcal{G}_{\texttt{roRO}}$ and $* := \mathcal{F}_{\texttt{CRS}}$).*

Proof. We begin with the construction of the ideal adversary. When $\mathcal{S}$ receives $(\texttt{Setup}, s)$ from $\mathcal{F}_{\texttt{aNIZK}}$, $\mathcal{S}$ returns the tuple of algorithms $\Pi_{\texttt{R}\vee\texttt{S}}^{\texttt{auc}}$. When $\mathcal{Z}$ issues a query $(\textbf{Corrupt}, P)$ for a party $P = (\texttt{pid}, \texttt{sid})$, $\mathcal{S}$ sends $(\textbf{Corrupt}, P)$ to $\mathcal{F}_{\texttt{aNIZK}}$ to obtain the stored tuples $(\texttt{Proof}, P, x_1, w_1, \pi_1, z_{\texttt{Sim}}, z_1), \ldots, (\texttt{Proof}, P, x_q, w_q, \pi_q, z_{\texttt{Sim}}, z_q)$ corresponding to each query $(\texttt{Prove}, P, x_i, w_i)$ that $\mathcal{Z}$ issued to P (who forwarded them to $\mathcal{F}_{\texttt{aNIZK}}$). $\mathcal{S}$ then runs $r_i \leftarrow \Pi_{\texttt{R}\vee\texttt{S}}^{\texttt{auc}}.\texttt{SimRand}(z_s, z_{\pi_i}, x, \pi, w)$ to obtain simulated randomness r_i. To reconstruct the prover's random tape R, $\mathcal{S}$ concatenates $R = r_1 || \ldots || r_q$ and returns R to $\mathcal{Z}$. Otherwise, $\mathcal{S}$ acts as a "dummy adversary" [19], forwarding all communications to and from $\mathcal{Z}$.

Our proof of UC-security proceeds via a series of hybrids, starting in the real-world experiment and replacing each piece of the real-world adaptive NISLE OR-protocol $\Pi_{\texttt{R}\vee\texttt{S}}^{\texttt{auc}}$ with the functionality of $\mathcal{F}_{\texttt{aNIZK}}$ and $\mathcal{S}$.

Hybrid 1. First, we replace all of the environment's and adversary's connections to the real-world protocol participants with the "challenger" of our reduction, $\mathcal{C}$. This difference is syntactic, so Hybrid 1 is identical to the real-world experiment.

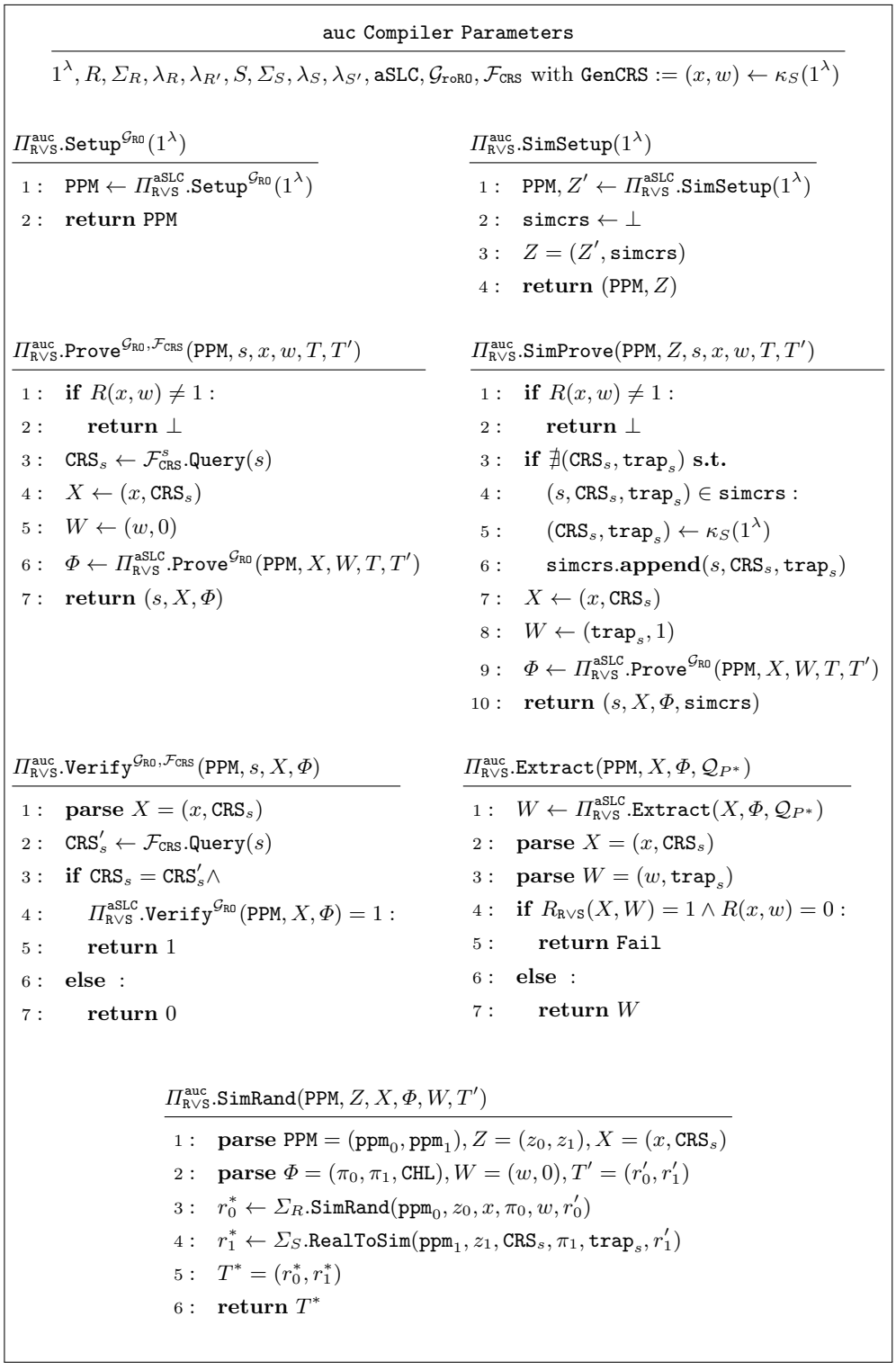


Fig. 6. Compiler $\Pi^{\mathtt{auc}}_{\mathtt{R\vee S}} \leftarrow \mathtt{auc}(\Sigma_R, \mathtt{SLC})$ for Σ_R in the $\mathcal{G}_{\mathtt{roRO}}$-$\mathcal{F}_{\mathtt{CRS}}$-hybrid Model

Hybrid 2. In the second hybrid, we replace $\mathcal{C}$'s real-world `Prove` functionality with the original straight-line compiled OR-protocol simulator algorithms $\Pi^{\mathtt{aSLC}}_{\mathtt{R}\vee\mathtt{S}}.\mathtt{SimSetup}$, $\Pi^{\mathtt{aSLC}}_{\mathtt{R}\vee\mathtt{S}}.\mathtt{SimProve}$, and $\Pi^{\mathtt{aSLC}}_{\mathtt{R}\vee\mathtt{S}}.\mathtt{SimRand}$, and show that Hybrids 1 and 2 are indistinguishable as long as $\Pi^{\mathtt{aSLC}}_{\mathtt{R}\vee\mathtt{S}}$ is adNIM-SHVZK. We bound $\mathcal{Z}$'s probability of distinguishing the hybrids based on the proofs and randomness by constructing a reduction to the adNIM-SHVZK property as follows. Whenever $\mathcal{Z}$ issues a query $(\mathtt{Prove}, P, x_i, w_i)$ to $\mathcal{C}$, $\mathcal{C}$ forwards the query to the adNIM-SHVZK challenger from Figure 4, who returns (Φ_i, T_i). $\mathcal{C}$ forwards Φ_i to $\mathcal{Z}$ and stores (P, Φ_i, T_i) for later. If $\mathcal{Z}$ issues a query $(\mathbf{Corrupt}, P)$, $\mathcal{C}$ retrieves all of the tuples (P, Φ_i, T_i) and sets the random tape T to be the concatenation of all T_i in the order they were stored. It then returns T to $\mathcal{C}$. For $\mathcal{Z}$'s queries $(\mathtt{Verify}, X, \Phi)$, $\mathcal{C}$ returns the result of $\Pi^{\mathtt{aSLC}}_{\mathtt{R}\vee\mathtt{S}}.\mathtt{Verify}(X, \Phi)$. $\mathcal{C}$ then outputs whatever $\mathcal{Z}$ outputs.

Note that if the adNIM-SHVZK challenger is running the 0-bit experiment (using $\Pi^{\mathtt{aSLC}}_{\mathtt{R}\vee\mathtt{S}}.\mathtt{Prove}$ and the prover's randomness), $\mathcal{C}$ simulates $\mathcal{Z}$'s exact view of the experiment in Hybrid 1. Else if the adNIM-SHVZK challenger is running the 1-bit experiment (using $\Pi^{\mathtt{aSLC}}_{\mathtt{R}\vee\mathtt{S}}.\mathtt{SimProve}$ and $\Pi^{\mathtt{SLC}}_{\mathtt{R}\vee\mathtt{S}}.\mathtt{SimRand}$), $\mathcal{C}$ simulates $\mathcal{Z}$'s view of Hybrid 2. Therefore, $\mathcal{C}$ succeeds at winning the adNIM-SHVZK game with the same probability that $\mathcal{Z}$ can distinguish the hybrids, proving Hybrid 1 is computationally indistinguishable to Hybrid 2.

Hybrid 3. In the third hybrid, we replace $\mathcal{C}$'s `Verify` functionality with the **Verify** functionality of $\mathcal{F}_{\mathtt{aNIZK}}$, and show the environment's views are indistinguishable between Hybrids 2 and 3 as long as $\Pi^{\mathtt{SLC}}_{R}$ is adaptive non-interactive special simulation-sound (adaptive NI-SSS) *and* S is a hard relation (i.e. given $\mathtt{CRS}_s$, the probability of finding $\mathtt{trap}_s$ such that $S(\mathtt{CRS}_s, \mathtt{trap}_s) = 1$ is negligible). $\mathcal{Z}$'s adaptive corruption queries are handled identically to those in Hybrid 2 (i.e. $\mathcal{S}$ patches together the prover's random tape by concatenating the outputs of $\Pi^{\mathtt{aSLC}}_{\mathtt{R}\vee\mathtt{S}}.\mathtt{SimRand}$ returned by the adaptive NI-SSS challenger). $\mathcal{C}$ plays the role of the adversary in the adaptive NI-SSS game, forwarding $\mathcal{Z}$'s queries to its challenger. When $\mathcal{Z}$ produces a valid proof that causes $\Pi^{\mathtt{auc}}_{\mathtt{R}\vee\mathtt{S}}.\mathtt{Extract}$ to output `Fail` (which happens with non-negligible probability by assumption, as the failure condition is the only difference between the hybrids), either R is not satisfied and $\mathcal{C}$ wins the adaptive NI-SSS game, or S is not satisfied and $\mathcal{C}$ breaks the hardness property of S.

Hybrid 4. In the fourth hybrid, $\mathcal{C}$ goes back to using $\Pi^{\mathtt{auc}}_{\mathtt{R}\vee\mathtt{S}}.\mathtt{Prove}$ and revealing the real prover's random tape upon corruption. As the only difference between Hybrids 3 and 4 is to reverse the transition between real and simulated proofs made in Hybrid 2, the proof is an almost identical reduction to the adNIM-SHVZK property of $\Pi^{\mathtt{SLC}}_{\mathtt{R}\vee\mathtt{S}}$, with one caveat: we must argue that $\mathcal{Z}$ is not able to use its interactions with the $\Pi^{\mathtt{auc}}_{\mathtt{R}\vee\mathtt{S}}.\mathtt{Extract}$ algorithm to help it distinguish between the real and simulated proofs. Following Lysyanskaya and Rosenbloom [42], we note that since $\Pi^{\mathtt{auc}}_{\mathtt{R}\vee\mathtt{S}}.\mathtt{Extract}$ operates using only information that $\mathcal{Z}$ already knows (its externalized proofs and RO queries), it cannot possibly gain any new insight from interacting with the extractor. Therefore, the only way for $\mathcal{Z}$ to distinguish between Hybrids 3 and 4 is to break the adNIM-SHVZK prop-

erty, which follows from the same reduction used to prove indistinguishability between Hybrids 1 and 2.

Hybrid 5. In the penultimate hybrid, $\mathcal{C}$ replaces the proof process with $\Pi^{\mathtt{auc}}_{\mathtt{R}\vee\mathtt{S}}.\mathtt{SimSetup}$, $\Pi^{\mathtt{auc}}_{\mathtt{R}\vee\mathtt{S}}.\mathtt{SimProve}$, and $\Pi^{\mathtt{auc}}_{\mathtt{R}\vee\mathtt{S}}.\mathtt{SimRand}$. Recall that $\Pi^{\mathtt{auc}}_{\mathtt{R}\vee\mathtt{S}}.\mathtt{SimProve}$ is essentially $\Pi^{\mathtt{auc}}_{\mathtt{R}\vee\mathtt{S}}.\mathtt{Prove}$, except that $\mathcal{C}$ generates and stores pairs $(\mathtt{CRS}_s, \mathtt{trap}_s)$ for each protocol session s in the list $\mathtt{simcrs}$, and uses the witness $\mathtt{trap}_s$ as input to $\Pi^{\mathtt{auc}}_{\mathtt{R}\vee\mathtt{S}}.\mathtt{Prove}$ rather than a "real" witness w. We show that Hybrid 4 is indistinguishable from Hybrid 5 as long as Σ_{OR} is adaptive witness indistinguishable (adWI) according to Theorem 1. Note that since Σ_R is given to be adSHVZK and randomness equivocable in the theorem statement and that Σ_S is an adaptive Σ-friendly relation and therefore also randomness equivocable and adSHVZK, adaptive witness indistinguishability of Σ_{OR} follows from Theorem 1. Since the adWI game captures only a single $\mathtt{Prove}$ query, we consider a hybrid experiment in which the first i queries (and corresponding sections of the prover's random tape) are answered according to Hybrid 4, and the $i+1^{st}$ query onwards are answered according to Hybrid 5. Our reduction $\mathscr{B}$ will use an environment $\mathcal{Z}$ that can distinguish between the i and $i+1^{st}$ hybrids with non-negligible advantage to win the adWI game from Figure 3 with non-negligible advantage as follows.

$\mathscr{B}$ answers $\mathcal{Z}$'s RO queries as usual. For up to and including the i^{th} $\mathtt{Prove}$ query that $\mathcal{Z}$ issues for a party P, $\mathscr{B}$ computes and returns proofs using $\Pi^{\mathtt{auc}}_{\mathtt{R}\vee\mathtt{S}}.\mathtt{Prove}$, exactly as $\mathcal{Z}$ expects from Hybrid 4; past the $i+1^{st}$ query, $\mathscr{B}$ computes and returns proofs using $\Pi^{\mathtt{auc}}_{\mathtt{R}\vee\mathtt{S}}.\mathtt{SimProve}$, exactly as $\mathcal{Z}$ expects from Hybrid 5. $\mathscr{B}$ stores all tuples (P, Φ_i, T_i). When $\mathcal{Z}$ issues the $i{+}1^{st}$ query $(\mathtt{Prove}, P, s, x_{i+1}, w_{i+1})$ for some party P in session s, $\mathscr{B}$ constructs a query $(\mathtt{Prove}, x_{i+1}, \mathtt{CRS}_s, w_{i+1}, \mathtt{trap}_s, 0, \mathtt{st})$ and forwards it to the adWI challenger, who returns a response $(\Phi, T, \mathtt{st})$. $\mathscr{B}$ stores the tuple $(P, \Phi_{i+1}, T_{i+1}) = (P, \Phi, T)$, programs the RO to agree with Φ, then returns Φ to $\mathcal{Z}$. If $\mathcal{Z}$ issues a query $(\mathbf{Corrupt}, P)$, $\mathscr{B}$ gathers all (P, Φ_i, T_i) and returns the concatenation of all T_i to $\mathcal{Z}$. If $\mathcal{Z}$ outputs "Hybrid 3" then $\mathscr{B}$ outputs 0; otherwise if $\mathcal{Z}$ outputs "Hybrid 4" then $\mathscr{B}$ outputs 1.

Note that if the adWI challenger is running the 0-bit experiment, it computed the proof Φ_{i+1} using the real witness w_{i+1} for x_{i+1} using tape T_{i+1}, which is exactly what $\mathcal{Z}$ expects from Hybrid $i+1$. Otherwise if the adWI challenger is running the 1-bit experiment then it computed the proofs using the opposite witness $\mathtt{trap}_s$ for statement $\mathtt{CRS}_s$, then "equivocated" the random tape to look like it used w_{i+1}, which is exactly what $\mathcal{Z}$ expects from Hybrid i. Therefore, $\mathscr{B}$ succeeds in winning the adWI game with the same probability as $\mathcal{Z}$ succeeds in distinguishing Hybrids i and $i+1$. Since the number of $\mathtt{Prove}$ queries must be polynomial in the security parameter, $\mathscr{B}$'s total advantage in winning the adWI game over all of the proofs is still non-negligible, completing the proof.

Hybrid 6. The final hybrid replaces $\mathcal{C}$ with $\mathcal{F}_{\mathtt{aNIZK}}$ and $\mathcal{S}$. Note that since $\mathcal{C}$ now runs all of $\mathcal{F}_{\mathtt{aNIZK}}$ and $\mathcal{S}$'s procedures, this is again a syntactic difference, and Hybrid 5 is identical to Hybrid 6. □

Acknowledgments

Many thanks to Yashvanth Kondi for ongoing fruitful conversations about the properties of straight-line extractors, and to Ran Cohen for pointing our attention to the universal composability with global subroutines model. This research was supported by NSF Grants 2312241, 2154170, and 2247305.

References

1. Christian Badertscher, Ran Canetti, Julia Hesse, Björn Tackmann, and Vassilis Zikas. Universal composition with global subroutines: Capturing global setup within plain uc. In *Theory of Cryptography Conference*, pages 1–30. Springer, 2020.
2. Foteini Baldimtsi, Jan Camenisch, Maria Dubovitskaya, Anna Lysyanskaya, Leonid Reyzin, Kai Samelin, and Sophia Yakoubov. Accumulators with applications to anonymity-preserving revocation. In *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 301–315. IEEE, 2017.
3. Foteini Baldimtsi and Anna Lysyanskaya. Anonymous credentials light. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 1087–1098, 2013.
4. Mihir Bellare, Elizabeth Crites, Chelsea Komlo, Mary Maller, S Tessaro, and C Zhu. Better than advertised security for non-interactive threshold signatures. In *Advances in Cryptology-CRYPTO*, 2022.
5. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 62–73, 1993.
6. Fabrice Benhamouda, Tancrède Lepoint, Julian Loss, Michele Orrù, and Mariana Raykova. On the (in) security of ros. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 33–53. Springer, 2021.
7. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *CRYPTO 2004*, pages 41–55. Springer, 2004.
8. Jan Camenisch and Ivan Damgård. Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 331–345. Springer, 2000.
9. Jan Camenisch, Manu Drijvers, Tommaso Gagliardoni, Anja Lehmann, and Gregory Neven. The wonderful world of global random oracles. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 280–312. Springer, 2018.
10. Jan Camenisch, Susan Hohenberger, Markulf Kohlweiss, Anna Lysyanskaya, and Mira Meyerovich. How to win the clonewars: efficient periodic n-times anonymous authentication. pages 201–210. ACM, 2006.
11. Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Compact E-cash. In Ronald Cramer, editor, *Advances in Cryptology — Eurocrypt 2005*, volume 3494, pages 302–321. Springer, 2005.
12. Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045, pages 93–118. Springer Verlag, 2001.

13. Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *CRYPTO*, pages 61–76. Springer, 2002.
14. Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In *SCN 2002*, volume 2576, pages 268–289, 2003.
15. Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *CRYPTO 2004*, volume 3152, pages 56–72, 2004.
16. Jan Camenisch and Victor Shoup. Practical verifiable encryption and decryption of discrete logarithms. In *CRYPTO 2003*, pages 126–144. Springer, 2003.
17. Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups. In *CRYPTO 1997*, pages 410–424. Springer, 1997.
18. Jan Camenisch and Markus Stadler. Proof systems for general statements about discrete logarithms. *Technical Report/ETH Zurich, Department of Computer Science*, 260, 1997.
19. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 136–145. IEEE, 2001.
20. Ran Canetti, Yevgeniy Dodis, Rafael Pass, and Shabsi Walfish. Universally composable security with global setup. In *Theory of Cryptography Conference*, pages 61–85. Springer, 2007.
21. Ran Canetti and Marc Fischlin. Universally composable commitments. In *CRYPTO 2001*, pages 19–40. Springer, 2001.
22. Ran Canetti, Abhishek Jain, and Alessandra Scafuro. Practical uc security with a global random oracle. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 597–608, 2014.
23. Ran Canetti, Pratik Sarkar, and Xiao Wang. Triply adaptive uc nizk. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 466–495. Springer, 2022.
24. Melissa Chase and Anna Lysyanskaya. On signatures of knowledge. In *CRYPTO 2006*, pages 78–96. Springer, 2006.
25. Alessandro Chiesa and Giacomo Fenzi. zksnarks in the rom with unconditional uc-security. *Cryptology ePrint Archive*, 2024.
26. Ran Cohen, Abhi Shelat, and Daniel Wichs. Adaptively secure mpc with sublinear communication complexity. In *Advances in Cryptology–CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019, Proceedings, Part II 39*, pages 30–60. Springer, 2019.
27. D Connolly, C Komlo, I Goldberg, and CA Wood. Rfc 9591: The flexible round-optimized schnorr threshold (frost) protocol for two-round schnorr signatures, 2024.
28. Henry Corrigan-Gibbs, Dan Boneh, and David Mazières. Riposte: An anonymous messaging system handling millions of users. In *2015 IEEE Symposium on Security and Privacy*, pages 321–338. IEEE, 2015.
29. Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO 1994*, pages 174–187. Springer, 1994.
30. Elizabeth Crites, Chelsea Komlo, and Mary Maller. How to prove schnorr assuming schnorr: Security of multi-and threshold signatures. *ePrint Archive*, 2021.
31. Elizabeth Crites, Chelsea Komlo, and Mary Maller. Fully adaptive schnorr threshold signatures. In *Annual International Cryptology Conference*, pages 678–709. Springer, 2023.
32. Ivan Damgård. On σ-protocols. University of Aarhus, Department of Computer Science, 2002.

33. Manu Drijvers, Kasra Edalatnejad, Bryan Ford, Eike Kiltz, Julian Loss, Gregory Neven, and Igors Stepanovs. On the security of two-round multi-signatures. In *2019 IEEE Symposium on Security and Privacy*, pages 1084–1101. IEEE, 2019.
34. Marc Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. 2005. Manuscript. Available from http://www.cryptoplexity.informatik.tu-darmstadt.de/media/crypt/publications_1/fischlinonline-extractor2005.pdf.
35. Marc Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In *Annual International Cryptology Conference*, pages 152–168. Springer, 2005.
36. Juan A Garay, Philip MacKenzie, and Ke Yang. Strengthening zero-knowledge protocols using signatures. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 177–194. Springer, 2003.
37. Jens Groth, Rafail Ostrovsky, and Amit Sahai. New techniques for noninteractive zero-knowledge. *Journal of the ACM (JACM)*, 59(3):1–35, 2012.
38. Eduard Hauck, Eike Kiltz, and Julian Loss. A modular treatment of blind signatures from identification schemes. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 345–375. Springer, 2019.
39. Markulf Kohlweiss, Anna Lysyanskaya, and An Nguyen. Privacy-preserving blueprints. In *EUROCRYPT 2023*, volume 14005 of *Lecture Notes in Computer Science*, pages 594–625. Springer, 2023.
40. Chelsea Komlo and Ian Goldberg. Frost: flexible round-optimized schnorr threshold signatures. In *Selected Areas in Cryptography: 27th International Conference, Halifax, NS, Canada (Virtual Event), October 21-23, 2020, Revised Selected Papers 27*, pages 34–65. Springer, 2021.
41. Yashvanth Kondi and Abhi Shelat. Improved straight-line extraction in the random oracle model with applications to signature aggregation. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 279–309. Springer, 2022.
42. Anna Lysyanskaya and Leah Namisa Rosenbloom. Universally composable σ-protocols in the global random-oracle model. In *Theory of Cryptography Conference*, pages 203–233. Springer, 2022.
43. C Andrew Neff. A verifiable secret shuffle and its application to e-voting. In *Proceedings of the 8th ACM conference on Computer and Communications Security*, pages 116–125, 2001.
44. Jonas Nick, Tim Ruffing, and Yannick Seurin. Musig2: simple two-round schnorr multi-signatures. In *CRYPTO 2021*, pages 189–221. Springer, 2021.
45. Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In *CRYPTO 1992*, pages 31–53. Springer, 1992.
46. Somnath Panja and Bimal Roy. A secure end-to-end verifiable e-voting system using blockchain and cloud server. *Journal of Information Security and Applications*, 59:102815, 2021.
47. Rafael Pass. On deniability in the common reference string and random oracle model. In *CRYPTO 2003*, pages 316–337, 2003.
48. Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of cryptology*, 4(3):161–174, 1991.
49. Berry Schoenmakers. A simple publicly verifiable secret sharing scheme and its application to electronic voting. In *CRYPTO 1999*, pages 148–164. Springer, 1999.
50. Proposal: Σ-protocols. Stephan krenn and michele orrù, 2021.

51. Stefano Tessaro and Chenzhi Zhu. Revisiting BBS signatures. In *EUROCRYPT 2023*, volume 14008 of *Lecture Notes in Computer Science*, pages 691–721. Springer, 2023.
52. Dominique Unruh. Non-interactive zero-knowledge proofs in the quantum random oracle model. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 755–784. Springer, 2015.
53. David Wagner. A generalized birthday problem. In *CRYPTO 2002*, pages 288–304. Springer, 2002.
54. Douglas Wikström. A commitment-consistent proof of a shuffle. In Colin Boyd and Juan Manuel González Nieto, editors, *ACISP*, pages 407–421. Springer, 2009.

A Appendix

A.1 Notation

We use λ for the security parameter, and say an algorithm $\mathscr{A}$ is efficient in λ if its runtime can be expressed as a polynomial $\mathsf{poly}(\lambda)$ on input λ. We say a function negl is negligible in λ if for every positive polynomial p there exists a threshold N such that for all $\lambda > N$, $\mathsf{negl}(\lambda) < \frac{1}{p(\lambda)}$.

When we write $y \leftarrow z$ where z is a quantity, we mean that y is assigned the value z. Similarly, $y \leftarrow \mathscr{A}(x)$ means that y is assigned the output of algorithm $\mathscr{A}$ on input x. We write $\perp \leftarrow \mathscr{A}(x)$ to indicate that $\mathscr{A}$ has halted on input x with no output, such that any process that invoked $\mathscr{A}$ can resume. By $y \leftarrow_{\$} Z$ where Z is a set or a probability distribution, we mean that y is assigned an element sampled uniformly at random from Z.

If two distributions Y and Z are equivalent, we use the notation $Y = Z$. If Y and Z are statistically indistinguishable, we use the notation $Y \approx_s Z$. If Y and Z are only computationally indistinguishable, we use the notation $Y \approx_c Z$. When we say two distributions are statistically (resp. computationally) indistinguishable, we mean that for all λ, the probability that any algorithm $\mathscr{A}$ (resp. PPT algorithm $\mathscr{A}$) can determine whether a mystery element x was sampled from Y or Z is only negligibly greater than a random guess, or $\frac{1}{2} + \mathsf{negl}(\lambda)$. We might also say in this case that $\mathscr{A}$ distinguishes Y from Z with negligible advantage over a random guess.

A.2 Global Random Oracles

Definition 13 (Observable Global RO $\mathcal{G}_{\mathtt{roRO}}$). [22,9] *The* observable global RO $\mathcal{G}_{\mathtt{roRO}}$ *is a tuple of algorithms* $(\mathtt{Query}, \mathtt{Observe})$ *defined over an output length* ℓ *and an initially empty list of queries* $\mathcal{Q}$*:*

- $v \leftarrow \mathtt{Query}(x)$: *Parse* x *as* (s, x') *where* s *is an SID. If a list* $\mathcal{Q}_s$ *of illegitimate queries for* s *does not yet exist, set* $\mathcal{Q}_s = \perp$*. If the caller's SID* $\neq s$*, add* (x, v) *to* $\mathcal{Q}_s$*. If there already exists a pair* (x, v) *in the query list* $\mathcal{Q}$*, return* v*. Otherwise, choose* v *uniformly at random from* $\{0,1\}^{\ell}$*, store the pair* (x, v) *in* $\mathcal{Q}$*, and return* v*.*
- $\mathcal{Q}_s \leftarrow \mathtt{Observe}(s)$: *If a list* $\mathcal{Q}_s$ *of illegitimate queries for* s *does not yet exist, set* $\mathcal{Q}_s = \perp$*. Return* $\mathcal{Q}_s$*.*

Definition 14 (Restricted Programmable Observable Global RO $\mathcal{G}_{\mathtt{rpoRO}}$). [9] *The* restricted programmable observable global random oracle $\mathcal{G}_{\mathtt{rpoRO}}$ *is a tuple of algorithms* $(\mathtt{Query}, \mathtt{Observe}, \mathtt{Program}, \mathtt{IsProgrammed})$ *defined over an output length* ℓ *and initially empty lists* $\mathcal{Q}$ *(queries) and* $\mathtt{prog}$ *(programmed queries):*

- $v \leftarrow \mathtt{Query}(x)$: *Same as Definition 13 above.*
- $\mathcal{Q}_s \leftarrow \mathtt{Observe}(s)$: *Same as Definition 13 above.*
- $\{0,1\} \leftarrow \mathtt{Program}(x, v)$: *If* $\exists v' \in \{0,1\}^{\ell}$ *such that* $(x, v') \in \mathcal{Q}$ *and* $v \neq v'$*, output* 0*. Otherwise, add* (x, v) *to* $\mathcal{Q}$ *and* $\mathtt{prog}$ *and output* 1*.*
- $\{0,1\} \leftarrow \mathtt{IsProgrammed}(x)$: *Parse* x *as* (s, x')*. If the caller's SID* $\neq s$*, output* $\perp$*. Otherwise if* $x \in \mathtt{prog}$*, output* 1*. Otherwise, output* 0*.*

A.3 $\mathcal{G}_{\mathtt{roRO}}$-$\mathcal{F}_{\mathtt{CRS}}$-hybrid Model Requirements

Definition 15 (Samplable-Hard Relation). [42] *A binary $\mathcal{NP}$ relation S is* samplable-hard *with respect to a security parameter λ if it has the following properties.*

1. **Sampling a statement-witness pair is easy.** *There exists a sampling algorithm κ_S that on input 1^λ outputs (x, w) such that $S(x, w) = 1$ and $|x| = \mathsf{poly}(\lambda)$.*
2. **Computing a witness from a statement is hard.** *For a randomly sampled statement-witness pair $(x, w) \leftarrow \kappa_S(1^\lambda)$ the probability that an efficient adversary $\mathscr{A}$ can find a valid witness given only the statement is negligible. Formally, for all PPT $\mathscr{A}$,*

$$\Pr[(x, w) \leftarrow \kappa_S(1^\lambda), w' \leftarrow \mathscr{A}(1^\lambda, x, \kappa_S) : (x, w') \in R] \leq \mathsf{negl}(\lambda).$$

Definition 16 (Adaptive Σ-Friendly Relation). *An* adaptive Σ-friendly relation *S is a binary $\mathcal{NP}$ relation with a corresponding* efficient, adaptive *(Definition 2), and* witness equivocable *(Definition 4) Σ-protocol Σ_S.*

Definition 17 (CRS Ideal Functionality). [42] *The ideal functionality $\mathcal{F}_{\mathtt{CRS}}$ of a* common reference string (CRS) *for a particular CRS generation mechanism* `GenCRS` *is defined as follows.*

Query: *Upon receiving a request* $(\mathtt{Query}, s)$ *from a party* $P = (\mathtt{pid}, \mathtt{sid})$*, first check whether* $\mathtt{sid} = s$*. If it doesn't, output* $\perp$*. Otherwise, if this is the first time that* $(\mathtt{Query}, s)$ *was received, compute x according to the algorithm* `GenCRS` *and store the tuple* $(\mathtt{CRS}, s, x)$*. Return* $(\mathtt{CRS}, s, x)$*.*

A.4 Adaptive Corruptions in the UC Model

Adaptive corruptions allow the (adversarial) environment $\mathcal{Z}$ in the UC model to obtain full internal views of honest parties *after* they have already participated in a cryptographic protocol. Briefly, adaptive corruptions in the UC framework are modeled as follows. When it wants to corrupt an extant party P, $\mathcal{Z}$ sends a message $(\mathbf{Corrupt}, P)$ to the control function [19].[4]

In the real world, the control function passes the message $(\mathbf{Corrupt}, P)$ to the traditional adversary $\mathscr{A}$, who passes the message to P. Upon receiving the corruption instruction, P relinquishes all of its hidden internal tapes, including its input, output, work, and random tapes, to $\mathscr{A}$. Note that coins from the

[4] The control function is the entity in the UC experiment in charge of passing messages back and forth between all protocol participants. It is easiest to think of the control function as a modeling technique that prevents $\mathcal{Z}$ from sending messages that are outside the scope of the desired security experiment. For example, in the passive corruption model, the control function would not allow a message $(\mathbf{Corrupt}, P)$ to go through after P was initialized. In the adaptive setting, corruption messages are allowed at any point during the security experiment.

random tape in the UC model [19] are defined (without loss of generality) to be read-once and flipped on-the-fly, such that the calling TM can generate as much randomness as it wants (within its polynomial run-time bound). This implies that the corrupted party P will only need to relinquish its random tape *history*, rather than a "full" tape (the length of which is undefined).

In the ideal world, the control function passes the message $(\textbf{Corrupt}, P)$ to the ideal adversary $\mathcal{S}$, who must be able to provide a convincing internal view of P to $\mathcal{Z}$. $\mathcal{S}$ does this by first querying the ideal functionality $\mathcal{F}$ (who has been running computations on "dummy party" P's behalf) for any relevant information about P; it then runs some algorithms (in our case `SimRand`) to simulate P's internal tapes. The control function routes all subsequent instructions for P to $\mathscr{A}$ in the real world, and $\mathcal{S}$ in the ideal world.

A.5 Adaptive Σ-protocol Template

Definition 18 (Adaptive Σ-protocol Template). *The* adaptive Σ-protocol template *for a relation R is a tuple of efficient algorithms $\tau = (\texttt{Setup}, \texttt{Commit}, \texttt{Challenge}, \texttt{Respond}, \texttt{Decision})$, defined as follows.*

- $\texttt{ppm} \leftarrow \texttt{Setup}(1^\lambda)$*: Given a security parameter, generates a set of public parameters* ppm *which minimally include 1^λ, challenge length ℓ, and randomness security parameter λ_P.*
- $\texttt{com} \leftarrow \texttt{Commit}(\texttt{ppm}, x, w, r)$*: Given statement x, witness w, and randomness $r \leftarrow_\$ \{0,1\}^{\lambda_P}$, P sends V a message* com*.*
- $\texttt{chl} \leftarrow \texttt{Challenge}(\texttt{ppm}, x, \texttt{com})$*: V sends P a random ℓ-bit string* chl*.*
- $\texttt{res} \leftarrow \texttt{Respond}(\texttt{ppm}, x, w, \texttt{com}, \texttt{chl})$*: P sends V a reply* res*.*
- $\{0,1\} \leftarrow \texttt{Decision}(\texttt{ppm}, x, \texttt{com}, \texttt{chl}, \texttt{res})$*: V decides whether to output 1 (accept) or 0 (reject) based on the input* $(\texttt{ppm}, x, \texttt{com}, \texttt{chl}, \texttt{res})$*.*

The tuple $(\texttt{com}, \texttt{chl}, \texttt{res})$ *is called a* transcript *or* proof*. We say a transcript or proof is* valid *or* accepting *if* $\texttt{Decision}(\texttt{ppm}, x, \texttt{com}, \texttt{chl}, \texttt{res})$ *outputs 1.*

A.6 Standard Completeness and Two-Special Soundness

Definition 19 (Completeness). *A Σ-protocol Σ_R for relation R is* complete *if for all $(x, w) \in R$ and $\pi \leftarrow \Sigma_R.\texttt{Prove}((x, w), x)$, $\Sigma_R.\texttt{Verify}(x, \pi) = 1$.*

Definition 20 (Two-Special Soundness). *A Σ-protocol Σ_R for relation R is* two-special sound *if there exists a* PPT *algorithm* Extract *such that for any security parameter λ, any* PPT *adversary $\mathscr{A}$,*

$$\Pr[\texttt{Fail} \leftarrow \mathsf{SS}_{\mathscr{A},\Sigma_R}(1^\lambda)] \leq \mathsf{negl}(\lambda),$$

where SS *is the special soundness game described in Figure 7. We say $\mathscr{A}$ wins the* SS *game if* $\Pr[\texttt{Fail} \leftarrow \mathsf{SS}_{\mathscr{A},\Sigma_R}(1^\lambda)] > \mathsf{negl}(\lambda)$.

$\mathsf{SS}_{\mathscr{A},\Sigma_R}(1^\lambda)$

1 : $\mathtt{ppm} \leftarrow \Sigma_R.\mathtt{Setup}(1^\lambda)$
2 : $(x, \pi, \pi') \leftarrow \mathscr{A}(1^\lambda, \mathtt{ppm})$
3 : **parse** $\pi = (\mathtt{com}, \mathtt{chl}, \mathtt{res}), \pi' = (\mathtt{com}', \mathtt{chl}', \mathtt{res}')$
4 : **if** $(\Sigma_R.\mathtt{Verify}(x, \pi) = \Sigma_R.\mathtt{Verify}(x, \pi') = 1 \wedge$
5 : $\quad \mathtt{com} = \mathtt{com}' \wedge \mathtt{chl} \neq \mathtt{chl}')$:
6 : $\quad w \leftarrow \Sigma_R.\mathtt{Extract}(x, \pi, \pi')$
7 : $\quad$ **if** $R(x, w) = 0$:
8 : $\quad\quad$ **return** `Fail`
9 : **return** `Success`

Fig. 7. Two-Special Soundness (SS) Game.

A.7 adSHVZK and RE Imply Adaptive WI Full Proof

Recall Theorem 1: Let $\Sigma_{OR} = (\mathtt{Setup}, \mathtt{Prove}, \mathtt{Verify}, \mathtt{SimSetup}, \mathtt{SimProve}, \mathtt{SimRand}, \mathtt{Extract})$ be an OR-protocol (Definition 5) based on two Σ-protocols Σ_{R_0,τ_0} and Σ_{R_1,τ_1} (Definition 2). If Σ_{OR} is adaptive special honest-verifier zero-knowledge (Definition 3) and randomness equivocable (Definition 4), then Σ_{OR} is adaptive witness indistinguishable (Definition 6).

Proof. We wish to show that as long as Σ_{OR} is adaptive special honest-verifier zero-knowledge (adSHVZK) and randomness equivocable (RE), $\mathscr{A}$'s advantage in distinguishing the 0-bit from 1-bit adaptive witness indistinguishability (adWI) experiments is negligible. Without loss of generality and to simplify the proof, we consider the equivalent formulation of all three definitions in which the statement $\Pr[b' = b] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$ is replaced with $|\Pr[b' = 0 \mid b = 0] - \Pr[b' = 0 \mid b = 1]| \leq \mathsf{negl}(\lambda)$, where b' represents $\mathscr{A}$'s output in either the $b = 0$ or $b = 1$ experiments.

Let the 0-bit adWI experiment be denoted RS for "real-sim," since the adversary's chosen witness w_b is used as input to $\Sigma_{OR}.\mathtt{Prove}$—such that π_b and r_b are real (according to $\Sigma_b.\mathtt{Prove}$ and the prover's random tape) and $\pi_{\overline{b}}$ and $r_{\overline{b}}$ are simulated (according to $\Sigma_{\overline{b}}.\mathtt{SimProve}$ and the simulator's random tape). Let the 1-bit adWI experiment be denoted SR for the above scenario's complement, "sim-real," where the only change is that the random tape is equivocated (according to $\Sigma_b.\mathtt{SimRand}$ and $\Sigma_{\overline{b}}.\mathtt{RealToSim}$). We fix $\mathscr{A}$'s probability of outputting 0 in the RS (resp. SR) experiment to be $p^{\mathscr{A}}_{\mathtt{RS}}(\lambda)$ (resp. $p^{\mathscr{A}}_{\mathtt{SR}}(\lambda)$). We will show via a hybrid experiment SS, or "sim-sim"—in which both halves of the proof are according to `SimProve`—that for all $\mathscr{A}$, as long as Σ_{OR} is adSHVZK and RE, $|p^{\mathscr{A}}_{\mathtt{RS}}(\lambda) - p^{\mathscr{A}}_{\mathtt{SR}}(\lambda)| \leq \mathsf{negl}(\lambda)$.

Lemma 1. *Let the 0-bit experiment of the adWI game (Figure 3) be denoted adWI-RS (for "real-sim"). Consider a hybrid experiment denoted adWI-SS (for*

"sim-sim") in which lines 4-8 of either experiment in the adWI game are replaced as follows. Set $W = (w_b, b)$. *Sample* $r_b \leftarrow_\$ \{0,1\}^{\lambda_{R'_b}}$, $r'_{\overline{b}} \leftarrow_\$ \{0,1\}^{\lambda_{R'_{\overline{b}}}}$, $\mathtt{chl}_0 \leftarrow_\$ \{0,1\}^{\ell_0}$, *and* $\mathtt{chl}_1 \leftarrow_\$ \{0,1\}^{\ell_1}$. *Set* $\mathtt{CHL} = \mathtt{chl}_0 \oplus \mathtt{chl}_1$. *Compute* $\pi_0 \leftarrow \Sigma_0.\mathtt{SimProve}(z_0, x_0, \mathtt{chl}_0, r_0)$ *and* $\pi_1 \leftarrow \Sigma_1.\mathtt{SimProve}(z_1, x_1, \mathtt{chl}_1, r_1)$. *Set* $\Phi = (\pi_0, \pi_1, \mathtt{CHL})$. *Finally, compute* $r_{\overline{b}} \leftarrow \Sigma_{\overline{b}}.\mathtt{SimRand}(\mathtt{ppm}_{\overline{b}}, z_{\overline{b}}, x_{\overline{b}}, \pi_{\overline{b}}, w_{\overline{b}}, r_{\overline{b}'})$ *and set* $T = r_0||r_1$. *Then for all* $\mathscr{A}$, *as long as both* Σ_0 *and* Σ_1 *are adaptive special honest-verifier zero-knowledge (adSHVZK),* $|p^{\mathscr{A}}_{\mathtt{RS}}(\lambda) - p^{\mathscr{A}}_{\mathtt{SS}}(\lambda)| \leq \mathsf{negl}(\lambda)$.

Proof. Assume for a contradiction that $\mathscr{A}$'s fixed 0-output probabilities $p^{\mathscr{A}}_{\mathtt{RS}}(\lambda)$ and $p^{\mathscr{A}}_{\mathtt{SS}}(\lambda)$ are such that $|p^{\mathscr{A}}_{\mathtt{RS}}(\lambda) - p^{\mathscr{A}}_{\mathtt{SS}}(\lambda)| > \mathsf{negl}(\lambda)$. We construct a reduction $\mathscr{B}$ that, given $\mathscr{A}$ as a black-box, outputs 0 with probability $p^{\mathscr{B}}_0$ when the challenge bit $b^* = 0$ and probability $p^{\mathscr{B}}_1$ when the challenge bit $b^* = 1$ in the adSHVZK game such that $|p^{\mathscr{B}}_0 - p^{\mathscr{B}}_1| > \mathsf{negl}(\lambda)$. $\mathscr{B}$ proceeds as follows.

$\mathscr{B}$ forwards $(1^\lambda, \mathtt{PPM})$ from the adSHVZK challenger $\mathcal{C}$ to $\mathscr{A}$, who returns $(x_0, w_0, x_1, w_1, b, \mathtt{st})$. Without loss of generality, let the adSHVZK challenger be parameterized over Σ_b, denoted $\mathcal{C}_b$. $\mathscr{B}$ sets $(x, w) = (x_b, w_b)$ and sends the query $(\mathtt{Prove}, x, w, \mathtt{st}')$ to $\mathcal{C}_b$, who returns (π, r). $\mathscr{B}$ sets $r_b = r$ and $\pi_b = \pi$, where $\pi_b = (\mathtt{com}_b, \mathtt{chl}_b, \mathtt{res}_b)$. It then samples $r_{\overline{b}} \leftarrow_\$ \{0,1\}^{\lambda_{R_{\overline{b}'}}}$ and computes $\pi_{\overline{b}} = \Sigma_{\overline{b}}.\mathtt{SimProve}(x_{\overline{b}}, \mathtt{CHL} \oplus \mathtt{chl}_b, r_{\overline{b}})$. Finally, it sets $\Phi = (\pi_0, \pi_1, \mathtt{CHL})$ and $T = r_0||r_1$, and returns $(\Phi, T, \mathtt{st})$ to $\mathscr{A}$. $\mathscr{B}$ outputs whatever $\mathscr{A}$ outputs.

If $\mathcal{C}_b$ is running on input a challenge bit $b^* = 0$, then the returned proof π will be according to $\Sigma_b.\mathtt{Prove}$ on input the random tape r while the second proof $\pi_{\overline{b}}$ is according to $\Sigma_{\overline{b}}.\mathtt{SimProve}$ and returned with the simulator's random coins, exactly as $\mathscr{A}$ expects from the adWI-RS experiment. Therefore, when $b^* = 0$, $p^{\mathscr{B}}_0 = p^{\mathscr{A}}_{\mathtt{RS}}$. If $\mathcal{C}_b$ is running on input $b^* = 1$, then π will be according to $\Sigma_b.\mathtt{SimProve}$ and the random tape r will be according to $\Sigma_b.\mathtt{SimRand}$. The second proof $\pi_{\overline{b}}$ will also be simulated and returned with the simulator's randomness, exactly as $\mathscr{A}$ expects from the adWI-SS experiment. Therefore, when $b^* = 1$, $p^{\mathscr{B}}_1 = p^{\mathscr{A}}_{\mathtt{SS}}$. Since we have $|p^{\mathscr{A}}_{\mathtt{RS}}(\lambda) - p^{\mathscr{A}}_{\mathtt{SR}}(\lambda)| > \mathsf{negl}(\lambda)$ by assumption, this implies $|p^{\mathscr{B}}_0 - p^{\mathscr{B}}_1| > \mathsf{negl}(\lambda)$, contradicting the adSHVZK property of Σ_b and completing the proof of Lemma 1. □

Lemma 2. *Let the 1-bit experiment of the adWI game (Figure 3) be denoted adWI-SR (for "sim-real") and let the hybrid experiment adWI-SS be as defined in Lemma 1, except this time (without loss of generality), sample* $r'_b \leftarrow_\$ \{0,1\}^{\lambda_{R'_b}}$ *and* $r'_{\overline{b}} \leftarrow_\$ \{0,1\}^{\lambda_{R'_{\overline{b}}}}$, *then compute* $r_b \leftarrow \Sigma_b.\mathtt{SimRand}(\mathtt{ppm}_b, z_b, x_b, \pi_b, w_b, r'_b)$. *For all* $\mathscr{A}$, *as long as both* Σ_0 *and* Σ_1 *are randomness equivocable (RE),* $|p^{\mathscr{A}}_{\mathtt{SR}}(\lambda) - p^{\mathscr{A}}_{\mathtt{SS}}(\lambda)| \leq \mathsf{negl}(\lambda)$.

Proof. Again assume for a contradiction that $\mathscr{A}$'s fixed 0-output probabilities $p^{\mathscr{A}}_{\mathtt{SR}}(\lambda)$ and $p^{\mathscr{A}}_{\mathtt{SS}}(\lambda)$ are such that $|p^{\mathscr{A}}_{\mathtt{SR}}(\lambda) - p^{\mathscr{A}}_{\mathtt{SS}}(\lambda)| > \mathsf{negl}(\lambda)$. We construct a reduction $\mathscr{B}$ that, given $\mathscr{A}$ as a black-box, outputs 0 with probability $p^{\mathscr{B}}_0$ when the challenge bit $b^* = 0$ and probability $p^{\mathscr{B}}_1$ when the challenge bit $b^* = 1$ in the RE game such that $|p^{\mathscr{B}}_0 - p^{\mathscr{B}}_1| > \mathsf{negl}(\lambda)$.

$\mathscr{B}$ proceeds the same as in Lemma 1, except for two differences. First, the challenger $\mathcal{C}_{\overline{b}}$ is for the RE experiment parameterized over $\Sigma_{\overline{b}}$ rather than the ad-

SHVZK experiment parameterized over Σ_b. Second, $\mathscr{B}$ computes π_b rather than $\pi_{\overline{b}}$, this time according to $\Sigma_b.\texttt{SimProve}(z_b, x_b, \texttt{chl}_b, r'_b)$ for $r'_b \leftarrow_\$ \{0,1\}^{\lambda_{R_{b'}}}$, and sets $r_b \leftarrow \Sigma_b.\texttt{SimRand}(z_b, x_b, \pi_b, w_b, r'_b)$. If $\mathcal{C}_{\overline{b}}$ is running on input bit $b^* = 0$, then $\pi_{\overline{b}}$ is according to $\Sigma_{\overline{b}}.\texttt{Prove}$ and $r_{\overline{b}}$ is according to $\Sigma_{\overline{b}}.\texttt{RealToSim}$ while π_b and r_b are according to $\Sigma_b.\texttt{SimProve}$ and $\Sigma_b.\texttt{SimRand}$, respectively, just as $\mathscr{A}$ expects from the adWI-SR experiment. If, on the other hand, $\mathcal{C}_{\overline{b}}$ is running on input $b^* = 1$, then $\pi_{\overline{b}}$ is according to $\Sigma_{\overline{b}}.\texttt{SimProve}$ and $r_{\overline{b}}$ is the simulator's random tape, while π_b and r_b are still according to $\Sigma_b.\texttt{SimProve}$ and $\Sigma_b.\texttt{SimRand}$, respectively, just as $\mathscr{A}$ expects from the adWI-SS experiment. Therefore, we again have $|p_0^{\mathscr{B}} - p_1^{\mathscr{B}}| > \mathsf{negl}(\lambda)$, contradicting the RE property of $\Sigma_{\overline{b}}$ and completing the proof of Lemma 2. □

By the triangle inequality, for all $\mathscr{A}$ we have

$$\begin{aligned} |p_{\texttt{RS}}^{\mathscr{A}}(\lambda) - p_{\texttt{SR}}^{\mathscr{A}}(\lambda)| &= |(p_{\texttt{RS}}^{\mathscr{A}}(\lambda) - p_{\texttt{SS}}^{\mathscr{A}}(\lambda)) + (p_{\texttt{SS}}^{\mathscr{A}}(\lambda) - p_{\texttt{SR}}^{\mathscr{A}}(\lambda))| \\ &\leq |p_{\texttt{RS}}^{\mathscr{A}}(\lambda) - p_{\texttt{SS}}^{\mathscr{A}}(\lambda)| + |p_{\texttt{SS}}^{\mathscr{A}}(\lambda) - p_{\texttt{SR}}^{\mathscr{A}}(\lambda)|. \end{aligned}$$

We showed via the two lemmas that $|p_{\texttt{RS}}^{\mathscr{A}}(\lambda) - p_{\texttt{SS}}^{\mathscr{A}}(\lambda)| \leq \mathsf{negl}(\lambda)$ and $|p_{\texttt{SS}}^{\mathscr{A}}(\lambda) - p_{\texttt{SR}}^{\mathscr{A}}(\lambda)| \leq \mathsf{negl}(\lambda)$. By substitution, we have

$$|p_{\texttt{RS}}^{\mathscr{A}}(\lambda) - p_{\texttt{SR}}^{\mathscr{A}}(\lambda)| \leq \mathsf{negl}(\lambda) + \mathsf{negl}(\lambda) \leq \mathsf{negl}(\lambda),$$

completing the proof of Theorem 1. □

A.8 The Standard Random Oracle Model

RO $H_f(x)$	Random List Oracle $H_L(x)$	Interface $\texttt{Prog}_L(x, v)$
1 : **return** $f(x)$	1 : **if** $\exists\ v$ **s.t.** $(x, v) \in L$:	1 : **if** $\nexists\ v'$ **s.t.** $(x, v') \in L$:
	2 : **return** v	2 : L.**append**(x, v)
	3 : **else** :	
	4 : $v \leftarrow \{0,1\}^\ell$	
	5 : L.**append**(x, v)	
	6 : **return** v	

Fig. 8. Random Oracle Functionalities for NIM-SHVZK and NI-SSS Games [42].

A.9 Non-Interactive Special Soundness

Definition 21 (Non-Interactive Special Soundness). [42] *A NISLE proof system* $\Pi_R^{\texttt{SLC}} = (\texttt{Setup}^H, \texttt{Prove}^H, \texttt{Verify}^H, \texttt{SimSetup}, \texttt{SimProve}, \texttt{Extract})$ non-interactive special sound (NI-SS) *in the random-oracle model if there exists an*

algorithm $\Pi_R^{\mathsf{SLC}}.\mathtt{Extract}$ *such that for any security parameter* λ *any random oracle* H*, and any* PPT *adversary* $\mathscr{A}$*,*

$$\Pr[\mathtt{Fail} \leftarrow \mathsf{NI\text{–}SS}_{\mathscr{A},\Pi_R^{\mathsf{SLC}}}(1^\lambda)] \leq \mathsf{negl}(\lambda),$$

where NI–SS *is the NI-SS game described in Figure 9. We say* $\mathscr{A}$ *wins the* NI–SS *game if* $\Pr[\mathtt{Fail} \leftarrow \mathsf{NI\text{–}SS}^H_{\mathscr{A},\Pi_R^{\mathsf{SLC}}}(1^\lambda)] > \mathsf{negl}(\lambda)$.

$\mathsf{NI\text{–}SS}^H_{\mathscr{A},\Pi_R^{\mathsf{SLC}}}(1^\lambda)$

```
1 :  ppm ← Π_R^SLC.Setup(1^λ)
2 :  st ← A^H(1^λ, ppm)
3 :  while st ≠ ⊥ :
4 :    (x, π, Q^A, st) ← A^H(st)
5 :    Response ← ⊥
6 :      if Π_R^SLC.Verify^H(x, π) = 1 :
7 :        w ← Π_R^SLC.Extract(x, π, Q^A)
8 :        if R(x, w) = 0 :
9 :          return Fail
10 :   st ← A^H(st, Response)
11 : return Success
```

Fig. 9. Non-Interactive Special Soundness (NI-SS) Game.

A.10 aUC NIZKPoK Required Properties Full Proof

Recall Theorem 6: Let Π be a protocol that aUC-realizes $\mathcal{F}_{\mathtt{aNIZK}}$ in the $\mathcal{G}_{\mathtt{RO}}$-hybrid model (Definition 1 where $\mathcal{G}$ is replaced with $\mathcal{G}_{\mathtt{RO}}$) with adaptive corruptions. Then Π must be overwhelmingly complete (Definition 9), adaptive NIM-SHVZK (Definition 10), and adaptive NI-SSS (Definition 11).

Proof. We proceed by cases and show that if Π is not overwhelmingly complete and adNIM-SHVZK then it does not aUC-realize $\mathcal{F}_{\mathtt{aNIZK}}$ in the $\mathcal{G}_{\mathtt{RO}}$-hybrid model with adaptive corruptions, and similarly that if Π is not adNI-SSS then it does not aUC-realize $\mathcal{F}_{\mathtt{aNIZK}}$ in the same model.

Our first reduction $\mathscr{B}$ uses an adversary $\mathscr{A}$ that wins the adNIM-SHVZK game from Figure 4 with non-negligible advantage to distinguish between the real- and ideal-world aUC experiments with non-negligible advantage. $\mathscr{B}$ proceeds as follows. After passing the security and public parameters that it received from the aUC experiment to $\mathscr{A}$, $\mathscr{B}$ forwards $\mathscr{A}$'s oracle queries to $\mathcal{G}_{\mathtt{RO}}$ and $\mathcal{G}_{\mathtt{RO}}$'s responses back to $\mathscr{A}$.

`Prove` queries proceed as follows. Note that according to the definition of adNIM-SHVZK, any time $\mathscr{A}$ issues a `Prove` query, it is expecting not only a proof, *but also the proof's randomness*, in return. In order to accurately simulate $\mathscr{A}$'s view, $\mathscr{B}$ first issues the `Prove` query as-is to a new honest party in the aUC experiment.[5] Before returning the proof to $\mathscr{A}$, $\mathscr{B}$ then *corrupts* the prover to obtain the prover's internal tapes. Since the `Prove` operation was the first performed by the honest party, $\mathscr{B}$ simply takes the first λ_r bits of the prover's random tape and returns these bits to $\mathscr{A}$ along with the proof.

If the aUC challenger is running the real-world experiment, the proof will be the result of running Π.`Prove` on the prover's witness and randomness. If the aUC challenger is running the ideal-world experiment, the proof will be the result of the ideal functionality $\mathcal{F}_{\texttt{aNIZK}}$ running Π.`SimProve`, and the randomness will have been generated by the simulator (ideal adversary) $\mathcal{S}$ using Π.`SimRand`. Therefore, $\mathscr{B}$ simulates $\mathscr{A}$'s view exactly, and succeeds in distinguishing the real- and ideal-world aUC experiments with the same probability that $\mathscr{A}$ distinguishes the real- from ideal-world adaptive NIM-SHVZK game.

The only other condition that might allow $\mathscr{B}$ to distinguish the two experiments are if the ideal functionality $\mathcal{F}_{\texttt{aNIZK}}$ in the ideal-world aUC experiment outputs `Fail` due to a completeness error. This condition occurs with negligible probability due to overwhelming completeness.

The second reduction uses two black-box algorithms: $\mathscr{A}$ that wins the adaptive NI-SSS game from Figure 5, and $\mathscr{A}'$ that wins the regular NI-SS game (Definition 21), in which the adversary does not have access to simulated proofs, with non-negligible advantage. $\mathscr{B}$ answers $\mathscr{A}$'s queries the same as in the previous reduction, by forwarding all of $\mathscr{A}$'s oracle queries to $\mathcal{G}_{\texttt{RO}}$ and `Prove` queries to the aUC challenger, then making adaptive corruptions to obtain the prover's randomness. $\mathscr{B}$ forwards $\mathscr{A}'$'s queries to $\mathcal{G}_{\texttt{RO}}$ (note $\mathscr{A}'$ does not make `Prove` queries, but can run Π.`Prove` itself).

$\mathscr{B}$ proceeds the same as before, forwarding all of $\mathscr{A}$'s oracle queries to $\mathcal{G}_{\texttt{RO}}$ and `Prove` queries to the aUC challenger, then making adaptive corruptions to obtain the prover's randomness. Whenever $\mathscr{A}$ (resp. $\mathscr{A}'$) outputs a proof π for a statement x such that $\Pi.\texttt{Verify}(x, \pi) = 1$, $\mathscr{B}$ gathers $\mathscr{A}$'s (resp. $\mathscr{A}'$'s) oracle queries $\mathcal{Q}_{\mathscr{A}}$ (resp. $\mathcal{Q}_{\mathscr{A}'}$) and runs $w \leftarrow \Pi.\texttt{Extract}(x, \pi, \mathcal{Q}_{\mathscr{A}(\text{resp.}\mathscr{A}')})$. If w is such that $R(x, w) = 0$, $\mathscr{B}$ invokes a new honest party and sends it the instruction $(\texttt{Verify}, x, \pi)$. If the aUC challenger is running the ideal-world experiment, then $\mathscr{B}$ has simulated $\mathscr{A}$'s expected view, and the honest (dummy) party will invoke $\mathcal{F}_{\texttt{aNIZK}}$ on $\mathscr{A}$'s proof and output `Fail`, causing $\mathscr{B}$ to output "ideal." If the aUC challenger is running the real-world experiment, then on input a proof π from $\mathscr{A}'$, the real-world honest party will output $\Pi.\texttt{Verify}(x, \pi) = 1$, causing $\mathscr{B}$ to output "real." $\mathscr{B}$ therefore distinguishes the ideal- from real-world aUC experiments with the same probability as $\mathscr{A}$ and $\mathscr{A}'$, respectively. □

[5] Recall that as part of the (adaptive) UC experiment, the environment is permitted to spawn polynomially-many protocol participants, subject to polynomial run-time restrictions [19].

A.11 aUC NIZKPoK in the $\mathcal{G}_{\texttt{rpoRO}}$-hybrid Model Full Proof

Recall Theorem 7: Let Σ_R be any adaptive Σ-protocol for relation R (Definition 2), $\mathcal{G}_{\texttt{rpoRO}}$ be the restricted programmable observable global random oracle (Definition 14) and $\texttt{SLC}$ be any adaptive straight-line compiler (Definition 8). Then the NISLE proof system $\Pi_R^{\texttt{SLC}} \leftarrow \texttt{SLC}(\Sigma_R)$ aUC-realizes $\mathcal{F}_{\texttt{aNIZK}}$ in the $\mathcal{G}_{\texttt{rpoRO}}$-hybrid model (Definition 1 where $\mathcal{G}$ is replaced with $\mathcal{G}_{\texttt{rpoRO}}$).
Proof. First, we make explicit the additional check due to Chiesa and Fenzi [25] that an honest verifier (and the **Verify** interface of the ideal functionality $\mathcal{F}_{\texttt{aNIZK}}$) must make in order to correctly evaluate a proof. Recall that $\mathcal{G}_{\texttt{rpoRO}}$ is programmable by any party participating in a protocol session s, including the adversarial session participants who are controlled by the environment. Therefore, honest verifiers (and $\mathcal{F}_{\texttt{aNIZK}}$) must check whether $\mathcal{G}_{\texttt{rpoRO}}$ was programmed at any index queried by the verification procedure, and if so, reject the proof.

Definition 22 ($\mathcal{G}_{\texttt{rpoRO}}$-hybrid Model Verification Check). *Let $\Pi_R^{\texttt{SLC}}$ be any NISLE proof system (Definition 8), $\mathcal{F}_{\texttt{aNIZK}}$ be the adaptive non-interactive zero-knowledge proof of knowledge ideal functionality (Definition 12), and $\mathcal{G}_{\texttt{rpoRO}}$ be the restricted programmable observable global random oracle (Definition 13). Every time the* $\texttt{Verify}$ *algorithm of $\Pi_R^{\texttt{SLC}}$ or the* **Verify** *interface of $\mathcal{F}_{\texttt{aNIZK}}$ queries $\mathcal{G}_{\texttt{rpoRO}}$ on some input* in*, insert the following check: send a query* $\texttt{IsProgrammed}(\mathsf{in})$ *to $\mathcal{G}_{\texttt{rpoRO}}$; if $\mathcal{G}_{\texttt{rpoRO}}$ returns* 1*, output* 0 *(reject). Otherwise, proceed.*

We now show that any adaptive straight-line compiler is sufficient to create adaptive UC NIZKPoK in the $\mathcal{G}_{\texttt{rpoRO}}$-hybrid model. We now proceed with the construction of the ideal adversary (simulator) $\mathcal{S}$.

Construction of the Ideal Adversary. When $\mathcal{S}$ receives $(\texttt{Setup}, s)$ from $\mathcal{F}_{\texttt{aNIZK}}$, $\mathcal{S}$ returns the tuple of algorithms $\Pi_R^{\texttt{SLC}}$. When corrupted parties issue $\texttt{IsProgrammed}$ queries, $\mathcal{S}$ returns $\texttt{false}$. When $\mathcal{Z}$ issues a query $(\textbf{Corrupt}, P)$ for a party $P = (\texttt{pid}, \texttt{sid})$, $\mathcal{S}$ sends $(\textbf{Corrupt}, P)$ to $\mathcal{F}_{\texttt{aNIZK}}$ to obtain the stored tuples $(\texttt{Proof}, P, x_1, w_1, \pi_1, z_1), \ldots, (\texttt{Proof}, P, x_q, w_q, \pi_q, z_q)$ corresponding to each query $(\texttt{Prove}, P, x_i, w_i)$ that $\mathcal{Z}$ issued to P. (who forwarded them to $\mathcal{F}_{\texttt{aNIZK}}$). For each tuple, $\mathcal{S}$ interprets $z_i = (z_s, z_{\pi_i})$ as the auxiliary outputs of $\Pi_R^{\texttt{SLC}}.\texttt{SimSetup}$ and $\Pi_R^{\texttt{SLC}}.\texttt{SimProve}$, respectively. $\mathcal{S}$ then runs $r_i \leftarrow \Pi_R^{\texttt{SLC}}.\texttt{SimRand}(z_s, z_{\pi_i}, x, \pi, w)$ to obtain simulated randomness r_i. To reconstruct the prover's random tape R, $\mathcal{S}$ concatenates $R = r_1 || \ldots || r_q$ and returns R to $\mathcal{Z}$. Otherwise, $\mathcal{S}$ forwards all communications between $\mathcal{Z}$ and the protocol.

We proceed by creating a hybrid argument that starts in the real-world experiment and replaces each piece of the real-world protocol $\Pi_R^{\texttt{SLC}}$ with the functionality of $\mathcal{F}_{\texttt{aNIZK}}$ and $\mathcal{S}$.

Hybrid 1. First, we replace all of the environment's and adversary's connections to the real-world protocol participants with the "challenger" of our reduction, $\mathcal{C}$. This difference is syntactic, so Hybrid 1 is identical to the real-world experiment.

Hybrid 2. In the second hybrid, we replace $\mathcal{C}$'s real-world $\texttt{Prove}$ functionality with the **Prove** interface of $\mathcal{F}_{\texttt{aNIZK}}$ and random tape simulation of $\mathcal{S}$, and

show the environment's views are indistinguishable between Hybrids 1 and 2 as long as Σ_R is adaptive and non-interactive multiple special honest-verifier zero-knowledge (adNIM-SHVZK). First, we specify $\mathcal{C}$ to simulate $\mathcal{G}_{\texttt{rpoRO}}$ according to its specification (noting that $\mathcal{C}$ can "program" its simulation) and return `false` to all of the corrupted parties' `IsProgrammed` queries. As long as $\Pi_R^{\texttt{SLC}}.\texttt{SimProve}$ produces valid proofs for statements $x \in L_R$ with overwhelming probability (which follows from overwhelming completeness), the environment's view of $\mathcal{G}_{\texttt{rpoRO}}$ remains statistically indistinguishable between the hybrids (which follows from the restriction of the `IsProgrammed` interface), $\mathcal{Z}$ is forced to distinguish the hybrids based on the only other difference—the proofs π_i and randomness r_i.

We bound $\mathcal{Z}$'s probability of distinguishing the hybrids based on the proofs and randomness by constructing a reduction to the adNIM-SHVZK property as follows. Whenever $\mathcal{Z}$ issues a query $(\texttt{Prove}, P, x_i, w_i)$ to $\mathcal{C}$, $\mathcal{C}$ forwards the query to the adNIM-SHVZK challenger from Figure 4, who returns (π_i, r_i). $\mathcal{C}$ forwards π_i to $\mathcal{Z}$ and stores (P, r_i) for later. If $\mathcal{Z}$ issues a query $(\textbf{Corrupt}, P)$, $\mathcal{C}$ retrieves all of the tuples (P, r_i) and sets the random tape R to be the concatenation of all r_i in the order they were stored. It then returns R to $\mathcal{C}$. For $\mathcal{Z}$'s queries $(\texttt{Verify}, x, \pi)$, $\mathcal{C}$ returns the result of running $\Pi_R^{\texttt{SLC}}.\texttt{Verify}(x, \pi)$. $\mathcal{C}$ outputs whatever $\mathcal{Z}$ outputs.

Note that if the adNIM-SHVZK challenger is running the 0-bit experiment (using $\Pi_R^{\texttt{SLC}}.\texttt{Prove}$ and the prover's randomness), $\mathcal{C}$ simulates $\mathcal{Z}$'s exact view of the experiment in Hybrid 1. Else if the adNIM-SHVZK challenger is running the 1-bit experiment (using $\Pi_R^{\texttt{SLC}}.\texttt{SimProve}$ and $\Pi_R^{\texttt{SLC}}.\texttt{SimRand}$), $\mathcal{C}$ simulates $\mathcal{Z}$'s view of Hybrid 2. Therefore, $\mathcal{C}$ succeeds at winning the adNIM-SHVZK game with the same probability that $\mathcal{Z}$ can distinguish the hybrids, proving Hybrid 1 is computationally indistinguishable to Hybrid 2.

Hybrid 3. In the third hybrid, we replace $\mathcal{C}$'s `Verify` functionality with the **Verify** functionality of $\mathcal{F}_{\texttt{aNIZK}}$, and show the environment's views are indistinguishable between Hybrids 2 and 3 as long as $\Pi_R^{\texttt{SLC}}$ is adaptive non-interactive special simulation-sound (adNI-SSS). We construct a reduction that uses an environment $\mathcal{Z}$ that can distinguish Hybrids 2 and 3 with non-negligible advantage to win the adNI-SSS game from Figure 5 as follows. First, note the only difference in output between Hybrids 2 and 3 is that the Hybrid 3 experiment can output `Fail`, while the Hybrid 2 experiment never does—in particular, Hybrid 3 will output `Fail` if $\mathcal{Z}$ succeeds in producing a valid, non-simulated proof that causes $\Pi_R^{\texttt{SLC}}.\texttt{Extract}$ to output `Fail`. For $\mathcal{Z}$'s `Prove` queries, the reduction acts according to Hybrid 2, this time forwarding the queries to the adNI-SSS challenger, returning the proofs, and saving random bits in case $\mathcal{Z}$ issues a corruption query on the prover. When $\mathcal{Z}$ issues a query $(\texttt{Verify}, x, \pi)$ for a proof π that $\mathcal{C}$ did not send to $\mathcal{Z}$, $\mathcal{C}$ sends $(\texttt{Challenge}, x, \pi, \mathcal{Q}_{P^*})$ to the adNI-SSS challenger. Since both the adNI-SSS challenger and $\mathcal{F}_{\texttt{aNIZK}}$ use the $\Pi_R^{\texttt{SLC}}.\texttt{Extract}$ algorithm and fail under the same conditions, $\mathcal{C}$ succeeds in winning the adNI-SSS game with the same probability that $\mathcal{Z}$ distinguishes Hybrids 2 and 3.

Hybrid 4. The final hybrid replaces $\mathcal{C}$ with $\mathcal{F}_{\texttt{aNIZK}}$ and $\mathcal{S}$. Note that since $\mathcal{C}$ now runs all of $\mathcal{F}_{\texttt{aNIZK}}$ and $\mathcal{S}$'s procedures, this is again a syntactic difference, and Hybrid 3 is identical to Hybrid 4. □

A.12 Requirements of the Randomized Fischlin Transform

To stop $\mathscr{A}$ from predicting $\texttt{com}$ and querying the RO on $(x, \texttt{com})$ before the prover does, the $\texttt{com}$ messages of Σ-protocols under not just Fischlin's transform but any non-interactive transform in the ROM need entropy that is superlogarithmic in the security parameter.

Definition 23 (Superlogarithmic Commitment Entropy). [34] *Let Σ_R be any Σ-protocol for binary $\mathcal{NP}$ relation R and template τ as specified in Definition 2. Σ_R has* superlogarithmic commitment entropy *if for all $(x, w) \in L_R$, the min-entropy of $\texttt{com} \leftarrow \tau.\texttt{Commit}(x, w)$ is superlogarithmic in λ.*

The strong special soundness property [41] says that the extractor must still work as long as there is *some* difference between the challenges and responses of two transcripts—in particular, it could be that $\texttt{chl} = \texttt{chl}'$, as long as $\texttt{res} \neq \texttt{res}'$.

Definition 24 (Strong Special Soundness). [41] *A Σ-protocol Σ_R for relation R (Definition 2) has the* strong special soundness *property if the condition $\texttt{chl} \neq \texttt{chl}'$ in the specification of the $\Sigma.\texttt{Extract}$ algorithm is replaced with the condition $(\texttt{chl}, \texttt{res}) \neq (\texttt{chl}', \texttt{res}')$.*

In the adaptive setting, generating a convincing version of the prover's entire challenge sequence is non-trivial, and in particular requires a simulation technique other than the dynamic programming strategy of the original and randomized Fischlin transforms [34,41]. To see why this is the case, consider the setting of witness indistinguishability, where there can be two witnesses, w_0 and w_1, that satisfy a particular statement x. Given the constraint of a challenge space that is logarithmic in the security parameter λ, the adversary can, upon receiving a proof from $\Pi_R^{\texttt{aFis}}.\texttt{SimProve}$ (but before issuing a corruption to receive the output of $\Pi_R^{\texttt{aFis}}.\texttt{SimRand}$), test outputs of the RO on transcripts it generates using w_0 or w_1 and the same commitments as $\Pi_R^{\texttt{aFis}}.\texttt{SimProve}$, but different challenges.

In the original version of the transform, the $\texttt{SimProve}$ algorithm samples a mapping μ of challenges to outputs, and simulates the proof transcript using the first challenge in μ that maps to the all-zero string [34]. It then programs the RO to return the all-zero string for the simulated proof transcript [34] and outputs the proof. Whenever the adversary makes any queries of the form described above, $\texttt{SimProve}$ programs the RO on input the adversary's proof transcript such that the challenge-to-RO-output is consistent with the mapping μ. Once the prover is corrupted and its witness w_b is revealed, however, the mapping μ becomes fixed to w_b. The adversary can now perform a new kind of replay attack, where it compares the distribution of the outputs that it queried (potentially with w_{1-b}) to the distribution of μ. In the real world, transcripts computed

over the same commitments and challenges but using w_{1-b} rather than w_b will produce an output distribution μ' that is independent of μ. At the time that `SimProve` programs the RO, it is not able to determine whether the witness used by the adversary matches the witness used by the prover, and can therefore only guess (with success probability $\frac{1}{2}$) whether it should be programming the RO output from the distribution μ or μ'.

A.13 Discussion of Modules

We form the Σ-protocol module over the challenge space $\mathcal{C}$ and the language (statement) space $\mathcal{X}$. The definition of a module requires that $\mathcal{C}$ is a ring with multiplicative identity $1_{\mathcal{C}}$, $\mathcal{X}$ is a group with additive commutativity, and for all $\mathtt{chl}, \mathtt{chl}' \in \mathcal{C}$ and $x, y \in \mathcal{X}$, there exists a map $\mathcal{C} \times \mathcal{X} \to \mathcal{X}$ satisfying the following properties: 1) $\mathtt{chl} \cdot (x + y) = \mathtt{chl} \cdot x + \mathtt{chl} \cdot y$; 2) $(\mathtt{chl} + \mathtt{chl}') \cdot x = \mathtt{chl} \cdot x + \mathtt{chl}' \cdot x$; 3) $(\mathtt{chl} \cdot \mathtt{chl}') \cdot x = \mathtt{chl} \cdot (\mathtt{chl}' \cdot x)$; and 4) $1_{\mathcal{C}} \cdot x = x$. In order to guarantee the special soundness property, we additionally require that for any two proofs $\pi = (\mathtt{com}, \mathtt{chl}, \mathtt{res})$ and $\pi' = (\mathtt{com}, \mathtt{chl}', \mathtt{res}')$ for x such that $\mathtt{Verify}(x, \pi) = \mathtt{Verify}(x, \pi') = 1$ and $\mathtt{chl} \neq \mathtt{chl}' \neq 0_{\mathcal{C}}$, the inverse of $(\mathtt{chl} - \mathtt{chl}')$ exists. We write the inversion $(\mathtt{chl} - \mathtt{chl}) \cdot (\mathtt{chl} - \mathtt{chl}')^{-1} = 1_{\mathcal{C}}$.

Note that some Σ-protocol constructions that would otherwise fit this template, such as the proof of knowledge of an RSA inverse, do not quite work because the extractor cannot compute the inverse of $(\mathtt{chl} - \mathtt{chl}')$ directly (in the case of the RSA inverse, the extractor uses Shamir's trick [13]). For such constructions, it might be convenient to use the abstraction for the adaptive SHVZK property and treat the special soundness property separately.

A.14 $\Sigma_{\mathtt{simple}}$ is Adaptive Full Proof

Recall Theorem 2: The simple adaptive Σ-protocol candidate (Definition 7) is an adaptive Σ-protocol (Definition 3) with randomness equivocability (Definition 4).

Proof. The proof relies on two core properties of the evaluation function $\mathtt{F}$, described below (for details in context, see Definition 4.1 [38]). First, the *pseudo-module homomorphism* (PMH) property states that for all $y, z \in \mathcal{W}$, $\mathtt{F}(\mathtt{chl} \cdot y + z) = \mathtt{chl} \cdot \mathtt{F}(y) + \mathtt{F}(z)$. Second, the *smoothness property* guarantees that for all $y \in \mathcal{W}$, $\mathtt{F}(y)$ is uniformly distributed over $\mathcal{X}$. (Recall these properties hold as well for $\mathcal{W} = \mathcal{R} = \mathcal{B}$ and $\mathcal{X} = \mathcal{A}$.) We will now show that the simple Σ-protocol format given above satisfies the necessary completeness, adaptive special honest-verifier zero-knowledge, and special soundness properties of an adaptive Σ-protocol, as well as randomness equivocability.

Completeness. The verifier checks whether $F(\mathtt{res}) = \mathtt{chl} \cdot x + \mathtt{com}$. By the PMH property, we have $F(\mathtt{chl} \cdot w + r) = \mathtt{chl} \cdot F(w) + F(r) = \mathtt{chl} \cdot x + \mathtt{com}$.

Adaptive SHVZK and Randomness Equivocability. As long as $r \leftarrow_{\$} \mathcal{R}$ and $\mathtt{chl} \leftarrow_{\$} \mathcal{C}$, the distribution of $\mathtt{res} = \mathtt{chl} \cdot w + r$ computed in $\Sigma_{\mathtt{simple}}.\mathtt{Prove}$ is uniformly random in $\mathcal{B}$. The $\mathtt{res} \leftarrow_{\$} \mathcal{B}$ and $\mathtt{chl} \leftarrow_{\$} \mathcal{C}$ sampled by $\Sigma_{\mathtt{simple}}.\mathtt{SimProve}$ are therefore identical to the $\mathtt{res}$ and $\mathtt{chl}$ in a real proof. By the smoothness

of $\mathtt{F}$, $\Sigma_{\mathtt{simple}}.\mathtt{SimProve}$'s $\mathtt{com} = F(\mathtt{res} - \mathtt{chl} \cdot w)$ is also uniformly random in $\mathcal{A}$. Since $\mathtt{res}$ and $\mathtt{chl}$ are uniformly random in $\mathcal{B}$ and $\mathcal{C}$ respectively, the distribution of $r = \mathtt{res} - (\mathtt{chl} \cdot w)$ will be similarly random. Note that both simulations are correct, since $\mathtt{F}(r) = \mathtt{F}(\mathtt{res} - \mathtt{chl} \cdot w) = -\mathtt{chl} \cdot \mathtt{F}(w) + \mathtt{F}(\mathtt{res}) = -\mathtt{chl} \cdot x + \mathtt{chl} \cdot x + \mathtt{com} = \mathtt{com}$ by the PMH property. The outputs of $\Sigma_{\mathtt{simple}}.\mathtt{Prove}$ and r are therefore distributed identically to the outputs of $\Sigma_{\mathtt{simple}}.\mathtt{SimProve}$ and $\Sigma_{\mathtt{simple}}.\mathtt{SimRand}$, respectively. Finally, in order to make a real proof look simulated, the $\mathtt{RealToSim}$ algorithm simply pretends that the uniform randomness of $\mathtt{res}$ formed the simulator's random tape.

Special Soundness. Solving the two verification equations for $\mathtt{com}$ and setting them equal, we get $\mathtt{F}(\mathtt{res}) - \mathtt{chl} \cdot x = \mathtt{F}(\mathtt{res}') - \mathtt{chl}' \cdot x$, which implies $\mathtt{F}(w) \cdot \mathtt{chl} - \mathtt{F}(w) \cdot \mathtt{chl}' = \mathtt{F}(\mathtt{res}) - \mathtt{F}(\mathtt{res}')$ (by substitution and commutativity) and $\mathtt{F}(w) \cdot (\mathtt{chl} - \mathtt{chl}') = \mathtt{F}(\mathtt{res}) - \mathtt{F}(\mathtt{res}')$ (by distributivity). By the invertability of $\mathtt{chl} - \mathtt{chl}'$, we obtain $\mathtt{F}(w) = \mathtt{F}(\mathtt{res}) - \mathtt{F}(\mathtt{res}') \cdot (\mathtt{chl} - \mathtt{chl}')^{-1}$. Finally, the PMH property implies $\mathtt{F}(w) = \mathtt{F}((\mathtt{res} - \mathtt{res}') \cdot (\mathtt{chl} - \mathtt{chl}')^{-1})$, then $w = (\mathtt{res} - \mathtt{res}') \cdot (\mathtt{chl} - \mathtt{chl}')^{-1}$. □

A.15 PoK DL is an Adaptive Σ-protocol Proof

Recall Theorem 3: A proof of knowledge of discrete logarithm can be described as a simple adaptive Σ-protocol with randomness equivocability (Definition 7).

The PoK DL can be described using the simple adaptive Σ-protocol format from Definition 7 as follows.

The public parameters $\mathtt{ppm}$ are $(\mathbb{G}, q, g)$, where $\mathbb{G}$ is a cyclic group of prime order q, and g is a generator of $\mathbb{G}$. The statement and first-message spaces are $\mathcal{X} = \mathcal{A} = \mathbb{G}$, the witness, randomness, and response spaces are $\mathcal{W} = \mathcal{B} = \mathcal{R} = \mathbb{Z}_q$, and the challenge space is $\mathcal{C} = \mathbb{Z}_q^*$. The evaluation function $\mathtt{F} : \mathbb{Z}_q \to \mathbb{G}$ is defined such that $\mathtt{F}(w) = w \cdot g$. For elements x and y in $\mathbb{Z}_q$, we define the operation $x \cdot y$ to be the process of adding y to itself (component-wise) x times modulo q. For elements $x \in \mathbb{Z}_q, \mathbb{Z}_q^*$ and $y \in \mathbb{G}$, we define $x \cdot y$ to be the process of performing the group operation on y with itself x times. (In traditional Schnorr, this would be exponentiation y^x; in the elliptic curve version, it would the scalar multiplication xy where y is a curve point.)

$\mathbb{Z}_q^*$ and $\mathbb{G}$ satisfy the requirements of a module, and all elements in $\mathcal{C} = \mathbb{Z}_q^*$ are invertible. Furthermore, the pseudo-module homomorphism property is satisfied because $\mathtt{F}(w) := w \cdot g \Rightarrow \mathtt{F}(\mathtt{chl} \cdot w + r) = (\mathtt{chl} \cdot w + r) \cdot g = \mathtt{chl} \cdot w \cdot g + r \cdot g = \mathtt{chl} \cdot F(w) + F(r)$. Finally, $\mathtt{F}$ is smooth, since for $r \leftarrow_{\$} \mathbb{Z}_q$, $F(r) = r \cdot g$ is uniformly distributed over $\mathbb{G}$.

A.16 PoK EqRep is an Adaptive Σ-protocol Proof

Recall Theorem 4: A proof of knowledge of equality of n representations can be described as a simple adaptive Σ-protocol (Definition 7).

Proof Sketch. PoK EqRep is a natural extension of PoK DL, and fits the abstract form from Definition 7 as follows. The public parameters $\mathtt{ppm}$ are $(\mathbb{G}, q, \overline{g}_1, \ldots, \overline{g}_n)$,

where $\mathbb{G}$ is a cyclic group of prime order q, and each $\overline{g}_i$ is a vector of m generators $g_{i,1}, \ldots, g_{i,m}$. The statement and first-message spaces are $\mathcal{X} = \mathcal{A} = \mathbb{G}^n$, the witness, randomness, and response spaces are $\mathcal{W} = \mathcal{R} = \mathcal{B} = \mathbb{Z}_q^m$, and the challenge space is $\mathcal{C} = \mathbb{Z}_q^*$. The evaluation function $\mathtt{F} : \mathbb{Z}_q^m \to \mathbb{G}^n$ is such that

$$\mathtt{F}(w) = \mathtt{F}(w_1, \ldots, w_m) = \prod_{i=1}^{m} w_i \cdot g_{1,i}, \ldots, \prod_{i=1}^{m} w_i \cdot g_{n,i}.$$

The operator $\cdot$ is defined the same as in the proof of Theorem 3 above (with emphasis on component-wise). Likewise, the pseudo-module homomorphism and smoothness properties are satisfied via component-wise application of the logic from Theorem 3. Since PoK EqRep is more involved than PoK DL, we provide a more comprehensive treatment of the construction and proof in A.16. □

Full Proof. We recall the setup. The public parameters $\mathtt{ppm}$ are $(\mathbb{G}, q, \overline{g}_1, \ldots, \overline{g}_n)$, where $\mathbb{G}$ is a cyclic group of prime order q, and each $\overline{g}_i$ is a vector of m generators $g_{i,1}, \ldots, g_{i,m}$. The statement and first-message spaces are $\mathcal{X} = \mathcal{A} = \mathbb{G}^n$, the witness, randomness, and response spaces are $\mathcal{W} = \mathcal{R} = \mathcal{B} = \mathbb{Z}_q^m$, and the challenge space is $\mathcal{C} = \mathbb{Z}_q^*$. The evaluation function $\mathtt{F} : \mathbb{Z}_q^m \to \mathbb{G}^n$ is defined such that

$$\mathtt{F}(w) = \mathtt{F}(w_1, \ldots, w_m) = \prod_{i=1}^{m} w_i \cdot g_{1,i}, \ldots, \prod_{i=1}^{m} w_i \cdot g_{n,i}.$$

The operator $\cdot$ is defined, component-wise, the same as in the proof of Theorem 3.

The prover's first message $\mathtt{com} = \mathtt{F}(r)$ becomes

$$\mathtt{F}(r_1, \ldots, r_m) = \prod_{i=1}^{m} w_i \cdot g_{1,i}, \ldots, \prod_{i=1}^{m} w_i \cdot g_{n,i}$$

for $r_1, \ldots, r_m \leftarrow_{\$} \mathbb{Z}_q$. We can therefore write $\mathtt{com} = \langle \mathtt{com}_1, \ldots, \mathtt{com}_n \rangle$. The challenge is a uniformly random element $\mathtt{chl} \leftarrow_{\$} \mathbb{Z}_q^*$, and the prover's response is $\mathtt{res} = \langle \mathtt{res}_1, \ldots, \mathtt{res}_m$, where each $\mathtt{res}_i = \mathtt{chl} \cdot w_i + r_m$. The verifier accepts the proof $(\mathtt{com}, \mathtt{chl}, \mathtt{res})$ if and only if

$$\mathtt{F}(\mathtt{res}) = \mathtt{F}(\mathtt{res}_1, \ldots, \mathtt{res}_m) = \prod_{i=1}^{m} \mathtt{res}_i \cdot g_{1,i}, \ldots, \prod_{i=1}^{m} \mathtt{res}_i \cdot g_{n,i} =$$

$$\prod_{i=1}^{m} (\mathtt{chl} \cdot w_i + r_i) \cdot g_{1,i}, \ldots, \prod_{i=1}^{m} (\mathtt{chl} \cdot w_i + r_i) \cdot g_{n,i} =$$

$$\mathtt{chl} \cdot \mathtt{F}(w) + \mathtt{F}(r) = \mathtt{chl} \cdot x + \mathtt{com}.$$

The $\mathtt{SimProve}$ algorithm samples a random m-length response vector from $\mathbb{Z}_q^m$ and computes each component of the n-length first message vector $\mathtt{com}_1, \ldots, \mathtt{com}_n$ as

$$\mathtt{com} = \mathtt{F}(\mathtt{res} - \mathtt{chl} \cdot w) = \prod_{i=1}^{m} (\mathtt{res}_i - \mathtt{chl} \cdot w_i) \cdot g_{1,i}, \ldots, \prod_{i=1}^{m} (\mathtt{res}_i - \mathtt{chl} \cdot w_i) \cdot g_{n,i},$$

and each random value r_i output by $\texttt{SimRand}$ is computed $\texttt{res}_i - (\texttt{chl} \cdot w_i)$ for $1 \leq i \leq m$. Similarly, each witness w_i is extracted by computing $(\texttt{res}_i - \texttt{res}'_i) \cdot (\texttt{chl} - \texttt{chl}')^{-1}$, where $(\texttt{chl} - \texttt{chl}') \neq 0$ is again invertible in $\mathbb{Z}_q^*$.

The pseudo-module homomorphism and smoothness properties follow from the component-wise application of the arguments from Theorem 3:

$$\texttt{F}(\texttt{chl} \cdot w + r) = \texttt{F}(\texttt{chl} \cdot w_1, \ldots, w_m + r_1, \ldots, r_m) =$$

$$\prod_{i=1}^{m}(\texttt{chl} \cdot w_i + r_i) \cdot g_{1,i}, \ldots, \prod_{i=1}^{m}(\texttt{chl} \cdot w_i + r_i) \cdot g_{n,i} =$$

$$\prod_{i=1}^{m}(\texttt{chl} \cdot w_i) \cdot g_{1,i} + r_i \cdot g_{1,i}, \ldots, \prod_{i=1}^{m}(\texttt{chl} \cdot w_i) \cdot g_{n,i} + r_i \cdot g_{n,i} =$$

$$\texttt{chl} \cdot \prod_{i=1}^{m} w_i \cdot g_{1,i} + r_i \cdot g_{1,i}, \ldots, \texttt{chl} \cdot \prod_{i=1}^{m} w_i \cdot g_{n,i} + r_i \cdot g_{n,i} =$$

$$\texttt{chl} \cdot \texttt{F}(w) + \texttt{F}(r).$$

Finally, since each $r_i \leftarrow \mathbb{Z}_q$ and $r_i \cdot g_{1,i}$ is a uniformly random element in $\mathbb{G}$, $\texttt{F}(r) = \texttt{F}(r_1, \ldots, r_m) = \prod_{i=1}^{m} r_i \cdot g_{1,i}, \ldots, \prod_{i=1}^{m} r_i \cdot g_{n,i}$ is distributed uniformly over $\mathbb{G}^n$. □

A.17 Efficiency of the Randomized Fischlin Transform

There are several metrics by which to evaluate the efficiency of the randomized Fischlin transform. We focus on communication and computation complexity.

To understand the communication and computation costs, consider the parameters of the randomized Fischlin transform b, n, t with respect to the security parameter λ and (underlying Σ-protocol's) challenge length ℓ. Here b represents the number of bits output by the random oracle H, n the number of repetitions (of the underlying Σ-protocol) and therefore total proof transcripts output by the prover, and t the number of bits in each test challenge. These quantities are related as follows: $b, r, t = O(\log \lambda)$, $b \leq t \leq \ell$, $br = \omega(\log \lambda)$, and $2^{t-b} = \omega(\log \lambda)$ [34]. The incurred communication cost is therefore polylogarithmic in the security parameter—precisely, n times the size of the underlying Σ-protocol proof transcript (similarly, the verifier will need to run the verification algorithm of the Σ-protocol n times).

With respect to the prover's computational overhead, Fischlin proves that any NISLE ZKPoK (with a non-programming extractor) must query H a number of times that is *at least* superlogarithmic in the security parameter—that is, the computational complexity of *any* NISLE ZKPoK prover is lower-bounded by $\omega(\log \lambda)$ (Proposition 2 in Section 3.4 [34]).[6] Kondi and shelat tighten this

[6] This bound does not capture any efficiency loss due to repeating expensive operations in the underlying Σ-protocol template—it is meant to capture the additional cost incurred by the transform itself.

bound, and demonstrate that the standard (and randomized) Fischlin transform is a factor of $\frac{n}{(n!)^{1/n}}$ worse on average than an optimally efficient prover (Lemma 5.1 and Claim 5.3, respectively [41]). They additionally introduce a "drop-in replacement" for Fischlin's minimal hash output predicate in the form of a hash *collision* predicate (i.e. rather than finding a transcript that maps to a minimal hash output, the prover finds two transcripts that map to the same hash output). If the underlying Σ-protocol has properties called $n+1$-special soundness and n-simulatability, the prover can apply a *multi-collision* predicate to obtain optimal efficiency (Section 5.3 [41]). For comparison, the cut-and-choose method of straight-line extraction due to Pass [47] incurs a minimal computational overhead of λ^2 [41].