

Efficient and Universally Composable Non-Interactive Zero-Knowledge Proofs of Knowledge with Security Against Adaptive Corruptions

Anna Lysyanskaya and Leah Namisa Rosenbloom

Brown University, Providence RI 02906, USA
{anna.lysyanskaya,leah.rosenbloom}@brown.edu

Abstract. Non-interactive zero-knowledge proofs of knowledge (NIZK-PoK) serve as a key building block in many important cryptographic constructions. Achieving universally composable NIZKPoK secure against adaptive corruptions was a long-standing open problem, recently solved by Canetti, Sarkar, and Wang (Asiacrypt'22). This sole known construction requires heavy cryptographic machinery such as correlation-intractable hash functions, and is not ready for use in practice. In this paper, we give constructions of adaptively secure universally composable NIZKPoK in the global random-oracle model; we consider both the programmable and the non-programmable versions of the model. For many practical NIZK proof systems, our constructions incur only a polylogarithmic slowdown factor compared to stand-alone security.

Table of Contents

1	Introduction	3
2	Preliminaries	9
2.1	The Global Random Oracle Model(s)	9
2.2	The $\mathcal{G}_{\text{roRO}}\text{-}\mathcal{F}_{\text{CRS}}$ -hybrid Model	11
2.3	Adaptive Corruptions in the UC Model	12
2.4	The NIZKPoK Ideal Functionality	12
2.5	UC Security with Adaptive Corruptions	13
3	Adaptive Σ-protocols	14
3.1	Σ -protocols	14
3.2	OR-protocols	17
3.3	Requirements for Adaptive Witness Indistinguishability	21
4	Adaptive Straight-Line Compilers	24
5	Adaptive and Universally Composable NIZKPoK	27
5.1	Adaptive UC NIZKPoK are adNIM-SHVZK and adNI-SSS	28
5.2	Adaptive UC NIZKPoK in the $\mathcal{G}_{\text{rpoRO}}$ -hybrid Model	29
5.3	Adaptive UC NIZKPoK in the $\mathcal{G}_{\text{roRO}}\text{-}\mathcal{F}_{\text{CRS}}$ -hybrid Model	31
6	Constructions via the Adaptive Fischlin Transform	35
6.1	Requirements of the Randomized Fischlin Transform	35
6.2	The Adaptive Randomized Fischlin Transform	36
6.3	Adaptive Randomized Fischlin is an Adaptive SLC	39
6.4	Realizing Efficient and Adaptive aUC NIZKPoK from Σ -protocols in the Global ROM(s)	42
7	Practical Adaptive Σ-protocols	43
7.1	Simple Adaptive Σ -protocol Abstraction	43
7.2	Adaptive Proof of Knowledge of Discrete Logarithm	45
7.3	Adaptive Proof of Knowledge of Equality of n Representations ..	45

1 Introduction

An adaptive adversary attacking a cryptographic protocol can decide on the fly which protocol participant(s) it wants to corrupt. As a result, the adversary gains access to the memory of a device that is currently running, or has previously run, the protocol. In the event that the adversary does not corrupt a party, the protocol must protect this party’s security. Achieving security from such adaptive corruptions is a notoriously difficult problem.

Non-interactive zero-knowledge proofs of knowledge (NIZKPoK) are an important building block in cryptographic constructions. For example, NIZKPoK allow honest protocol participants to prove they have formatted their protocol messages correctly, and catch any adversary who is trying to deviate maliciously from a protocol specification. This technique transforms protocols secure in the “honest-but-curious” model (in which protocol participants are guaranteed to act according to the protocol) into those secure in the more realistic “malicious” model (in which the adversary can issue adaptive corruptions and execute arbitrary code). NIZKPoK that are safe to deploy in such a complex cryptographic system must have *composable security*: they must maintain security properties when composed concurrently with other protocols.

Thus, constructing a non-interactive zero-knowledge proof system that can withstand adaptive corruptions in the *universal composition* (UC) framework of Canetti [14] is a natural and well-motivated problem. Obtaining *adaptive and composable* NIZKPoK was a long-standing open problem until Canetti, Sarkar, and Wang [19] developed a compiler that leverages UC non-interactive commitments, Camenisch and Damgård’s commitment-based straight-line extractor [5], and a correlation intractable hash function to obtain adaptive UC NIZKPoK from standard assumptions. In particular, Canetti et al. consider Σ -protocols in the $\mathcal{F}_{\text{NIZCOM}}$ model, which assumes the first message of the Σ -protocol is a UC non-interactive commitment, instantiated using equivocal commitments and CCA-2 secure public-key encryption with oblivious ciphertext sampling. Camenisch and Damgård’s extractor adds an additional $O(\lambda)$ “commit-and-open” operations for security parameter λ , and correlation intractable hash functions typically rely on heavy-weight primitives like fully homomorphic encryption or indistinguishability obfuscation.

In this paper, we show how to obtain *efficient and adaptive UC NIZKPoK* from any Σ -protocol with a natural adaptivity property in the global random-oracle model. For most practical Σ -protocols—i.e. those on the cusp of widespread adoption in practice—our construction does not require any additional cryptographic machinery; rather, it is as efficient as the Σ -protocol under the randomized Fischlin transform [24,27], which creates a multiplicative overhead for the prover that is only superlogarithmic in λ . By treating the random oracle (RO) as a global subroutine in the universal composition with global subroutines (UCGS) model [1], our adaptive NIZKPoK retain composability even when the global RO is shared among different sessions and protocols, as is likely in practice.

Adaptive Σ -protocols. A standard Σ -protocol [23] 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 $(x, w) \in R$. The “three-move” form of a Σ -protocol is defined as follows: a prover P sends a verifier V a first message com , V sends P a uniformly random challenge ch1 , P responds with a value res , and V decides whether or not to “accept” (output 1) based on the proof transcript $(x, \text{com}, \text{ch1}, \text{res})$. The “zero-knowledge proof system” piece of the definition implies three properties: *completeness*, *special honest-verifier zero-knowledge* (SHVZK), and *special soundness* (SS). The completeness property says that if P forms a proof using the three-move form on input $(x, w) \in R$, then V always accepts. The SHVZK property states there must exist a *simulator* algorithm SimProve that, on input the statement *and the challenge in advance*, can produce a proof that looks statistically close to that of a “real” prover without using a witness. Finally, the SS property implies an *extractor* algorithm Extract that can compute a witness w from any two proofs of a statement x with the same first message but different challenges.

Our key insight is that P is a *probabilistic* Turing machine, and its random coins r determine the value of the first message com . This randomness is the crux of the zero-knowledge property—it is the only information hidden from the adversary during the SHVZK experiment. In the *adaptive* SHVZK experiment, the adversary can corrupt P after P has already issued proofs, revealing the entirety of P ’s random tape. We therefore consider an “adaptive” Σ -protocol to be one that has an additional simulator algorithm, SimRand , that uses information from SimProve and the witness that was supposedly used to compute the proof to generate convincing-looking coins for P ’s random tape. Many popular Σ -protocols are adaptive according to this definition.

Universal Composition with Global Subroutines. The security experiment in the UC framework [14,1] 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” half of the security experiment, the honest parties form their outputs according to the protocol Π , and the adversary machine is the traditional notion of a cryptographic protocol adversary \mathcal{A} (i.e. one that executes arbitrary instructions). In the “ideal-world” experiment, honest parties are just placeholder “dummies” who pass all of their inputs to an ideal functionality \mathcal{F} , which acts according to an “ideal” version of the protocol. The adversary \mathcal{A} is replaced with an “ideal adversary” \mathcal{S} , who acts like \mathcal{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 \mathcal{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} .

In the original version of the framework [14], the “test session” of a protocol must be *subroutine respecting*—its subroutines cannot process inputs coming from other sessions or protocols. Since it is reasonable to expect protocols in real-world applications to share a common reference string (CRS) or RO, subsequent versions of the UC framework such as joint-state UC (JUC) [18] and general UC (GUC) [15] tweak the model to allow this feature. The GUC model in particular is designed to incorporate *global functionalities* \mathcal{G} that can be accessed by *any* party in the security experiment. However, Badertscher, Canetti, Hesse, Tackmann, and Zikas [1] observe subtle inconsistencies in the GUC model stemming from the (unconstrained) environment’s ability to spawn parties with arbitrary session identifiers (see Appendix A [1]). Rather than relax the constraints of the model, Badertscher et al. leverage the “shell-and-body” construct of the original UC framework [14] to create a UC with global subroutines (UCGS) model. In the UCGS model, the *global subroutine protocol* \mathcal{G} is wrapped into a joint body with the test protocol session π by a “shell” protocol M . The shell M processes all communications in and out of π and \mathcal{G} and ensures that the combined entity $M[\pi, \mathcal{G}]$ is subroutine respecting, even though π and \mathcal{G} are not.

In this work, the ideal functionality \mathcal{F} is the ideal functionality $\mathcal{F}_{\text{aNIZK}}$ for adaptive NIZKPoK. The global subroutines are the restricted programmable observable global RO $\mathcal{G}_{\text{rpoRO}}$ of Camenisch, Drijvers, Gagliardini, Lehmann, and Neven [6] and the restricted observable (non-programmable) global RO $\mathcal{G}_{\text{roRO}}$ of Canetti, Jain, and Scafuro [15].

Adaptive Straight-Line Compilers. A straight-line compiler (SLC) [29] is an algorithm 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 Π_R^{SLC} for R in the random-oracle model (ROM). Recall that a proof system is straight-line extractable if the challenger in the security experiment can obtain the two proofs needed to run the **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 random oracle (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 **Extract** algorithm can immediately compute a witness for the prover’s statement. The randomized Fischlin transform [24,27] sets the proof repetition parameters such that the prover is guaranteed (with probability that is overwhelming in the security parameter) to issue two such transcripts to the RO. Other forms of straight-line extraction, such as the aforementioned commit-and-open construction due to Camenisch and Damgård [5] work in the plain model, but require the prover to couple each repetition with a cryptographic commitment, creating substantial computational overhead.

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

Π_R^{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 17-19) are both *necessary* and *sufficient* to obtain adaptive UC NIZKPoK in the global ROM.

Theorem 2 (Informal). *If a protocol Π creates adaptive UC NIZKPoK in the ROM, then it must have the properties from Definitions 17-19.*

Proof Insight. We show by contradiction that if there exists some distinguisher \mathcal{A} that can win the adaptive NIM-SHVZK or NI-SSS games (that is, if Π does not have the properties from Definitions 17-19), we can construct a reduction \mathcal{B} that can distinguish between the real- and ideal-world adaptive UC experiments. The key observation is that when \mathcal{B} uses the NIM-SHVZK adversary as a black-box (while playing the role of the adversarial environment \mathcal{Z} against the UC challenger), \mathcal{B} is able to corrupt parties in the UC experiment in order to simulate \mathcal{A} 's view exactly as it would expect from the NIM-SHVZK challenger. If the UC challenger is running the ideal-world experiment, \mathcal{B} would be talking to the **Corrupt** interface of our adaptive NIZKPoK ideal functionality (Definition 6), which returns an output according to the simulator's **SimRand** algorithm; in the real world experiment, \mathcal{B} would get the prover's actual random tape.

Theorem 3 (Informal). *Given any adaptive Σ -protocol Σ_R and adaptive straight-line compiler, we construct adaptive UC NIZKPoK in the programmable global ROM (i.e. assuming the global RO can be programmed by the security experiment).*

Proof Insight. Given an adversary \mathcal{A} that can distinguish the real- from ideal-world UC experiments, we construct a series of hybrids that use \mathcal{A} as a black-box to win the adaptive NIM-SHVZK and adaptive NI-SSS games where the corruption operations are handled similarly to the proof of Theorem 2. Since this proof operates in the programmable global ROM, the NIM-SHVZK and NI-SSS challengers are simulating proofs according to the traditional (programming-based) **SimProve** algorithm, with the added constraint that there must exist a corresponding **SimRand** algorithm that, given the simulated proof and the witness supposedly used to compute it, outputs convincing-looking randomness for the prover's random tape.

While the programmable global ROM is convenient, there are subtle modeling differences between the observable-only global RO $\mathcal{G}_{\text{roRO}}$'s observation interface, which reveals the illegitimate queries of the environment in the UC experiment to anyone who asks (including the environment), and the localized programming interface of the programmable global RO $\mathcal{G}_{\text{rpoRO}}$, which is only revealed to participants in a particular protocol session [29,6]. Moreover, recent works in obtaining adaptive [19] and efficient [25] UC proof systems have operated in some version of the global RO-CRS hybrid model, indicating it is deserving of further exploration. We therefore provide a construction of adaptive UC NIZKPoK in the observable-only (non-programmable) global RO-CRS hybrid model.

Theorem 4 (Informal). *Given any adaptive Σ -protocol Σ_R , any adaptive straight-line compiler, and an adaptive version of Lysyanskaya and Rosenbloom’s OR-protocol compiler [29], we construct adaptive UC NIZKPoK in the observable-only (non-programmable) global RO-CRS hybrid model.*

Proof Insight. This proof proceeds similarly to the proof above, except the simulator can no longer program the RO. We use a technique similar to that of Lysyanskaya and Rosenbloom [29], in which the simulator has a special witness, the trapdoor to the local CRS_s for protocol session s , that the prover uses to show that either it has a valid witness w for the statement x , or else it has the secret trapdoor trap_s to CRS_s . In our construction, the compiler relies on the `SimRand` functionality of the underlying adaptive OR-protocol and a supplementary algorithm, `RealToSim`, to provide convincing randomness for either witness on the prover’s tape after the prover has been corrupted. We formalize this intuition as an updated version of witness indistinguishability for the adaptive corruption setting, and demonstrate that it can be achieved as long as both Σ -protocols underlying the OR-protocol are adaptive special honest-verifier zero-knowledge and have a new property we call *randomness equivocability*.

Theorem 1 (Informal). *Given any adaptive OR-protocol Σ_{OR} based on Σ -protocols Σ_0 and Σ_1 , if Σ_0 and Σ_1 are both adaptive special honest-verifier zero-knowledge and randomness equivocable, then Σ_{OR} is adaptive witness indistinguishable.*

Proof Insight. Recall that as part of an OR-protocol, the proof of a chosen statement x_b is computed according to the $\Sigma_b.\text{Prove}$ algorithm on a “real” witness w_b , while the proof of the other statement $x_{\bar{b}}$ is computed according to the simulator algorithm, $\Sigma_{\bar{b}}.\text{SimProve}$. In order to maintain a notion of witness indistinguishability in the adaptive corruption setting, the simulator must be able to make the prover’s randomness look as if it formed either proof honestly. To make the simulated proof look “real,” the simulator can use $\Sigma_{\bar{b}}.\text{SimRand}$ to produce “real”-looking coins on the prover’s tape. However, note that this functionality is not sufficient to take a “real”-looking proof and produce the simulator’s random coins, such that the adversary is convinced it was produced according to $\Sigma_b.\text{SimProve}$. We introduce a second property, randomness equivocability (RE), which uses the algorithm `RealToSim` in order to make a “real” proof appear simulated. We then prove via a sequence of two reductions that, as long as the underlying Σ -protocols have both the adSHVZK and RE properties respectively, the OR-protocol based on those Σ -protocols are adaptive witness indistinguishable.

Finally, we extend the randomized Fischlin transform in Definition 26 by adding a `SimRand` functionality, and show the extended version meets our definition of an adaptive SLC. This implies we can efficiently transform any adaptive Σ -protocol into adaptive UC NIZKPoK in both global ROMs.

Theorem 5 (Informal). Our extended randomized Fischlin transform [24,27] is an adaptive straight-line compiler; that is, it preserves the security properties of adaptive Σ -protocols under transformation.

Proof Insight. In order for the adaptive version of the transform to provide a convincing version of the prover’s random tape, the **SimRand** algorithm must produce an acceptable version of the challenge selection step—one that agrees with both the simulated proofs and, in the case of witness indistinguishability (the property required for our $\mathcal{G}_{\text{roRO}}\text{-}\mathcal{F}_{\text{CRS}}$ -hybrid model construction), any proofs that the adversary can generate and query to the RO itself using the real witness. Our extended simulator proceeds according to Kondi and shelat’s randomized version of the Fischlin transform [27] up until it is time for the simulator to output the prover’s random tape. To construct the tape, the **SimRand** algorithm of the transform samples fresh randomness from its random tape for the prover’s challenges, computing proofs using the real witness (which it obtains upon corruption) and querying the transcripts to the RO until it reaches an output string 0^b , at which point it inserts the randomness used to sample the challenge of the simulated proof. The first-message randomness of the simulated proof is computed according to the **SimRand** algorithm of the underlying Σ -protocol; the first messages for all of the other proofs are drawn freshly at random. Because the prover in Kondi and shelat’s transform samples the challenges at random from the challenge space, the adversary cannot distinguish the “real” prover’s challenge selection path from the simulator’s.

Applications. NIZKPoK are widely used in group [12,5], blind [26], threshold [3], aggregate [27], and multi- [3,22] signatures, cryptographic shuffles [36,30] and accumulators [2,10], anonymous networks [20], credentials [9], e-cash [8], e-token [7], and voting [32,30,35], distributed ledgers [32], verifiable secret sharing [35] and encryption schemes [5,11] that are secure against adaptively chosen ciphertext attacks (CCA security). We demonstrate that many of the Σ -protocols used as core building blocks in these constructions, including proofs of knowledge of discrete logarithm [34] and proofs of knowledge of n representations of discrete logarithm [13,11], qualify as adaptive Σ -protocols, and can therefore be efficiently transformed into adaptive UC NIZKPoK in the global ROM. The result is an immediate and significant boost in the security potential of all of the above real-world systems.

Theorems 6-7 (Informal). *Many common Σ -protocols are adaptive Σ -protocols, and can therefore be converted to efficient and adaptive UC NIZPoK in the global ROM using our techniques.*

Proof Insight. We observe that the abstract treatment of identification schemes from linear function families due to Hauck, Kiltz, and Loss [26] closely resembles the structure of common Σ -protocols such as proofs of knowledge of discrete logarithm [34] and n representations of discrete logarithm [13,11]. We remodel this abstraction to include **SimProve**, **SimRand**, and **Extract** algorithms, resulting in a natural class of adaptive Σ -protocols.

Organization. In Section 2, we introduce the various oracles, models, and security definitions we will use in our constructions. Section 3 contains formal definitions of adaptive Σ -protocols and adaptive SLCs. We prove in Section 5 that the security guaranteed by adaptive SLCs is both necessary and sufficient to create adaptive UC NIZKPoK in the programmable global ROM, and sufficient along with an ideal CRS functionality [29] in the (non-programmable) global RO-CRS hybrid model. In Section 6 we extend the randomized Fischlin transform [24,27] such that it satisfies our definition of an adaptive SLC, and can therefore create efficient and adaptive UC NIZKPoK from any adaptive Σ -protocol. Finally in Section 7, we prove that many common Σ -protocols are adaptive, concluding that efficient and adaptive UC NIZKPoK are realizable for a variety of real-world systems.

2 Preliminaries

In this section, we give preliminary definitions of the global RO (Section 2.1) and RO-CRS hybrid (Section 2.2) models we will use in our constructions, as well as a specification of the adaptive corruption mechanism (Section 2.3), the ideal adaptive NIZKPoK functionality $\mathcal{F}_{\text{aNIZK}}$ (Section 2.4), and finally adaptive security in the UCGS model (Section 2.5).

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. [17] and the restricted *programmable* observable global ROM of Camenisch et al. [6]. Recall from the introduction that NIZKPoK in the ROM traditionally require a proof simulator algorithm `SimProve` that *programs* the outputs of the RO, and an extractor algorithm `Extract` that *observes* the adversary’s RO queries. While making the global RO programmable is easier from a construction standpoint, the non-programmable model is closer to the intended vision of a truly “global” RO that cannot be edited or controlled by any one entity. We highlight and discuss the differences between the two models below.

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 [4]. The “global” designation refers to the fact that there exists a single instance of the oracle for all sessions of a protocol, and potentially across protocols (as opposed to one functionality per protocol session, as in the standard UC model [14]). 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 $\mathcal{G}_{\text{roRO}}$ [17,6] records all “illegitimate” queries made for a session s by parties with an $\text{sid} \neq s$, 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— $\mathcal{G}_{\text{roRO}}$ can release it to anyone who asks [6].

Definition 1 (Observable Global RO $\mathcal{G}_{\text{roRO}}$). [17,6] *The observable global RO $\mathcal{G}_{\text{roRO}}$ is a tuple of algorithms (**Query**, **Observe**) defined over an output length ℓ and an initially empty list of queries \mathcal{Q} :*

- $v \leftarrow \text{Query}(x)$: 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 .
- $\mathcal{Q}_s \leftarrow \text{Observe}(s)$: If a list \mathcal{Q}_s of illegitimate queries for s does not yet exist, set $\mathcal{Q}_s = \perp$. Return \mathcal{Q}_s .

The *programmable* version $\mathcal{G}_{\text{rpoRO}}$ [6] builds on the observable-only functionality with an additional interface, the **IsProgrammed** interface, which reveals the programmed entries for a session s to any parties in the same session. Again 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). It is unclear to date how this distinction affects the security of protocols under composition in the global ROM, if at all. In the meantime, we recall in the next section the $\mathcal{G}_{\text{roRO}}$ - \mathcal{F}_{CRS} -hybrid model, which will allow us to obtain adaptive UC NIZKPoK without programming the global RO.

Definition 2 (Restricted Programmable Observable Global RO $\mathcal{G}_{\text{rpoRO}}$). [6] *The restricted programmable observable global random oracle $\mathcal{G}_{\text{rpoRO}}$ is a tuple of algorithms (**Query**, **Observe**, **Program**, **IsProgrammed**) defined over an output length ℓ and initially empty lists \mathcal{Q} (queries) and **prog** (programmed queries):*

- $v \leftarrow \text{Query}(x)$: Same as Definition 1 above.
- $\mathcal{Q}_s \leftarrow \text{Observe}(s)$: Same as Definition 1 above.
- $\{0, 1\} \leftarrow \text{Program}(x, v)$: 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 **prog** and output 1.
- $\{0, 1\} \leftarrow \text{IsProgrammed}(x)$: Parse x as (s, x') . If the caller’s SID $\neq s$, output \perp . Otherwise if $x \in \text{prog}$, output 1. Otherwise, output 0.

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

In the observable-only $\mathcal{G}_{\text{roRO}}$ -hybrid model, the `SimProve` algorithm has no additional power over a regular prover, since it cannot program $\mathcal{G}_{\text{roRO}}$. Thus, NIZKPoK in the plain $\mathcal{G}_{\text{roRO}}$ -hybrid model are impossible [17,6,33,16,15]. To work around this impossibility result and construct UC NIZKPoK while avoiding the session-localized `IsProgrammed` interface of the programmable global RO $\mathcal{G}_{\text{roRO}}$, Lysyanskaya and Rosenbloom introduce the $\mathcal{G}_{\text{roRO}}\text{-}\mathcal{F}_{\text{CRS}}$ -hybrid model [29], where \mathcal{F}_{CRS} is the (local) ideal common reference string (CRS) functionality.

In the real-world execution of the $\mathcal{G}_{\text{roRO}}\text{-}\mathcal{F}_{\text{CRS}}$ -hybrid model, \mathcal{F}_{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}_{CRS} , and can generate CRS_s for each session s with a secret trapdoor trap_s . Provers in the $\mathcal{G}_{\text{roRO}}\text{-}\mathcal{F}_{\text{CRS}}$ -hybrid model prove via an OR-type Σ -protocol [21,23] that *either* they know a real witness to a statement, *or* they know the trapdoor trap_s to 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 $\text{CRS}_s \in L_S$, it must be overwhelmingly difficult to generate a trap_s such that $S(\text{CRS}_s, \text{trap}_s) = 1$ [29]. The relation S must additionally be *Σ -friendly*: it must have a corresponding *efficient* Σ -protocol Σ_S . Proofs in the $\mathcal{G}_{\text{roRO}}\text{-}\mathcal{F}_{\text{CRS}}$ -hybrid model are therefore well-specified as OR-protocols $\Sigma_{R \vee S}$ over the relation $R \vee S$, where R is the relation of the original Σ -protocol Σ_R , and S is the samplable-hard relation underlying Σ_S . Formal definitions of the ideal functionality and context-friendly properties of \mathcal{F}_{CRS} are given below.

Definition 3 (Samplable-Hard Relation). [29] *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 $|x| = \text{poly}(\lambda)$.*
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 \mathcal{A} can find a valid witness given only the statement is negligible. Formally, for all PPT \mathcal{A} ,*

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

Definition 4 (Σ -Friendly Relation). [29] *A Σ -friendly relation S is a binary \mathcal{NP} relation with a corresponding efficient Σ -protocol Σ_S .*

Definition 5 (CRS Ideal Functionality). [29] *The ideal functionality \mathcal{F}_{CRS} of a common reference string (CRS) for a particular CRS generation mechanism `GenCRS` is defined as follows.*

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

2.3 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 $(\text{Corrupt}, P)$ to the control function [14].¹

In the real world, the control function passes the message $(\text{Corrupt}, P)$ to the traditional adversary \mathcal{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 \mathcal{A} .²

In the ideal world, the control function passes the message $(\text{Corrupt}, P)$ 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 \mathcal{A} in the real world, and \mathcal{S} in the ideal world. We call any protocol that satisfies the UC security definition (given in the next section) with adaptive corruptions *adaptive UC*, or aUC for short.

2.4 The 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 [14]. Upon receiving instructions from the ideal adversary \mathcal{S} and setting up any necessary parameters, $\mathcal{F}_{\text{NIZK}}$ proves statements for honest parties using the SimSetup and SimProve algorithms (which do not take witnesses as input) and verifies proofs using the Extract algorithm. If the algorithms from \mathcal{S} do not function

¹ 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 $(\text{Corrupt}, P)$ to go through after P was initialized. In the adaptive setting, corruption messages are allowed at any point during the security experiment.

² Coins from the random tape in the UC model [14] 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).

as $\mathcal{F}_{\text{aNIZK}}$ intends, for instance if **SimProve** produces proofs that do not verify or **Extract** outputs invalid witnesses, $\mathcal{F}_{\text{aNIZK}}$ outputs **Fail**. During its processing, $\mathcal{F}_{\text{aNIZK}}$ stores information about the proofs it has computed on each dummy party’s behalf. \mathcal{S} retrieves this information via $\mathcal{F}_{\text{aNIZK}}$ ’s **Corrupt** interface whenever an honest party is corrupted by the environment.

Definition 6 (Adaptive NIZKPoK Ideal Functionality). *The ideal functionality $\mathcal{F}_{\text{aNIZK}}$ of an adaptive non-interactive zero-knowledge proof of knowledge (adaptive NIZKPoK) for a particular session s is defined as follows.*

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

Prove: Upon receiving a request (Prove, x, w) from a party $P = (\text{pid}, \text{sid})$, check that $\text{sid} = s$, $P \notin C$, 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 $\text{Verify}(x, \pi) = 1$. If it doesn’t, output **Fail**. Otherwise, record the tuple $(\text{Proof}, P, x, w, \pi, z_s, z_\pi)$, where z_π is any auxiliary output of the **SimProve** algorithm. Output $(\text{Proof}, P, x, \pi)$.

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

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

2.5 UC Security with Adaptive Corruptions

At a high level, a protocol Π qualifies as adaptive UC (aUC) 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 \mathcal{A}) from the ideal-world experiment (with $\mathcal{F}_{\text{aNIZK}}$ and \mathcal{S}).

In the UC with global subroutines (UCGS) model, Π UC-emulates \mathcal{F} in the presence of a global subroutine \mathcal{G} if $\mathbb{M}[\Pi, \mathcal{G}]$ UC-emulates $\mathbb{M}[\mathcal{F}, \mathcal{G}]$, where \mathbb{M} is the “shell” wrapper discussed in the introduction. Badertscher et al. show in Proposition 3.4 [1] 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 $\mathsf{M}[\Pi, \mathcal{G}]$ (resp. $\mathsf{M}[\mathcal{F}, \mathcal{G}]$) 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 Section 2.1 that both $\mathcal{G}_{\text{rpoRO}}$ and $\mathcal{G}_{\text{roRO}}$ qualify as global subroutines, and as $\mathcal{G}_{\text{rpoRO}}$ and $\mathcal{G}_{\text{roRO}}$ are the only global subroutines in our experiments, Π and $\mathcal{F}_{\text{anIZK}}$ are $\mathcal{G}_{\text{rpoRO}}$ - and $\mathcal{G}_{\text{roRO}}$ -subroutine respecting. Therefore, without loss of generality, we consider our UC experiments “in the presence of global subroutines” $\mathcal{G}_{\text{rpoRO}}$ and $\mathcal{G}_{\text{roRO}}$ —or in the $\mathcal{G}_{\text{rpoRO}}$ - and $\mathcal{G}_{\text{roRO}}$ -hybrid models for short—and make no further reference to M (for details, see Section 3 in Badertscher et al. [1]).

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}_{RO} , $\mathcal{G}_{\text{rpoRO}}$, $\mathcal{G}_{\text{roRO}}$, and \mathcal{F}_{CRS}) as needed throughout the paper.

Definition 7 (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 \mathcal{A} , there exists an ideal adversary \mathcal{S} efficient in expectation such that for all efficient environments \mathcal{Z} that can issue adaptive corruptions,*

$$\text{IDEAL}_{\mathcal{F}, \mathcal{S}, \mathcal{Z}}^{\mathcal{G}}(1^\lambda, \text{aux}) \approx_c \text{REAL}_{\Pi, \mathcal{A}, \mathcal{Z}}^{\mathcal{G}, *}(1^\lambda, \text{aux}),$$

where \mathcal{G} is a global subroutine, 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

In this section, we formalize the notion of adaptive Σ -protocols.

3.1 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. In the traditional definitions of Σ -protocols [23, 29], the contents of P ’s random tape are not explicitly captured in the inputs and outputs of the Σ -protocol algorithms. However, P ’s randomness is vital to the security experiment—it is the only piece of information hidden from the adversary (recall the adversary in the special honest-verifier zero-knowledge game is allowed to query for proofs of statements using witnesses of its choosing), and in the adaptive corruption setting, the adversary will have access to this randomness after proofs have been generated. Therefore, rather than keeping the randomness necessary to compute proofs implicit in P ’s random tape, we make it an explicit quantity, denoted r .

The basic three-move form of a Σ -protocol is generally described as an interactive “protocol template” τ [23,29]. The algorithmic version of the protocol template definition [29] consists of the following algorithms: τ .Setup, τ .Commit, τ .Challenge, τ .Respond, and τ .Decision. 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.

Definition 8 (Adaptive Σ -protocol Template). *The adaptive Σ -protocol template for a relation R is a tuple of efficient algorithms $\tau = (\text{Setup}, \text{Commit}, \text{Challenge}, \text{Respond}, \text{Decision})$, defined as follows.*

- $\text{ppm} \leftarrow \text{Setup}(1^\lambda)$: *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 .*
- $\text{com} \leftarrow \text{Commit}(\text{ppm}, x, w, r)$: *Given statement x , witness w , and randomness $r \leftarrow_{\S} \{0, 1\}^{\lambda_P}$, P sends V a message `com`.*
- $\text{chl} \leftarrow \text{Challenge}(\text{ppm}, x, \text{com})$: *V sends P a random ℓ -bit string `chl`.*
- $\text{res} \leftarrow \text{Respond}(\text{ppm}, x, w, \text{com}, \text{chl})$: *P sends V a reply `res`.*
- $\{0, 1\} \leftarrow \text{Decision}(\text{ppm}, x, \text{com}, \text{chl}, \text{res})$: *V decides whether to output 1 (accept) or 0 (reject) based on the input $(\text{ppm}, x, \text{com}, \text{chl}, \text{res})$.*

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

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). Lysyanskaya and Rosenbloom formalized a Σ -protocol for relation R as a tuple of algorithms, $\Sigma_R = (\text{Setup}, \text{Prove}, \text{Verify}, \text{SimSetup}, \text{SimProve}, \text{Extract})$, that capture the requirements of the three-move form as well as the correctness and security properties. 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_{sim} , 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. 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 by the adversary. We will show

in Section 7 that several widely-used instantiations of Σ -protocols are adaptive according to this definition.

Finally, adaptive Σ -protocols require an adaptive version of special honest-verifier zero-knowledge (SHVZK), in which the adversary should not be able to tell the difference between the outputs of **Prove** and **SimProve** or between the randomness r of a real prover and the output of **SimRand**.

Definition 9 (Adaptive Σ -protocol). An adaptive Σ -protocol for a relation R based on adaptive protocol template τ (Definition 8) is a tuple of efficient procedures $\Sigma_{R,\tau} = (\text{Setup}, \text{Prove}, \text{Verify}, \text{SimSetup}, \text{SimProve}, \text{SimRand}, \text{Extract})$, defined as follows.

- $\text{ppm} \leftarrow \text{Setup}(1^\lambda)$: Given a security parameter 1^λ , invoke $\tau.\text{Setup}(1^\lambda)$ to obtain the public parameters ppm , which include randomness security parameters λ_P and λ_{Sim} for the prover and simulator, respectively.
- $\pi \leftarrow \text{Prove}(\text{ppm}, x, w, r, (\text{ppm}, x))$: 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 $r \leftarrow_{\S} \{0, 1\}^{\lambda_P}$. P and V run $\tau.\text{Commit}$, $\tau.\text{Challenge}$, and $\tau.\text{Respond}$. Output $\pi = (\text{com}, \text{chl}, \text{res})$.
- $\{0, 1\} \leftarrow \text{Verify}(\text{ppm}, x, \pi)$: Given a proof π for statement x , parse π as $(\text{com}, \text{chl}, \text{res})$ and output the result of running $\tau.\text{Decision}$ on input $(x, \text{com}, \text{chl}, \text{res})$. **Verify** must satisfy the completeness property (Definition 10).
- $(\text{ppm}, z_{\text{Sim}}) \leftarrow \text{SimSetup}(1^\lambda)$: Generate ppm and a general simulation trapdoor z_{Sim} . Together, **SimSetup**, **SimProve**, and **SimRand** must satisfy the adaptive special honest-verifier zero-knowledge property (Definition 11).
- $(\pi, z_\pi) \leftarrow \text{SimProve}(\text{ppm}, z, x, \text{chl}, r_{\text{Sim}})$: Given public parameters ppm , trapdoor z_{Sim} , statement x , challenge chl , and randomness $r_{\text{Sim}} \leftarrow \{0, 1\}^{\lambda_{\text{Sim}}}$, produce a proof $\pi = (\text{com}, \text{chl}, \text{res})$ and proof trapdoor z_π .
- $r \leftarrow \text{SimRand}(\text{ppm}, z_{\text{Sim}}, z_\pi, x, \pi, w, r_{\text{Sim}})$: Given public parameters ppm , general trapdoor z_{Sim} , proof trapdoor z_π , proof π for statement x , a witness w such that $R(x, w) = 1$, and randomness r_{Sim} , produce randomness r .
- $w \leftarrow \text{Extract}(\text{ppm}, x, \pi, \pi')$: Given two proofs $\pi = (\text{com}, \text{chl}, \text{res})$ and $\pi' = (\text{com}, \text{chl}', \text{res}')$ for a statement x , output a witness w . **Extract** must satisfy the two-special soundness property (Definition 12).

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

Definition 10 (Completeness). A Σ -protocol Σ_R for relation R is complete if for all $(x, w) \in R$ and $\pi \leftarrow \Sigma_R.\text{Prove}((x, w), x)$, $\Sigma_R.\text{Verify}(x, \pi) = 1$.

Definition 11 (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

SimSetup, **SimProve**, and **SimRand** such that for any security parameter λ , any adversary (resp. any PPT adversary) \mathcal{A} , and a bit $b \leftarrow_{\$} \{0, 1\}$, there exists some negligible function negl such that $\Pr[b' = b] \leq \frac{1}{2} + \text{negl}(\lambda)$, where b' is the result of running the game $\text{adSHVZK}_{\mathcal{A}, \Sigma_R}(1^\lambda, b)$ from Figure 1. We say \mathcal{A} wins the adSHVZK game if $\Pr[b' = b] > \frac{1}{2} + \text{negl}(\lambda)$.

$\text{adSHVZK}_{\mathcal{A}, \Sigma_R}(1^\lambda, 0)$: REAL	$\text{adSHVZK}_{\mathcal{A}, \Sigma_R}(1^\lambda, 1)$: IDEAL
1 : $\text{ppm} \leftarrow \Sigma_R.\text{Setup}(1^\lambda)$	1 : $(\text{ppm}, z_s) \leftarrow \Sigma_R.\text{SimSetup}(1^\lambda)$
2 : $(x, w, \text{st}) \leftarrow \mathcal{A}(1^\lambda, \text{ppm})$	2 : $(x, w, \text{st}) \leftarrow \mathcal{A}(1^\lambda, \text{ppm})$
3 : if $R(x, w) = 1$:	3 : if $R(x, w) = 1$:
4 : $r \leftarrow_{\$} \{0, 1\}^{\lambda_P}$	4 : $r_{\text{Sim}} \leftarrow_{\$} \{0, 1\}^{\lambda_{\text{Sim}}}$
5 : $\text{chl} \leftarrow \{0, 1\}^\ell$	5 : $\text{chl} \leftarrow \{0, 1\}^\ell$
6 : $\pi \leftarrow \Sigma_R.\text{Prove}((x, w, r), (x, \text{chl}))$	6 : $(\pi, z_\pi) \leftarrow \Sigma_R.\text{SimProve}(z_{\text{Sim}}, x, \text{chl}, r_{\text{Sim}})$
7 : else :	7 : $r \leftarrow \Sigma_R.\text{SimRand}(z_{\text{Sim}}, z_\pi, x, \pi, w, r_{\text{Sim}})$
8 : $\pi \leftarrow \perp$	8 : else :
9 : $b' \leftarrow \mathcal{A}(\text{st}, \pi, r)$	9 : $\pi \leftarrow \perp$
10 : return b'	10 : $b' \leftarrow \mathcal{A}(\text{st}, \pi, r)$
	11 : return b'

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

Definition 12 (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 \mathcal{A} ,

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

where **SS** is the special soundness game described in Figure 2. We say \mathcal{A} wins the **SS** game if $\Pr[\text{Fail} \leftarrow \text{SS}_{\mathcal{A}, \Sigma_R}(1^\lambda)] > \text{negl}(\lambda)$.

3.2 OR-protocols

An OR-protocol [23,29] is a special type of 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 $\Phi = (\pi_0, \pi_1, \text{CHL})$, where one of the proofs π_b is “real” (computed according to the $\Sigma_b.\text{Prove}$ algorithm) and the other is simulated (according to the $\Sigma_{\bar{b}}.\text{SimProve}$ algorithm).

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

Fig. 2. Two-Special Soundness (SS) Game.

Zero-knowledge OR-protocols necessarily prevent the adversary from learning anything about the witness other than the fact that the combined relation $R_{OR} = R_0 \vee R_1$ is satisfied. An OR-protocol Σ_{OR} based on Σ -protocols Σ_0 and Σ_1 must therefore have a property called *witness indistinguishability*, which follows from the special honest-verifier zero-knowledge property of Σ_0 and Σ_1 [23]. At a high level, witness indistinguishability says that even when an adversary \mathcal{A} is allowed to choose the witness w_b used to compute the proof, the best \mathcal{A} can do in distinguishing whether the prover used w_b or $w_{\bar{b}}$ to compute the proof is only negligibly better than a random guess.

As part of our *adaptive* OR-protocol formalization, which we will need for the proof of Theorem 4, we introduce an adaptive version of the witness indistinguishability property, which states (informally) that an adversary cannot tell whether a prover used w_b or $w_{\bar{b}}$ 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. Since the traditional OR-protocol simulator computes both halves π_b and $\pi_{\bar{b}}$ of the proof according to $\Sigma_b.\text{SimProve}$ and $\Sigma_{\bar{b}}.\text{SimProve}$, respectively, in order to make the simulated random tape look "real," $\Sigma_{OR}.\text{SimRand}$ must produce real-looking randomness for the proof π_b corresponding to the witness w_b that was supposedly used to run $\Sigma_{OR}.\text{Prove}$. Note, however, that the existence of the **SimProve** and **SimRand** algorithms alone does not satisfy our intuition of adaptive witness indistinguishability, which requires the challenger to issue a *real proof* that, once the prover's randomness is revealed, could have been computed according to either w_b or $w_{\bar{b}}$.

We realize the adaptive witness indistinguishability property, from Σ -protocols that satisfy the *randomness equivocability* property we define below.

Let **RealToSim** be an algorithm that takes as input a proof π for statement x computed according to $\Sigma.\text{Prove}$ using witness w and randomness r , and outputs simulated random coins r_{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 $w_{\bar{b}}$), we say that a Σ -protocol is randomness equivocable if for all adversaries \mathcal{A} , \mathcal{A} cannot tell the difference between a simulated proof accompanied by the simulator’s actual random tape r_{sim} and a real proof accompanied by the output of `RealToSim`. We formalize this property below.

Definition 13 (Randomness Equivocability). *A Σ -protocol Σ_R for relation R is randomness equivocable (RE) if there exist algorithms `SimSetup`, `SimProve`, `SimRand`, and `RealToSim` such that Σ_R is adaptive special honest-verifier zero-knowledge (Definition 11) and for any security parameter λ , any adversary (resp. any PPT adversary) \mathcal{A} , and a bit $b \leftarrow_{\S} \{0, 1\}$, there exists some negligible function negl such that $\Pr[b' = b] \leq \frac{1}{2} + \text{negl}(\lambda)$, where b' is the result of running the game $\text{RE}_{\mathcal{A}, \Sigma_R}(1^\lambda, b)$ from Figure 3. We say \mathcal{A} wins the adSHVZK game if $\Pr[b' = b] > \frac{1}{2} + \text{negl}(\lambda)$.*

$\text{RE}_{\mathcal{A}, \Sigma_R}(1^\lambda, 0)$	$\text{RE}_{\mathcal{A}, \Sigma_R}(1^\lambda, 1)$
1 : $(\text{ppm}, z_s) \leftarrow \Sigma_R.\text{SimSetup}(1^\lambda)$	1 : $(\text{ppm}, z_s) \leftarrow \Sigma_R.\text{SimSetup}(1^\lambda)$
2 : $(\text{Prove}, x, w, \text{st}) \leftarrow \mathcal{A}(1^\lambda, \text{ppm})$	2 : $(\text{Prove}, x, w, \text{st}) \leftarrow \mathcal{A}(1^\lambda, \text{ppm})$
3 : if $R(x, w) = 1$:	3 : if $R(x, w) = 1$:
4 : $r \leftarrow_{\S} \{0, 1\}^{\lambda r}$	4 : $r' \leftarrow \{0, 1\}^{\lambda R'}$
5 : $\text{chl} \leftarrow \{0, 1\}^\ell$	5 : $\text{chl} \leftarrow \{0, 1\}^\ell$
6 : $\pi \leftarrow \Sigma_R.\text{Prove}((x, w, r), (x, \text{chl}))$	6 : $(\pi, z_\pi) \leftarrow \Sigma_R.\text{SimProve}(z_s, x, \text{chl}, r')$
7 : $r' \leftarrow \Sigma_R.\text{RealToSim}(\text{ppm}, z_s, x, \pi, w, r)$	7 :
8 : else :	8 : else :
9 : $\pi \leftarrow \perp$	9 : $\pi \leftarrow \perp$
10 : $b' \leftarrow \mathcal{A}(\text{st}, \pi, r')$	10 : $b' \leftarrow \mathcal{A}(\text{st}, \pi, r')$
11 : return b'	11 : return b'

Fig. 3. Randomness Equivocability (RE) Game.

Together, the `SimRand` and `RealToSim` algorithms essentially decouple the nature of the proof as real or simulated from the nature of the random coins as real or simulated—they allow the simulator to decide post-proof whether the prover’s revealed randomness corresponds to $\Sigma.\text{Prove}$ or $\Sigma.\text{SimProve}$. When plugged into an OR-protocol, this allows 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 $\Sigma_{\bar{b}}.\text{SimRand}$) and the simulator’s random tape for the previously “real” proof π_b (according to $\Sigma_b.\text{RealToSim}$). We call this OR-protocol algorithm `EquivRand`.

Definition 14 (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} (Defi-*

inition 9) is a tuple of procedures $\Sigma_{OR} = (\text{Setup}, \text{Prove}, \text{Verify}, \text{SimSetup}, \text{SimProve}, \text{SimRand}, \text{Extract})$, defined as follows.

- $\text{PPM} \leftarrow \text{Setup}(1^\lambda)$: Given a security parameter 1^λ , run $\Sigma_{R_0}.\text{Setup}(1^\lambda)$ to obtain ppm_0 and $\Sigma_{R_1}.\text{Setup}(1^\lambda)$ to obtain ppm_1 . Output $\text{PPM} = (\text{ppm}_0, \text{ppm}_1)$.
- $\Phi \leftarrow \text{Prove}(X, W, T)$: Parse $X = (x_0, x_1)$, $W = (w, b)$, and $T = (r_0, r_1)$, and let b be the bit such that $(x_b, w) \in R_b$. Execute the following:
 - $\text{Com} \leftarrow \text{Commit}(X, W, T)$: P computes com_b according to $\tau_b.\text{Commit}(x_b, w, r_b)$. P chooses $\text{chl}_{\bar{b}}$ at random and generates $(\text{com}_{\bar{b}}, \text{chl}_{\bar{b}}, \text{res}_{\bar{b}})$ by running $\Sigma_{R_{\bar{b}}}.\text{SimProve}(x_{\bar{b}}, z_{\bar{b}}, \text{chl}_{\bar{b}}, r_{\bar{b}})$. P sends $V \text{Com} = (\text{com}_0, \text{com}_1)$.
 - $\text{ChL} \leftarrow \text{Challenge}(X, \text{Com})$: V sends P a random ℓ -bit string ChL .
 - $\text{Res} \leftarrow \text{Respond}(X, W, \text{Com}, \text{ChL})$: P sets $\text{chl}_b = \text{ChL} \oplus \text{chl}_{\bar{b}}$ and computes res_b according to $\tau_b.\text{Respond}(x_b, w, \text{com}_b, \text{chl}_b)$. P sends $(\text{ChL}, \text{Res}) = (\text{chl}_0, \text{chl}_1, \text{res}_0, \text{res}_1)$ to V .

The output “proof” Φ is a tuple $(\pi_0, \pi_1, \text{ChL})$, where $\pi_b = (\text{com}_b, \text{chl}_b, \text{res}_b)$.

- $\{0, 1\} \leftarrow \text{Verify}(X, \Phi)$: Parse Φ as $(\pi_0, \pi_1, \text{ChL})$, where $\pi_b = (\text{com}_b, \text{chl}_b, \text{res}_b)$ for $b \in \{0, 1\}$. Execute the following:
 - $\{0, 1\} \leftarrow \text{Decision}(X, \text{Com}, \text{ChL}, \text{Res})$: If $\tau_0.\text{Decision}(x_0, \text{com}_0, \text{chl}_0, \text{res}_0) = 1$ and $\tau_1.\text{Decision}(x_1, \text{com}_1, \text{chl}_1, \text{res}_1) = 1$, return 1 (accept). Otherwise, return 0 (reject).

If $\text{Decision}(X, \text{Com}, \text{ChL}, \text{Res}) = 1$ and $\text{chl}_0 \oplus \text{chl}_1 = \text{ChL}$, output 1 (accept). Otherwise, output 0 (reject).

- $(\text{PPM}, Z) \leftarrow \text{SimSetup}(1^\lambda)$: Compute $(\text{ppm}_0, z_0) \leftarrow \Sigma_{R_0}.\text{SimSetup}(1^\lambda)$ and $(\text{ppm}_1, z_1) \leftarrow \Sigma_{R_1}.\text{SimSetup}(1^\lambda)$. Return (PPM, Z) where $\text{PPM} = (\text{ppm}_0, \text{ppm}_1)$ and $Z = (z_0, z_1)$.
- $\Phi \leftarrow \text{SimProve}(\text{PPM}, X, Z, \text{ChL}, T')$: Parse $\text{PPM} = (\text{ppm}_0, \text{ppm}_1)$, $X = (x_0, x_1)$, $Z = (z_0, z_1)$, and $T' = (r'_0, r'_1)$. Generate chl_0 uniformly at random and set $\text{chl}_1 = \text{chl}_0 \oplus \text{ChL}$. Compute $\pi_0 \leftarrow \Sigma_{R_0}.\text{SimProve}(x_0, \text{chl}_0)$ and $\pi_1 \leftarrow \Sigma_{R_1}.\text{SimProve}(x_1, \text{chl}_1)$. Return $\Phi = (\pi_0, \pi_1, \text{ChL})$.
- $T \leftarrow \text{SimRand}(\text{PPM}, Z, X, \Phi, W, T')$: Parse $\text{PPM} = (\text{ppm}_0, \text{ppm}_1)$, $Z = (z_0, z_1)$, $X = (x_0, x_1)$, $\Phi = (\pi_0, \pi_1, \text{ChL})$, $W = (w, b)$, and $T' = (r'_0, r'_1)$. Compute $r_b \leftarrow \Sigma_{R_b}.\text{SimRand}(\text{ppm}_b, z_b, x_b, \pi_b, w, r'_b)$ and let $r_{\bar{b}} = r'_{\bar{b}}$. Return $T = (r_0, r_1)$.
- $T \leftarrow \text{EquivRand}(\text{PPM}, Z, X, \Phi, W, T')$: Parse $\text{PPM} = (\text{ppm}_0, \text{ppm}_1)$, $Z = (z_0, z_1)$, $X = (x_0, x_1)$, $\Phi = (\pi_0, \pi_1, \text{ChL})$, $W = (w, b)$, and $T' = (r'_0, r'_1)$. Compute $r_b \leftarrow \Sigma_{R_b}.\text{RealToSim}(\text{ppm}_b, z_b, x_b, \pi_b, w, r'_b)$ and $r_{\bar{b}} \leftarrow \Sigma_{R_{\bar{b}}}.\text{SimRand}(\text{ppm}_b, z_b, x_b, \pi_b, w, r'_b)$. Return $T = (r_0, r_1)$.
- $W \leftarrow \text{Extract}(X, \Phi, \Phi')$: Parse $X = (x_0, x_1)$, $\Phi = (\pi_0, \pi_1)$, and $\Phi' = (\pi'_0, \pi'_1)$. Compute $w_0 \leftarrow \Sigma_{R_0}.\text{Extract}(x_0, \pi_0, \pi'_0)$ and $w_1 \leftarrow \Sigma_{R_1}.\text{Extract}(x_1, \pi_1, \pi'_1)$. Return $W = (w_0, w_1)$.

3.3 Requirements for Adaptive Witness Indistinguishability

Finally, we formalize the definition of adaptive witness indistinguishability for OR-protocols discussed in Section 3.2 and show in Theorem 1 that as long as the underlying Σ -protocols Σ_0 and Σ_1 are both adaptive special honest-verifier zero-knowledge *and* randomness equivocal, the OR-protocol based on Σ_0 and Σ_1 is adaptive witness indistinguishable.

Definition 15 (Adaptive Witness Indistinguishability). *An adaptive OR-protocol $\Sigma_{OR} = (\text{Setup}, \text{Prove}, \text{Verify}, \text{SimSetup}, \text{SimProve}, \text{SimRand}, \text{Extract})$ (Definition 14) based on two adaptive Σ -protocols Σ_{R_0} and Σ_{R_1} (Definition 9) is adaptive witness indistinguishable (adWI) if there exists an algorithm EquivRand such that for all security parameters 1^λ , all PPT \mathcal{A} , and a bit $b \leftarrow_{\S} \{0, 1\}$, there exists some negligible function negl such that $\Pr[b' = b] \leq \frac{1}{2} + \text{negl}(\lambda)$, where b' is the result of running the game $\text{adWI}_{\mathcal{A}, \Sigma_{OR}}(1^\lambda, b)$ from Figure 4. We say \mathcal{A} wins the adWI game if $\Pr[b' = b] > \frac{1}{2} + \text{negl}(\lambda)$.*

$\text{adWI}_{\mathcal{A}, \Sigma_{OR}}(1^\lambda, 0)$	$\text{adWI}_{\mathcal{A}, \Sigma_{OR}}(1^\lambda, 1)$
1 : $\text{PPM} \leftarrow \Sigma_{OR}.\text{Setup}(1^\lambda)$	1 : $\text{PPM} \leftarrow \Sigma_{OR}.\text{Setup}(1^\lambda)$
2 : $(x_0, x_1, w_0, w_1, b, \text{st}) \leftarrow \mathcal{A}(1^\lambda, \text{ppm})$	2 : $(x_0, x_1, w_0, w_1, b, \text{st}) \leftarrow \mathcal{A}(1^\lambda, \text{ppm})$
3 : if $R_0(x_0, w_0) = R_1(x_1, w_1) = 1$:	3 : if $R_0(x_0, w_0) = R_1(x_1, w_1) = 1$:
4 : Set $W = (w_b, b)$	4 : Set $W = (w_{\bar{b}}, \bar{b})$
5 : $T \leftarrow_{\S} \{0, 1\}^{\lambda T}$	5 : $T' \leftarrow_{\S} \{0, 1\}^{\lambda T'}$
6 : $\text{CHL} \leftarrow \{0, 1\}^\ell$	6 : $\text{CHL} \leftarrow \{0, 1\}^\ell$
7 : $\Phi \leftarrow \Sigma_{OR}.\text{Prove}((X, W, T), (X, \text{CHL}))$	7 : $\Phi \leftarrow \Sigma_{OR}.\text{Prove}((X, W, T), (X, \text{CHL}))$
8 :	8 : $T \leftarrow \Sigma_{OR}.\text{EquivRand}(Z, X, \Phi, W, T')$
9 : else :	9 : else :
10 : $\Phi \leftarrow \perp$	10 : $\Phi \leftarrow \perp$
11 : $b' \leftarrow \mathcal{A}(\Phi, T, \text{st})$	11 : $b' \leftarrow \mathcal{A}(\Phi, T, \text{st})$
12 : return b'	12 : return b'

Fig. 4. Adaptive Witness Indistinguishability (adWI) Game.

Theorem 1 (adSHVZK and RE ImPLY Adaptive WI). *Let $\Sigma_{OR} = (\text{Setup}, \text{Prove}, \text{Verify}, \text{SimSetup}, \text{SimProve}, \text{SimRand}, \text{Extract})$ be an OR-protocol (Definition 14) based on two Σ -protocols Σ_{R_0, τ_0} and Σ_{R_1, τ_1} (Definition 9). If Σ_{OR} is adaptive special honest-verifier zero-knowledge (Definition 11) and randomness equivocal (Definition 13), then Σ_{OR} is adaptive witness indistinguishable (Definition 15).*

Proof. We wish to show that as long as Σ_{OR} is adaptive special honest-verifier zero-knowledge (adSHVZK) and randomness equivocal (RE), \mathcal{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 $\Pr[b' = b] \leq \frac{1}{2} + \text{negl}(\lambda)$ is replaced with $|\Pr[b' = 0 \mid b = 0] - \Pr[b' = 0 \mid b = 1]| \leq \text{negl}(\lambda)$, where b' represents \mathcal{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 $\Sigma_{OR}.\text{Prove}$ —such that π_b and r_b are real (according to $\Sigma_b.\text{Prove}$ and the prover’s random tape) and $\pi_{\bar{b}}$ and $r_{\bar{b}}$ are simulated (according to $\Sigma_{\bar{b}}.\text{SimProve}$ 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 $\Sigma_b.\text{SimRand}$ and $\Sigma_{\bar{b}}.\text{RealToSim}$). We fix \mathcal{A} 's probability of outputting 0 in the RS (resp. SR) experiment to be $p_{RS}^{\mathcal{A}}(\lambda)$ (resp. $p_{SR}^{\mathcal{A}}(\lambda)$). 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 \mathcal{A} , as long as Σ_{OR} is adSHVZK and RE, $|p_{RS}^{\mathcal{A}}(\lambda) - p_{SR}^{\mathcal{A}}(\lambda)| \leq \text{negl}(\lambda)$.

Lemma 1. *Let the 0-bit experiment of the adWI game (Figure 4) 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 $r_b \leftarrow_{\S} \{0, 1\}^{\lambda R_b}$, $r'_b \leftarrow_{\S} \{0, 1\}^{\lambda R'_b}$, $\text{chl}_0 \leftarrow_{\S} \{0, 1\}^{\ell_0}$, and $\text{chl}_1 \leftarrow_{\S} \{0, 1\}^{\ell_1}$. Set $\text{CHL} = \text{chl}_0 \oplus \text{chl}_1$. Compute $\pi_0 \leftarrow \Sigma_0.\text{SimProve}(z_0, x_0, \text{chl}_0, r_0)$ and $\pi_1 \leftarrow \Sigma_1.\text{SimProve}(z_1, x_1, \text{chl}_1, r_1)$. Set $\Phi = (\pi_0, \pi_1, \text{CHL})$. Finally, compute $r_{\bar{b}} \leftarrow \Sigma_{\bar{b}}.\text{SimRand}(\text{ppm}_{\bar{b}}, z_{\bar{b}}, x_{\bar{b}}, \pi_{\bar{b}}, w_{\bar{b}}, r'_{\bar{b}})$ and set $T = r_0 \| r_1$. Then for all \mathcal{A} , as long as both Σ_0 and Σ_1 are adaptive special honest-verifier zero-knowledge (adSHVZK), $|p_{RS}^{\mathcal{A}}(\lambda) - p_{SS}^{\mathcal{A}}(\lambda)| \leq \text{negl}(\lambda)$.*

Proof. Assume for a contradiction that \mathcal{A} 's fixed 0-output probabilities $p_{RS}^{\mathcal{A}}(\lambda)$ and $p_{SS}^{\mathcal{A}}(\lambda)$ are such that $|p_{RS}^{\mathcal{A}}(\lambda) - p_{SS}^{\mathcal{A}}(\lambda)| > \text{negl}(\lambda)$. We construct a reduction \mathcal{B} that, given \mathcal{A} as a black-box, outputs 0 with probability $p_0^{\mathcal{B}}$ when the challenge bit $b^* = 0$ and probability $p_1^{\mathcal{B}}$ when the challenge bit $b^* = 1$ in the adSHVZK game such that $|p_0^{\mathcal{B}} - p_1^{\mathcal{B}}| > \text{negl}(\lambda)$. \mathcal{B} proceeds as follows.

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

If \mathcal{C}_b is running on input a challenge bit $b^* = 0$, then the returned proof π will be according to $\Sigma_b.\text{Prove}$ on input the random tape r while the second proof $\pi_{\bar{b}}$ is according to $\Sigma_{\bar{b}}.\text{SimProve}$ and returned with the simulator’s random coins, exactly as \mathcal{A} expects from the adWI-RS experiment. Therefore, when

$b^* = 0$, $p_0^{\mathcal{B}} = p_{\text{RS}}^{\mathcal{A}}$. If \mathcal{C}_b is running on input $b^* = 1$, then π will be according to $\Sigma_b.\text{SimProve}$ and the random tape r will be according to $\Sigma_b.\text{SimRand}$. The second proof $\pi_{\bar{b}}$ will also be simulated and returned with the simulator's randomness, exactly as \mathcal{A} expects from the adWI-SS experiment. Therefore, when $b^* = 1$, $p_1^{\mathcal{B}} = p_{\text{SS}}^{\mathcal{A}}$. Since we have $|p_{\text{RS}}^{\mathcal{A}}(\lambda) - p_{\text{SR}}^{\mathcal{A}}(\lambda)| > \text{negl}(\lambda)$ by assumption, this implies $|p_0^{\mathcal{B}} - p_1^{\mathcal{B}}| > \text{negl}(\lambda)$, contradicting the adSHVZK property of Σ_b and completing the proof of Lemma 1. \square

Lemma 2. *Let the 1-bit experiment of the adWI game (Figure 4) 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 $r'_b \leftarrow_{\mathcal{S}} \{0, 1\}^{\lambda R'_b}$ and $r''_b \leftarrow_{\mathcal{S}} \{0, 1\}^{\lambda R''_b}$, then compute $r_b \leftarrow \Sigma_b.\text{SimRand}(\text{ppm}_b, z_b, x_b, \pi_b, w_b, r'_b)$. For all \mathcal{A} , as long as both Σ_0 and Σ_1 are randomness equivocable (RE), $|p_{\text{SR}}^{\mathcal{A}}(\lambda) - p_{\text{SS}}^{\mathcal{A}}(\lambda)| \leq \text{negl}(\lambda)$.*

Proof. Again assume for a contradiction that \mathcal{A} 's fixed 0-output probabilities $p_{\text{SR}}^{\mathcal{A}}(\lambda)$ and $p_{\text{SS}}^{\mathcal{A}}(\lambda)$ are such that $|p_{\text{SR}}^{\mathcal{A}}(\lambda) - p_{\text{SS}}^{\mathcal{A}}(\lambda)| > \text{negl}(\lambda)$. We construct a reduction \mathcal{B} that, given \mathcal{A} as a black-box, outputs 0 with probability $p_0^{\mathcal{B}}$ when the challenge bit $b^* = 0$ and probability $p_1^{\mathcal{B}}$ when the challenge bit $b^* = 1$ in the RE game such that $|p_0^{\mathcal{B}} - p_1^{\mathcal{B}}| > \text{negl}(\lambda)$.

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

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

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

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

$$|p_{\text{RS}}^{\mathcal{A}}(\lambda) - p_{\text{SR}}^{\mathcal{A}}(\lambda)| \leq \text{negl}(\lambda) + \text{negl}(\lambda) \leq \text{negl}(\lambda),$$

completing the proof of Theorem 1. \square

4 Adaptive Straight-Line Compilers

Adaptive straight-line compilers are straight-line compilers [29] that preserve the adaptive security properties of the Σ -protocol being transformed. Recall from the introduction that a regular straight-line compiler SLC takes a Σ -protocol Σ_R for a relation R as input and produces a new proof system Π_R^{SLC} that is a tuple of (non-interactive) *algorithms* ($\text{Setup}^H, \text{Prove}^H, \text{Verify}^H, \text{SimSetup}, \text{SimProve}, \text{Extract}$), where H is a traditional RO.

In order to be considered a straight-line compiler, Π_R^{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). Our *adaptive* version of an SLC, denoted aSLC, says that Π_R^{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 true randomness with the output of SimRand .

Definition 16 (Adaptive Straight-Line Compiler). *An algorithm 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 9) as input, it outputs a tuple of algorithms $\Pi_R^{\text{SLC}} = (\text{Setup}^H, \text{Prove}^H, \text{Verify}^H, \text{SimSetup}, \text{SimProve}, \text{SimRand}, \text{Extract})$ based on Σ_R that satisfy the following properties: overwhelming completeness (Definition 17), adaptive non-interactive multiple special honest-verifier zero-knowledge (Definition 18), and adaptive non-interactive special simulation soundness (Definition 19).*

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

Definition 17 (Overwhelming Completeness). *An adaptive NISLE proof system Π_R^{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 $\pi \leftarrow \Pi_R^{\text{aSLC}}.\text{Prove}^H(x, w)$,*

$$\Pr[\Pi_R^{\text{aSLC}}.\text{Verify}^H(x, \pi) = 1] \geq 1 - \text{negl}(\lambda).$$

The RO in the “real-world” experiment H_f is parameterized by a function $f \leftarrow_{\S} F$ selected from random function family F , while the RO in the “ideal-world” experiment is a list oracle H_L parameterized by an initially empty list L that the challenger in the security experiment can program via the interface Prog_L . 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

³ To satisfy NIM-SHVZK when $\Sigma_R.\text{SimProve}$ involves programming, the first message com will need entropy that is superlogarithmic in the security parameter [24,29].

RO $H_f(x)$	Random List Oracle $H_L(x)$	Interface $\mathbf{Prog}_L(x, v)$
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 : $L.\mathbf{append}(x, v)$
	3 : else :	
	4 : $v \leftarrow \{0, 1\}^\ell$	
	5 : $L.\mathbf{append}(x, v)$	
	6 : return v	

Fig. 5. Random Oracle Functionalities for NIM-SHVZK and NI-SSS Games [29].

For convenience and consistency the \mathbf{Prog} interface is included in the definitions of NIM-SHVZK and NI-SSS; in non-programmable models such as the $\mathcal{G}_{\text{roRO}}$ -hybrid model, we consider \mathbf{Prog} a defunct oracle that returns \perp .

Definition 18 (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 $\Pi_R^{\text{aSLC}}.\mathbf{SimSetup}$, $\Pi_R^{\text{aSLC}}.\mathbf{SimProve}$, and $\Pi_R^{\text{aSLC}}.\mathbf{SimRand}$ such that for any security parameter λ , any PPT adversary \mathcal{A} , and a bit $b \leftarrow_{\mathcal{S}} \{0, 1\}$, there exists some negligible function negl such that $\Pr[b' = b] \leq \frac{1}{2} + \text{negl}(\lambda)$, where b' is the result of running the game $\text{adNIM-SHVZK}_{\mathcal{A}, \Pi_R^{\text{aSLC}}}^{H_*, *}(1^\lambda, b)$ from Figure 6.*

Similarly, we extend Lysyanskaya and Rosenbloom’s definition of NI-SSS [29] by giving \mathcal{A} not only the output of $\Pi_R^{\text{aSLC}}.\mathbf{SimProve}$, but of $\Pi_R^{\text{aSLC}}.\mathbf{SimRand}$ as well. The adaptive NI-SSS property says that soundness must hold even when \mathcal{A} can see polynomially-many proofs from the simulator *and* can corrupt polynomially-many provers (i.e., see the contents of polynomially-many random tapes). Our work has the same limitation of Lysyanskaya and Rosenbloom [29] in that we formalize the $\Pi_R^{\text{aSLC}}.\mathbf{Extract}$ algorithm to work based on the adversary’s queries to the random oracle, denoted $\mathcal{Q}_{\mathcal{A}}$. Extending the formalization of SLCs to include “key-based” extractors that leverage verifiable encryption schemes [5]—and determining whether or not such extractors can be used to obtain (adaptive) UC NIZKPoK—is left for future work.

Definition 19 (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 $\Pi_R^{\text{SLC}}.\mathbf{Extract}$ such that for any security parameter λ and any PPT adversary \mathcal{A} ,*

$$\Pr[\text{Fail} \leftarrow \text{adNI-SSS}_{\mathcal{A}, \Pi_R^{\text{SLC}}}^H(1^\lambda)] \leq \text{negl}(\lambda),$$

where H is any random oracle and adNI-SSS is the game described in Figure 7.

$\text{adNIM-SHVZK}_{\mathcal{A}, \Pi_R^{\text{SLC}}}^{H_*, F}(1^\lambda, 0): \text{REAL}$	$\text{adNIM-SHVZK}_{\mathcal{A}, \Pi_R^{\text{SLC}}}^{H_*, \text{Prog}}(1^\lambda, 1): \text{IDEAL}$
1 : $f \leftarrow_{\S} F$	1 : $L \leftarrow \perp$
2 : $\text{ppm} \leftarrow \Pi_R^{\text{SLC}}.\text{Setup}^{H_f}(1^\lambda)$	2 : $(\text{ppm}, z_s) \leftarrow \Pi_R^{\text{SLC}}.\text{SimSetup}^{\text{Prog}_L}(1^\lambda)$
3 : $\text{st} \leftarrow \mathcal{A}^{H_f}(1^\lambda, \text{ppm})$	3 : $\text{st} \leftarrow \mathcal{A}^{H_L}(1^\lambda, \text{ppm})$
4 : while $\text{st} \notin \{0, 1\}$:	4 : while $\text{st} \notin \{0, 1\}$:
5 : $(\text{Prove}, x, w, \text{st}) \leftarrow \mathcal{A}^{H_f}(\text{st})$	5 : $(\text{Prove}, x, w, \text{st}) \leftarrow \mathcal{A}^{H_L}(\text{st})$
6 : if $R(x, w) = 1$:	6 : if $R(x, w) = 1$:
7 : $r \leftarrow_{\S} \{0, 1\}^{\lambda r}$	7 : $(\pi, z_\pi) \leftarrow \Pi_R^{\text{SLC}}.\text{SimProve}^{\text{Prog}_L}(z_s, x)$
8 : $\pi \leftarrow \Pi_R^{\text{SLC}}.\text{Prove}^{H_f}(x, w, r)$	8 : $r \leftarrow \Pi_R^{\text{SLC}}.\text{SimRand}^{\text{Prog}_L}(z_s, z_\pi, x, \pi, w)$
9 : else :	9 : else :
10 : $\pi, r \leftarrow \perp$	10 : $\pi, r \leftarrow \perp$
11 : $\text{st} \leftarrow \mathcal{A}^{H_f}(\text{st}, \pi, r)$	11 : $\text{st} \leftarrow \mathcal{A}^{H_L}(\text{st}, \pi, r)$
12 : return st	12 : return st

Fig. 6. Adaptive Non-Interactive Multiple SHVZK (adNIM-SHVZK) Game.

$\text{adNI-SSS}_{\mathcal{A}, \Pi_R^{\text{SLC}}}^{H_L, \text{Prog}_L}(1^\lambda)$
1 : $L, \text{pflist}, \text{Response}, \text{st} \leftarrow \perp;$
2 : $\text{ppm}, z \leftarrow \Pi_R^{\text{SLC}}.\text{SimSetup}^{\text{Prog}_L}(1^\lambda)$
3 : while $\mathcal{A}^{H_L}(1^\lambda, \text{ppm}, \text{st}) \neq \text{halt}$:
4 : $(\text{Query}, \mathcal{Q}_{\mathcal{A}}, \text{st}) \leftarrow \mathcal{A}^{H_L}(\text{st})$
5 : if $\text{Query} = (\text{Prove}, x, w) \wedge R(x, w) = 1$:
6 : $\pi, z_\pi \leftarrow \Pi_R^{\text{SLC}}.\text{SimProve}^{\text{Prog}_L}(z, x)$
7 : $\text{pflist.append}(x, \pi)$
8 : $r \leftarrow \Pi_R^{\text{SLC}}.\text{SimRand}^{\text{Prog}_L}(z, z_\pi, x, \pi, w)$
9 : $\text{Response} \leftarrow (x, \pi, r)$
10 : elseif $\text{Query} = (\text{Challenge}, x, \pi)$
11 : if $\Pi_R^{\text{SLC}}.\text{Verify}^{H_L}(x, \pi) = 1 \wedge (x, \pi) \notin \text{pflist}$:
12 : $w \leftarrow \Pi_R^{\text{SLC}}.\text{Extract}(x, \pi, \mathcal{Q}_{\mathcal{A}})$
13 : if $R(x, w) = 0$: return Fail
14 : $\text{st} \leftarrow \mathcal{A}^{H_L}(\text{st}, \text{Response})$
15 : return Success

Fig. 7. Adaptive Non-Interactive Special Simulation Soundness (adNI-SSS) Game.

Finally, non-interactive special soundness (NI-SS) is a weakened version of the NI-SSS game where \mathcal{A} does not get to interact with the simulator. This is not a sufficient property to obtain adaptive UC NIZKPoK, but we will need it later as a building block for the proof of Theorems 2 and 5.

Definition 20 (Non-Interactive Special Soundness). [29] *A NISLE proof system $\Pi_R^{\text{SLC}} = (\text{Setup}^H, \text{Prove}^H, \text{Verify}^H, \text{SimSetup}, \text{SimProve}, \text{Extract})$ non-interactive special sound (NI-SS) in the random-oracle model if there exists an algorithm $\Pi_R^{\text{SLC}}.\text{Extract}$ such that for any security parameter λ any random oracle H , and any PPT adversary \mathcal{A} ,*

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

where NI-SS is the NI-SS game described in Figure 8. We say \mathcal{A} wins the NI-SS game if $\Pr[\text{Fail} \leftarrow \text{NI-SS}_{\mathcal{A}, \Pi_R^{\text{SLC}}}(1^\lambda)] > \text{negl}(\lambda)$.

$\text{NI-SS}_{\mathcal{A}, \Pi_R^{\text{SLC}}}(1^\lambda)$
1 : $\text{ppm} \leftarrow \Pi_R^{\text{SLC}}.\text{Setup}(1^\lambda)$
2 : $\text{st} \leftarrow \mathcal{A}^H(1^\lambda, \text{ppm})$
3 : while $\text{st} \neq \perp$:
4 : $(x, \pi, \mathcal{Q}^{\mathcal{A}}, \text{st}) \leftarrow \mathcal{A}^H(\text{st})$
5 : Response $\leftarrow \perp$
6 : if $\Pi_R^{\text{SLC}}.\text{Verify}^H(x, \pi) = 1$:
7 : $w \leftarrow \Pi_R^{\text{SLC}}.\text{Extract}(x, \pi, \mathcal{Q}^{\mathcal{A}})$
8 : if $R(x, w) = 0$:
9 : return Fail
10 : $\text{st} \leftarrow \mathcal{A}^H(\text{st}, \text{Response})$
11 : return Success

Fig. 8. Non-Interactive Special Soundness (NI-SS) Game.

5 Adaptive and Universally Composable 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 any adaptive straight-line compiler (SLC) are *necessary* to achieve adaptive UC NIZKPoK in any global ROM (Section 5.1), and that they are *sufficient* to transform any Σ -protocol into an adaptive UC NIZKPoK in the $\mathcal{G}_{\text{rpoRo}}$ -hybrid model (Section 5.2).

For our construction in the $\mathcal{G}_{\text{roRO}}\text{-}\mathcal{F}_{\text{CRS}}$ -hybrid model, we show how to adapt Lysyanskaya and Rosenbloom’s OR-protocol compiler [29] to obtain adaptive UC NIZKPoK without programming the global RO (Section 5.3).

5.1 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 2.1), we use the generic notation \mathcal{G}_{RO} to represent the global RO.

Theorem 2. *Let Π be a protocol that aUC-realizes $\mathcal{F}_{\text{aNIZK}}$ in the \mathcal{G}_{RO} -hybrid model (Definition 7 where \mathcal{G} is replaced with \mathcal{G}_{RO}) with adaptive corruptions. Then Π must be overwhelmingly complete (Definition 17), adaptive NIM-SHVZK (Definition 18), and adaptive NI-SSS (Definition 19).*

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

Our first reduction \mathcal{B} uses an adversary \mathcal{A} that wins the adNIM-SHVZK game from Figure 6 with non-negligible advantage to distinguish between the real- and ideal-world aUC experiments with non-negligible advantage. \mathcal{B} proceeds as follows. After passing the security and public parameters that it received from the aUC experiment to \mathcal{A} , \mathcal{B} forwards \mathcal{A} ’s oracle queries to \mathcal{G}_{RO} and \mathcal{G}_{RO} ’s responses back to \mathcal{A} .

`Prove` queries proceed as follows. Note that according to the definition of adNIM-SHVZK, any time \mathcal{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 \mathcal{A} ’s view, \mathcal{B} first issues the `Prove` query as-is to a new honest party in the aUC experiment.⁴ Before returning the proof to \mathcal{A} , \mathcal{B} then *corrupts* the prover to obtain the prover’s internal tapes. Since the `Prove` operation was the first performed by the honest party, \mathcal{B} simply takes the first λ_r bits of the prover’s random tape and returns these bits to \mathcal{A} along with the proof.

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

⁴ 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 [14].

The only other condition that might allow \mathcal{B} to distinguish the two experiments are if the ideal functionality $\mathcal{F}_{\text{aNIZK}}$ 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: \mathcal{A} that wins the adaptive NI-SSS game from Figure 7, and \mathcal{A}' that wins the regular NI-SS game (Definition 20), in which the adversary does not have access to simulated proofs, with non-negligible advantage. \mathcal{B} answers \mathcal{A} 's queries the same as in the previous reduction, by forwarding all of \mathcal{A} 's oracle queries to \mathcal{G}_{RO} and **Prove** queries to the aUC challenger, then making adaptive corruptions to obtain the prover's randomness. \mathcal{B} forwards \mathcal{A}' 's queries to \mathcal{G}_{RO} (note \mathcal{A}' does not make **Prove** queries, but can run II.Prove itself).

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

5.2 Adaptive UC NIZKPoK in the $\mathcal{G}_{\text{rpoRO}}$ -hybrid Model

We prove in this section 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) $\mathcal{G}_{\text{rpoRO}}$ -hybrid model. First, we make explicit an additional check that an honest verifier (and the **Verify** interface of the ideal functionality $\mathcal{F}_{\text{aNIZK}}$) must make in order to correctly evaluate a proof. Recall that $\mathcal{G}_{\text{rpoRO}}$ 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 $\mathcal{F}_{\text{aNIZK}}$) must check whether $\mathcal{G}_{\text{rpoRO}}$ was programmed at any index queried by the verification procedure, and if so, reject the proof.

Definition 21 (*$\mathcal{G}_{\text{rpoRO}}$ -hybrid Model Verification Check*). *Let Π_R^{SLC} be any NISLE proof system (Definition 16), $\mathcal{F}_{\text{aNIZK}}$ be the adaptive non-interactive zero-knowledge proof of knowledge ideal functionality (Definition 6), and $\mathcal{G}_{\text{rpoRO}}$ be the restricted programmable observable global random oracle (Definition 1). Every time the **Verify** algorithm of Π_R^{SLC} or the **Verify** interface of $\mathcal{F}_{\text{aNIZK}}$ queries $\mathcal{G}_{\text{rpoRO}}$ on some input in , insert the following check: send a query $\text{IsProgrammed}(\text{in})$ to $\mathcal{G}_{\text{rpoRO}}$; if $\mathcal{G}_{\text{rpoRO}}$ returns 1, output 0 (reject).*

We now show that any adaptive straight-line compiler is sufficient to create adaptive UC NIZKPoK in the $\mathcal{G}_{\text{rpoRO}}$ -hybrid model.

Theorem 3. *Let Σ_R be any adaptive Σ -protocol for relation R (Definition 9), $\mathcal{G}_{\text{rpoRO}}$ be the restricted programmable observable global random oracle (Definition 1 in A.2) and SLC be any adaptive straight-line compiler (Definition 16). Then the NISLE proof system $\Pi_R^{\text{SLC}} \leftarrow \text{SLC}(\Sigma_R)$ aUC-realizes $\mathcal{F}_{\text{aNIZK}}$ in the $\mathcal{G}_{\text{rpoRO}}$ -hybrid model (Definition 7 where \mathcal{G} is replaced with $\mathcal{G}_{\text{rpoRO}}$).*

Proof. We begin with the construction of the ideal adversary (simulator) \mathcal{S} .

Construction of the Ideal Adversary. When \mathcal{S} receives (Setup, s) from $\mathcal{F}_{\text{aNIZK}}$, \mathcal{S} returns the tuple of algorithms Π_R^{SLC} . When corrupted parties issue IsProgrammed queries, \mathcal{S} returns **false**. When \mathcal{Z} issues a query $(\text{Corrupt}, P)$ for a party $P = (\text{pid}, \text{sid})$, \mathcal{S} sends $(\text{Corrupt}, P)$ to $\mathcal{F}_{\text{aNIZK}}$ to obtain the stored tuples $(\text{Proof}, P, x_1, w_1, \pi_1, z_1), \dots, (\text{Proof}, P, x_q, w_q, \pi_q, z_q)$ corresponding to each query $(\text{Prove}, P, x_i, w_i)$ that \mathcal{Z} issued to P . (who forwarded them to $\mathcal{F}_{\text{aNIZK}}$). For each tuple, \mathcal{S} interprets $z_i = (z_s, z_{\pi_i})$ as the auxiliary outputs of $\Pi_R^{\text{SLC}}.\text{SimSetup}$ and $\Pi_R^{\text{SLC}}.\text{SimProve}$, respectively. \mathcal{S} then runs $r_i \leftarrow \Pi_R^{\text{SLC}}.\text{SimRand}(z_s, z_{\pi_i}, x, \pi, w)$ to obtain simulated randomness r_i . To reconstruct the prover’s random tape R , \mathcal{S} concatenates $R = r_1 || \dots || r_q$ 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 Π_R^{SLC} with the functionality of $\mathcal{F}_{\text{aNIZK}}$ 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 **Prove** functionality with the **Prove** interface of $\mathcal{F}_{\text{aNIZK}}$ 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 $\mathcal{G}_{\text{rpoRO}}$ 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 $\Pi_R^{\text{SLC}}.\text{SimProve}$ produces valid proofs for statements $x \in L_R$ with overwhelming probability (which follows from overwhelming completeness), the environment’s view of $\mathcal{G}_{\text{rpoRO}}$ 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 $(\text{Prove}, P, x_i, w_i)$ to \mathcal{C} , \mathcal{C} forwards the query to the adNIM-SHVZK challenger from Figure 6, 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 (**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 (**Verify**, x, π), \mathcal{C} returns the result of running $\Pi_R^{\text{SLC}}.\text{Verify}(x, \pi)$. \mathcal{C} outputs whatever \mathcal{Z} outputs.

Note that if the adNIM-SHVZK challenger is running the 0-bit experiment (using $\Pi_R^{\text{SLC}}.\text{Prove}$ 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 $\Pi_R^{\text{SLC}}.\text{SimProve}$ and $\Pi_R^{\text{SLC}}.\text{SimRand}$), \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 $\mathcal{F}_{\text{aNIZK}}$, and show the environment's views are indistinguishable between Hybrids 2 and 3 as long as Π_R^{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 7 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 $\Pi_R^{\text{SLC}}.\text{Extract}$ 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 (**Verify**, x, π) for a proof π that \mathcal{C} did not send to \mathcal{Z} , \mathcal{C} sends (**Challenge**, x, π, Q_{P^*}) to the adNI-SSS challenger. Since both the adNI-SSS challenger and $\mathcal{F}_{\text{aNIZK}}$ use the $\Pi_R^{\text{SLC}}.\text{Extract}$ 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 $\mathcal{F}_{\text{aNIZK}}$ and \mathcal{S} . Note that since \mathcal{C} now runs all of $\mathcal{F}_{\text{aNIZK}}$ and \mathcal{S} 's procedures, this is again a syntactic difference, and Hybrid 3 is identical to Hybrid 4. \square

5.3 Adaptive UC NIZKPoK in the $\mathcal{G}_{\text{roRO}}\text{-}\mathcal{F}_{\text{CRS}}$ -hybrid Model

Similarly, any adaptive SLC in conjunction with our adaptive version of the OR-protocol construction given by Lysyanskaya and Rosenbloom [29] is sufficient to create adaptive UC NIZKPoK in the $\mathcal{G}_{\text{roRO}}\text{-}\mathcal{F}_{\text{CRS}}$ -hybrid model. We begin by amending the required properties for the CRS relation (which we reviewed in Section 2.2) to include adaptivity, and by formalizing an adaptive OR-protocol. We then update the algorithms of Lysyanskaya and Rosenbloom's OR-protocol construction to create an *adaptive* NISLE proof system, denoted $\Pi_{\text{RVS}}^{\text{auc}}$.

Definition 22 (Adaptive Σ -Friendly Relation). *An adaptive Σ -friendly relation S is a binary \mathcal{NP} relation with a corresponding efficient and adaptive Σ -protocol Σ_S .*

Finally, we specify a candidate aUC Σ -protocol compiler for the $\mathcal{G}_{\text{roRD}}\text{-}\mathcal{F}_{\text{CRS}}$ -hybrid model, and show that, when composed with an adaptive Σ -friendly relation S , any Σ -protocol Σ_R can be made into an aUC NIZKPoK in the same model.

Definition 23 (Candidate Compiler). *Let Σ_R be any adaptive Σ -protocol for relation R (Definition 9), $\mathcal{G}_{\text{roRD}}$ be the restricted observable global random oracle (Definition 1), Σ_S be an efficient Σ -protocol for samplable-hard relation S (Definition 3), \mathcal{F}_{CRS} be the ideal CRS functionality (Definition 5) where $\text{GenCRS} := \kappa_S$, and aSLC be any adaptive straight-line compiler (Definition 16). Then our candidate compiler auc is an algorithm that, on input Σ_R and aSLC , produces a tuple of algorithms $\Pi_{\text{RVS}}^{\text{auc}} = (\text{Setup}^{\mathcal{G}_{\text{roRD}}}, \text{Prove}^{\mathcal{G}_{\text{roRD}}, \mathcal{F}_{\text{CRS}}}, \text{Verify}^{\mathcal{G}_{\text{roRD}}, \mathcal{F}_{\text{CRS}}}, \text{SimSetup}, \text{SimProve}, \text{SimRand}, \text{Extract})$, defined in Figure 9.*

Theorem 4. *Let Σ_R be any adaptive Σ -protocol for relation R (Definition 9), $\mathcal{G}_{\text{roRD}}$ be the restricted observable global random oracle (Definition 1), Σ_S be an efficient Σ -protocol for samplable-hard relation S (Definition 3), \mathcal{F}_{CRS} be the ideal CRS functionality (Definition 5), aSLC be any adaptive straight-line compiler (Definition 16), and auc be the adaptive OR-protocol compiler (Definition 23). Then $\Pi_{\text{RVS}}^{\text{aSLC}} \leftarrow \text{auc}(\Sigma_R, \text{aSLC})$ aUC-realizes $\mathcal{F}_{\text{aNIZK}}$ in the $\mathcal{G}_{\text{roRD}}\text{-}\mathcal{F}_{\text{CRS}}$ -hybrid model (Definition 7 where \mathcal{G} is replaced with $\mathcal{G}_{\text{roRD}}$ and $*$ is replaced with \mathcal{F}_{CRS}).*

Proof. The construction of the ideal adversary (simulator) \mathcal{S} is the same as in the proof of Theorem 3, except it returns $\Pi_{\text{RVS}}^{\text{auc}}$ to $\mathcal{F}_{\text{aNIZK}}$ rather than Π_R^{SLC} , and there are no `IsProgrammed` queries. (Note that the simulation and proof trapdoors, z_s and z_π , are simply the simulator's CRS list `simcrs` and section of the random tape T' corresponding to π , respectively.)

We again create a hybrid reduction that starts in the real-world experiment and replaces each piece of the real-world adaptive NISLE OR-protocol $\Pi_{\text{RVS}}^{\text{auc}}$ with the functionality of $\mathcal{F}_{\text{aNIZK}}$ and \mathcal{S} .

Hybrid 1. Identical to Hybrid 1 in the proof of Theorem 3.

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 $\Pi_{\text{RVS}}^{\text{aSLC}}.\text{SimSetup}$, $\Pi_{\text{RVS}}^{\text{aSLC}}.\text{SimProve}$, and $\Pi_{\text{RVS}}^{\text{aSLC}}.\text{SimRand}$. This step allows us to avoid giving \mathcal{C} control over the CRS trapdoors for now, such that we are able to show in the next hybrid argument that \mathcal{C} can use a proof-forging environment to break either the adNI-SSS property or the hardness property of the samplable-hard relation S (i.e. the reduction produces a CRS trapdoor). The proof that Hybrid 2 is indistinguishable from Hybrid 1 proceeds identically to the proof under Hybrid 2 in Theorem 3 above, modulo the `IsProgrammed` interface.

Hybrid 3. In the third hybrid, we replace \mathcal{C} 's `Verify` functionality with the `Verify` functionality of $\mathcal{F}_{\text{aNIZK}}$, and show the environment's views are indistinguishable between Hybrids 2 and 3 as long as Π_R^{SLC} is adaptive non-interactive special simulation-sound (adaptive NI-SSS) and S is a hard relation (i.e. given

auc Compiler Parameters	
$1^\lambda, R, \Sigma_R, \lambda_R, \lambda_{R'}, S, \Sigma_S, \lambda_S, \lambda_{S'}, \text{aSLC}, \mathcal{G}_{\text{roR0}}, \mathcal{F}_{\text{CRS}}$ with $\text{GenCRS} := (x, w) \leftarrow \kappa_S(1^\lambda)$	
$\Pi_{\text{RVS}}^{\text{auc}}.\text{Setup}^{\mathcal{G}_{\text{roR0}}}(1^\lambda)$	$\Pi_{\text{RVS}}^{\text{auc}}.\text{SimSetup}(1^\lambda)$
<pre> 1: PPM $\leftarrow \Pi_{\text{RVS}}^{\text{aSLC}}.\text{Setup}^{\mathcal{G}_{\text{roR0}}}(1^\lambda)$ 2: return PPM </pre>	<pre> 1: PPM, Z' $\leftarrow \Pi_{\text{RVS}}^{\text{aSLC}}.\text{SimSetup}(1^\lambda)$ 2: simcrs $\leftarrow \perp$ 3: Z = (Z', simcrs) 4: return (PPM, Z) </pre>
$\Pi_{\text{RVS}}^{\text{auc}}.\text{Prove}^{\mathcal{G}_{\text{roR0}}, \mathcal{F}_{\text{CRS}}}(\text{PPM}, s, x, w, T, T')$	$\Pi_{\text{RVS}}^{\text{auc}}.\text{SimProve}(\text{PPM}, Z, s, x, w, T, T')$
<pre> 1: if R(x, w) $\neq 1$: 2: return \perp 3: CRS_s $\leftarrow \mathcal{F}_{\text{CRS}}^s.\text{Query}(s)$ 4: X $\leftarrow (x, \text{CRS}_s)$ 5: W $\leftarrow (w, 0)$ 6: $\Phi \leftarrow \Pi_{\text{RVS}}^{\text{aSLC}}.\text{Prove}^{\mathcal{G}_{\text{roR0}}}(\text{PPM}, X, W, T, T')$ 7: return (s, X, Φ) </pre>	<pre> 1: if R(x, w) $\neq 1$: 2: return \perp 3: if $\nexists (\text{CRS}_s, \text{trap}_s)$ s.t. 4: (s, CRS_s, trap_s) \in simcrs : 5: (CRS_s, trap_s) $\leftarrow \kappa_S(1^\lambda)$ 6: simcrs.append(s, CRS_s, trap_s) 7: X $\leftarrow (x, \text{CRS}_s)$ 8: W $\leftarrow (\text{trap}_s, 1)$ 9: $\Phi \leftarrow \Pi_{\text{RVS}}^{\text{aSLC}}.\text{Prove}^{\mathcal{G}_{\text{roR0}}}(\text{PPM}, X, W, T, T')$ 10: return (s, X, Φ, simcrs) </pre>
$\Pi_{\text{RVS}}^{\text{auc}}.\text{Verify}^{\mathcal{G}_{\text{roR0}}, \mathcal{F}_{\text{CRS}}}(\text{PPM}, s, X, \Phi)$	$\Pi_{\text{RVS}}^{\text{auc}}.\text{Extract}(\text{PPM}, X, \Phi, Q_{P^*})$
<pre> 1: parse X = (x, CRS_s) 2: CRS'_s $\leftarrow \mathcal{F}_{\text{CRS}}.\text{Query}(s)$ 3: if CRS_s = CRS'_s \wedge 4: $\Pi_{\text{RVS}}^{\text{aSLC}}.\text{Verify}^{\mathcal{G}_{\text{roR0}}}(\text{PPM}, X, \Phi) = 1$: 5: return 1 6: else : 7: return 0 </pre>	<pre> 1: W $\leftarrow \Pi_{\text{RVS}}^{\text{aSLC}}.\text{Extract}(X, \Phi, Q_{P^*})$ 2: parse X = (x, CRS_s) 3: parse W = (w, trap_s) 4: if $R_{\text{RVS}}(X, W) = 1 \wedge R(x, w) = 0$: 5: return Fail 6: else : 7: return W </pre>
$\Pi_{\text{RVS}}^{\text{auc}}.\text{SimRand}(\text{PPM}, Z, X, \Phi, W, T')$	
<pre> 1: parse PPM = (ppm₀, ppm₁), Z = (z₀, z₁), X = (x, CRS_s) 2: parse $\Phi = (\pi_0, \pi_1, \text{CHL}), W = (w, 0), T' = (r'_0, r'_1)$ 3: r₀[*] $\leftarrow \Sigma_R.\text{SimRand}(\text{ppm}_0, z_0, x, \pi_0, w, r'_0)$ 4: r₁[*] $\leftarrow \Sigma_S.\text{RealToSim}(\text{ppm}_1, z_1, \text{CRS}_s, \pi_1, \text{trap}_s, r'_1)$ 5: T[*] = (r₀[*], r₁[*]) 6: return T[*] </pre>	

Fig. 9. Compiler $\Pi_{\text{RVS}}^{\text{auc}} \leftarrow \text{auc}(\Sigma_R, \text{SLC})$ for Σ_R in the $\mathcal{G}_{\text{roR0}}\text{-}\mathcal{F}_{\text{CRS}}$ -hybrid Model

CRS_s , the probability of finding trap_s such that $S(\text{CRS}_s, \text{trap}_s) = 1$ 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 $\Pi_{\text{RVS}}^{\text{SLC}}.\text{SimRand}$ 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 $\Pi_{\text{RVS}}^{\text{auc}}.\text{Extract}$ 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 $\Pi_{\text{RVS}}^{\text{auc}}.\text{Prove}$ 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 $\Pi_{\text{RVS}}^{\text{SLC}}$, with one caveat: we must argue that \mathcal{Z} is not able to use its interactions with the $\Pi_{\text{RVS}}^{\text{auc}}.\text{Extract}$ algorithm to help it distinguish between the real and simulated proofs. Following Lysyanskaya and Rosenbloom [28], we note that since $\Pi_{\text{RVS}}^{\text{auc}}.\text{Extract}$ 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 property, 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 $\Pi_{\text{RVS}}^{\text{auc}}.\text{SimSetup}$, $\Pi_{\text{RVS}}^{\text{auc}}.\text{SimProve}$, and $\Pi_{\text{RVS}}^{\text{auc}}.\text{SimRand}$. Recall that $\Pi_{\text{RVS}}^{\text{auc}}.\text{SimProve}$ is essentially $\Pi_{\text{RVS}}^{\text{auc}}.\text{Prove}$, except that \mathcal{C} generates and stores pairs $(\text{CRS}_s, \text{trap}_s)$ for each protocol session s in the list simcrs , and uses the witness trap_s as input to $\Pi_{\text{RVS}}^{\text{auc}}.\text{Prove}$ rather than a "real" witness w . We show that Hybrid 4 is indistinguishable from Hybrid 5 as long as $\Pi_{\text{RVS}}^{\text{auc}}$ is witness equivocal (WE), which follows from the adSHVZK property of Σ_{RVS} according to Theorem 1. Since the WE game captures only a single **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 $i+1^{\text{st}}$ query onwards are answered according to Hybrid 5. Our reduction \mathcal{B} will use an environment \mathcal{Z} that can distinguish between the i and $i+1^{\text{st}}$ with non-negligible advantage to win the WE game from Figure 4 with non-negligible advantage. \mathcal{B} proceeds as follows.

\mathcal{B} answers \mathcal{Z} 's RO queries as usual. For up to and including the i^{th} **Prove** query that \mathcal{Z} issues for a party P , \mathcal{B} computes and returns proofs using $\Pi_{\text{RVS}}^{\text{auc}}.\text{Prove}$, exactly as \mathcal{Z} expects from Hybrid 4; past the $i+1^{\text{st}}$ query, \mathcal{B} computes and returns proofs using $\Pi_{\text{RVS}}^{\text{auc}}.\text{SimProve}$, exactly as \mathcal{Z} expects from Hybrid 5. Along the way, \mathcal{B} stores the tuples (P, Φ_i, T_i) . When \mathcal{Z} issues the $i+1^{\text{st}}$ query (**Prove**, P , s, x_{i+1}, w_{i+1}) for some party P in session s , \mathcal{B} constructs a query (**Prove**, x_{i+1} , $\text{CRS}_s, w_{i+1}, \text{trap}_s, 0, \text{st}$) and forwards it to the WE challenger, who returns a response (Φ, T, st) . \mathcal{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 (**Corrupt**, P), \mathcal{B} gathers all of the tuples (P, Φ_i, T_i) and returns the concatenation of all T_i to \mathcal{Z} . If \mathcal{Z} outputs “Hybrid 3” then \mathcal{B} outputs 0; otherwise if \mathcal{Z} outputs “Hybrid 4” then \mathcal{B} outputs 1.

Note that if the WE 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 WE challenger is running the 1-bit experiment then it computed the proofs using the opposite witness trap_s for statement 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, \mathcal{B} succeeds in winning the WE game with the same probability as \mathcal{Z} succeeds in distinguishing Hybrids i and $i + 1$. Since the number of **Prove** queries must be polynomial in the security parameter, \mathcal{B} ’s total advantage in winning the WE game over all of the proofs is still non-negligible, completing the proof.

Hybrid 6. Identical to Hybrid 4 in Theorem 3. \square

6 Constructions via the Adaptive Fischlin Transform

In this section, we extend the randomized Fischlin transform [24,27] for the adaptive setting (Section 6.2), and prove that it is an adaptive SLC (Section 6.3).

6.1 Requirements of the Randomized Fischlin Transform

To stop \mathcal{A} from predicting com and querying the RO on (x, com) before the prover does, the 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 24 (Superlogarithmic Commitment Entropy). [24] *Let Σ_R be any Σ -protocol for binary \mathcal{NP} relation R and template τ as specified in Definition 9. Σ_R has superlogarithmic commitment entropy if for all $(x, w) \in L_R$, the min-entropy of $\text{com} \leftarrow \tau.\text{Commit}(x, w)$ is superlogarithmic in λ .*

The strong special soundness property [27] 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 $\text{chl} = \text{chl}'$, as long as $\text{res} \neq \text{res}'$.

Definition 25 (Strong Special Soundness). [27] *A Σ -protocol Σ_R for relation R (Definition 9) has the strong special soundness property if the condition $\text{chl} \neq \text{chl}'$ in the specification of the $\Sigma.\text{Extract}$ algorithm is replaced with the condition $(\text{chl}, \text{res}) \neq (\text{chl}', \text{res}')$.*

6.2 The Adaptive Randomized Fischlin Transform

The standard randomized Fischlin transform [24,27] is a straight-line compiler **rFis** [29] that transforms any Σ -protocol with certain general properties into a non-interactive, straight-line extractable (NISLE) proof system Π_R^{rFis} in the ROM. The prover in the randomized Fischlin transform 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 algorithm $\Pi_R^{\text{rFis}}.\text{Extract}$ takes these proofs as input (via the adversary’s oracle query history $\mathcal{Q}_{\mathcal{A}}$), and can therefore extract a witness for valid, adversarially-created proofs without any further interaction with the prover.

We argue in this section that our *adaptive* randomized Fischlin transform **aFis**, which extends **rFis** with a **SimRand** functionality, preserves the adaptive security property of the underlying Σ -protocol: we will show that as long as Σ_R is adaptive and conforms to standards of **rFis** discussed in the previous section, **aFis** is an adaptive SLC. The $\Pi_R^{\text{aFis}}.\text{SimRand}$ 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 $\tau.\text{Commit}$), the $\Pi_R^{\text{aFis}}.\text{SimRand}$ algorithm runs $\Sigma_{R,\tau}.\text{SimRand}$ using auxiliary output from $\Pi_R^{\text{aFis}}.\text{SimProve}$. To simulate the random “challenge-selection” section of Q (i.e. the section used to generate the prover’s random selection of challenges), $\Pi_R^{\text{aFis}}.\text{SimRand}$ first gathers the random coins corresponding to all of the challenges sampled by $\Pi_R^{\text{aFis}}.\text{SimProve}$. 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, $\Pi_R^{\text{aFis}}.\text{SimRand}$ 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 [24,27]. 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 $\Pi_R^{\text{aFis}}.\text{SimProve}$ (but before issuing a corruption to receive the output of $\Pi_R^{\text{aFis}}.\text{SimRand}$), test outputs of the RO on transcripts it generates using w_0 or w_1 and the same commitments as $\Pi_R^{\text{aFis}}.\text{SimProve}$, but different challenges.

In the original version of the transform, the **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 [24]. It then programs the RO to return the all-zero string for the simulated proof transcript [24] and outputs the proof. Whenever the adversary makes any queries of the form described above, **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' .

Thankfully, Kondi and shelat's randomization of the challenge space allows the Π_R^{aFis} .`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 μ , and inserting the challenge used to compute the simulated transcript as the first input to produce the all-zero string. Due to subtle technical difficulties that are the subject of future work, 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, but not all (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.

We will show that as long as the Σ -protocol used as input to the compiler is adaptive special honest-verifier zero-knowledge (adSHVZK), the output of the adaptive simulator algorithms Π_R^{aFis} .`SimProve` and Π_R^{aFis} .`SimRand` are indistinguishable from that of Π_R^{aFis} .`Prove`.

Definition 26 (Adaptive Randomized Fischlin Transform). *Let $\Sigma_{R,\tau}$ be any adaptive Σ -protocol for relation R (Definition 9) based on protocol template τ (Definition 8) with the required properties for `rFis` (Definitions 24 and 25). Let H be any random oracle. Then the randomized Fischlin transform of $\Sigma_{R,\tau}$, denoted `aFis`, is an algorithm that takes $\Sigma_{R,\tau}$ as input and creates a tuple of algorithms $\Pi_R^{\text{aFis}} = (\text{Setup}^H, \text{Prove}^H, \text{Verify}^H, \text{SimSetup}, \text{SimProve}, \text{SimRand}, \text{Extract})$, defined as follows.*

- `ppm` \leftarrow `Setup` ^{H} (1^λ) : H is fixed. Let b, n, t be set according to the randomized Fischlin transform [27, 24]: $bn = \omega(\log \lambda)$, $2^{t-b} = \omega(\log \lambda)$, $b, n, t = O(\log \lambda)$ 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 $\Sigma_{R,\tau}$.⁵ Run $\text{ppm}_\Sigma \leftarrow \tau.\text{Setup}(1^\lambda)$. Output the public parameters $\text{ppm} = (\text{ppm}_\Sigma, b, n, t, \ell, \lambda_Q)$.

- $(x, \Phi) \leftarrow \text{Prove}^H(x, w, Q)$: Compute the vector of n first messages $\overline{\text{com}} = \langle \text{com}_1, \dots, \text{com}_n \rangle$ by running $\tau.\text{Commit}(x, w, r_i)$ 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 chl_j from Q and compute $\text{res}_j \leftarrow \tau.\text{Respond}(x, w, \text{com}_i, \text{chl}_j)$ until $H(x, \overline{\text{com}}, i, \text{chl}_j, \text{res}_j) = 0^b$, then set $\pi_i = (\text{com}_i, \text{chl}_j, \text{res}_j)$. Finally, return (x, Φ) , where $\Phi = (\pi_1, \dots, \pi_n)$.
- $\{0, 1\} \leftarrow \text{Verify}^H(x, \Phi)$: Parse $\Phi = (\pi_1, \dots, \pi_n)$. Output 1 (accept) if and only if $\Sigma_R.\text{Verify}(x, \pi_i) = 1$ and $\sum_{i=1}^n H(x, \overline{\text{com}}, i, \text{chl}_i, \text{res}_i) = 0^b$ for $1 \leq i \leq n$. Otherwise, output 0 (reject).
- $(\text{ppm}, z_s) \leftarrow \text{SimSetup}(1^\lambda)$: Fix H and generate b, n, t the same as in Setup. Generate ppm_Σ and simulator state information z_s by running $\Sigma_{R,\tau}.\text{SimSetup}$. Set $\text{ppm} = (\text{ppm}_\Sigma, b, n, t, \lambda_R)$ and return (ppm, z_s) .
- $(x, \Phi, Z_\Phi) \leftarrow \text{SimProve}(x, z_s, Q')$: For each proof $1 \leq i \leq n$, sample a random r_i and a random challenge chl_i from Q' . Then run $\Sigma_{R,\tau}.\text{SimProve}(x, z_s, \text{chl}_i, r_i)$ to obtain $\pi_i = (\text{com}_i, \text{chl}_i, \text{res}_i)$ and z_{π_i} . For each proof, program the output of H on input $(x, \overline{\text{com}}, i, \text{chl}_i, \text{res}_i)$ to be 0^b . Finally, output the proof tuple (x, Φ) and auxiliary information Z_Φ , where $\Phi = (\Phi_1, \dots, \Phi_n)$ and $Z_\Phi = (z_{\pi_1}, \dots, z_{\pi_n})$.
- $Q \leftarrow \text{SimRand}(\text{ppm}, z_s, Z_\Phi, x, \Phi, w, Q')$: Parse $\Phi = (\pi_1, \dots, \pi_n)$ and $Z_\Phi = (z_{\pi_1}, \dots, z_{\pi_n})$. Reconstruct the proof randomness Q corresponding to Φ as follows. For each proof $\pi_i = (\text{com}_i, \text{chl}_i, \text{res}_i)$, run $\Sigma_{R,\tau}.\text{SimRand}(\text{ppm}, z_s, z_{\pi_i}, x, \pi_i, w)$ to obtain r_i , and concatenate it to Q . Then, sample a fresh random challenge chl_j from Q' and compute $\text{res}_j \leftarrow \tau.\text{Respond}(\text{ppm}, x, w, \text{com}_j, \text{chl}_j)$. If $H(x, \overline{\text{com}}, j, \text{chl}_j, \text{res}_j) = 0^b$, concatenate chl_j to Q and move on to the next proof π_{i+1} ; otherwise, concatenate chl_j to Q and continue sampling challenges until the condition is met. Finally, return Q .
- $w \leftarrow \text{Extract}(X, \Phi, Q_{\mathcal{A}})$: Parse $\Phi = (\pi_1, \dots, \pi_n)$ and each $\pi_i = (\text{com}_i, \text{chl}_i, \text{res}_i)$. Given a list $Q_{\mathcal{A}}$ the adversary's queries to H , search for two queries $(x, \overline{\text{com}}, i, \text{chl}_i, \text{res}_i)$ and $(x, \overline{\text{com}}, i, \text{chl}'_i, \text{res}'_i)$ such that $(\text{chl}_i, \text{res}_i) \neq (\text{chl}'_i, \text{res}'_i)$ and $\Sigma_R.\text{Verify}(x, \pi_i) = \Sigma_R.\text{Verify}(x, \pi'_i) = 1$. If no such queries exist, output Fail. Otherwise, obtain w by running $\Sigma_R.\text{Extract}(x, \pi, \pi')$.

⁵ For each execution of $\Pi_R^{\text{afis}}.\text{Prove}$, we will need enough randomness to compute n proofs, each requiring λ_r bits to compute the first message and $2^t t$ bits to sample 2^t t -bit random challenges. For each execution of simulator algorithms $\Pi_R^{\text{afis}}.\text{SimProve}$ and $\Pi_R^{\text{afis}}.\text{SimRand}$, we will need enough randomness to sample 2^t b -bit random challenges and $n(\ell - b)$ bits to program the random oracle, as well $n(\lambda_{r'})$ bits to feed into n executions of $\Sigma_{S,\tau}.\text{SimRand}$ and $n2^t t$ bits to simulate the challenge selection process.

6.3 Adaptive Randomized Fischlin is an Adaptive SLC

We now prove that the adaptive randomized Fischlin transform \mathbf{aFis} qualifies as an adaptive SLC, and can therefore efficiently bootstrap adaptive Σ -protocols into adaptive UC NIZKPoK in the global ROM(s).

Theorem 5. *Let $\Sigma_{R,\tau}$ be an adaptive Σ -protocol based on protocol template τ for relation R (Definition 9) with the required properties for \mathbf{rFis} (Definitions 24 and 25), and $H : \{0,1\}^* \rightarrow \{0,1\}^b$ be any random oracle. Then the adaptive randomized Fischlin transform \mathbf{aFis} (Definition 26) is an adaptive straight-line compiler for $\Sigma_{R,\tau}$ (Definition 16) in the random-oracle model.*

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 $\Pi_R^{\mathbf{aFis}}$.**Prove** does not functionally change between \mathbf{rFis} and \mathbf{aFis} (as our explicit treatment of the prover's randomness is syntactic), \mathbf{aFis} is as complete as \mathbf{rFis} [24,27]. We proceed by contrapositive to show that if $\Pi_R^{\mathbf{aFis}}$ is not additionally adNIM-SHVZK and adNI-SSS, then the underlying Σ -protocol Σ_R cannot be regular adaptive special honest-verifier zero-knowledge (adSHVZK), contradicting the assumption in the theorem statement.

We begin by constructing a reduction \mathcal{B} that uses an algorithm \mathcal{A} that can win the adNIM-SHVZK game (Definition 18) parameterized over $\Pi_R^{\mathbf{aFis}}$ with non-negligible advantage as a black-box in order to win the adSHVZK game (Definition 11) parameterized over Σ_R with non-negligible advantage. \mathcal{B} proceeds as follows. When it obtains \mathbf{ppm} from the adSHVZK challenger, \mathcal{B} passes \mathbf{ppm} to \mathcal{A} . Note that the adSHVZK challenger is expecting exactly one **Prove** query (i.e. it is not *multi*-adaptive), so we modify \mathcal{A} to distinguish two hybrids $i-1$ and i , where in the $i-1$ st hybrid, the first $i-1$ proofs and random coins are according to $\Pi_R^{\mathbf{aFis}}$.**Prove** using randomness $r \in \{0,1\}^{\lambda_r}$, and the i th onward are according to $\Pi_R^{\mathbf{aFis}}$.**SimProve** and $\Pi_R^{\mathbf{aFis}}$.**SimRand** using randomness $r' \in \{0,1\}^{\lambda_{r'}}$. \mathcal{B} models the RO with an initially empty list $L \leftarrow \perp$. To answer the first $i-1$ queries, \mathcal{B} first samples $f \leftarrow_{\mathcal{S}} F$ then runs H_f on the query, storing the input-output pair in L . From the i th query onward, \mathcal{B} programs the RO using the interface \mathbf{Prog}_L according to $\Pi_R^{\mathbf{aFis}}$.**SimProve** and $\Pi_R^{\mathbf{aFis}}$.**SimRand**.

For the first $i-1$ queries (**Prove**, x, w) from \mathcal{A} , \mathcal{B} samples randomness $Q \leftarrow \{0,1\}^{\lambda_Q}$, runs $\Pi_R^{\mathbf{aFis}}$.**Prove**(x, w, Q), and returns (Φ, Q) to \mathcal{A} . On \mathcal{A} 's i th query (**Prove**, x^*, w^*), \mathcal{B} passes (**Prove**, x^*, w^*) to its challenger and receives (π^*, r^*) for $\pi^* = (\mathbf{com}^*, \mathbf{chl}^*, \mathbf{res}^*)$ that is either the result of running Σ_R .**Prove** on randomness r^* or the result of running Σ_R .**SimProve** and Σ_R .**SimRand**. Note that that \mathcal{A} is expecting more proofs than just π^* , as the output of $\Pi_R^{\mathbf{aFis}}$.**Prove** and $\Pi_R^{\mathbf{aFis}}$.**SimProve** should actually be a tuple of proofs $\Phi_i = \pi_1, \dots, \pi_n$. 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, \mathcal{A} is expecting more randomness in the prover's tape 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 $t\ell$ -length segment corresponding to the randomness used to sample the 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 . We can then modify \mathcal{A} again to distinguish the hybrids i, j from $i, j + 1$, where in the i, j^{th} hybrid, all of the proofs up to and including π_j are computed according to $\Pi_R^{\text{aFis}}.\text{Prove}$ and Q_j is the prover's random tape, while in the $i, j + 1^{\text{st}}$ hybrid onward, π_j is computed according to $\Pi_R^{\text{aFis}}.\text{SimProve}$ and Q_j is generated according to $\Pi_R^{\text{aFis}}.\text{SimRand}$. \mathcal{B} constructs the hybrid proof tuple Φ_i from the challenge query as follows.

Let the j^{th} commitment $\text{com}_j = \text{com}^*$, and $\pi_j = \pi^* = (\text{com}^*, \text{chl}^*, \text{res}^*)$. Since it must have the entire tuple $\text{com}_1, \dots, \text{com}_n$ of commitments before running $\Pi_R^{\text{aFis}}.\text{Prove}$, \mathcal{B} starts by running the steps of $\Pi_R^{\text{aFis}}.\text{SimProve}$ $n - j$ times to obtain π_{j+1}, \dots, π_n . \mathcal{B} then samples each r_1, \dots, r_{j-1} from Q' and runs $\tau.\text{Commit}$ $j - 1$ times to obtain $\text{com}_1, \dots, \text{com}_{j-1}$. It then finalizes the commitment vector $\overline{\text{com}} = \text{com}_1, \dots, \text{com}_n$ and programs H to output 0^b on input $(x, \overline{\text{com}}, k, \text{chl}_k, \text{res}_k)$ for each simulated proof π_k , where $k \in [j + 1, n]$. \mathcal{B} computes each proof π_1, \dots, π_{j-1} by running the steps of $\Pi_R^{\text{aFis}}.\text{Prove}$ starting right after the generation of the commitment vector.

\mathcal{B} generates the prover's hybrid random tape Q for the proof tuple Φ as follows. For each of the first chunks $Q_m \in Q_1, \dots, Q_{j-1}$, \mathcal{B} concatenates the first-message randomness r_m and challenge randomness $\text{chl}_1, \dots, \text{chl}_t$ it used in the execution of $\Pi_R^{\text{aFis}}.\text{Prove}$. The j^{th} chunk, which must contain the $\lambda_r + t\ell$ bits corresponding to the challenge proof $\pi^* = \pi_j$, are generated as follows. \mathcal{B} sets the first λ_r bits to be the value r^* returned by the adSHVZK challenger. \mathcal{B} then samples a fresh random challenge chl_m , computes $\text{res}_m \leftarrow \tau.\text{Respond}(\text{ppm}, x, w, \text{com}^*, \text{chl}_m)$, and checks whether $H(x, \overline{\text{com}}, j, \text{chl}_m, \text{res}_m) = 0^b$. If it does, \mathcal{B} concatenates chl^* to Q and moves on to the next step. Otherwise, \mathcal{B} concatenates chl_m to Q and continues sampling challenges until the condition is met. Finally for the remaining chunks $Q_m \in Q_{j+1}, \dots, Q_n$, \mathcal{B} concatenates the first-message randomness r_m and challenge randomness $\text{chl}_1, \dots, \text{chl}_t$ returned by executing $\Pi_R^{\text{aFis}}.\text{SimRand}$ on input the proofs π_{j+1}, \dots, π_n and simulation trapdoors $z_{\pi_{j+1}}, \dots, z_{\pi_n}$ output by its execution of SimProve .

Finally, \mathcal{B} returns Φ and Q to \mathcal{A} .

For the analysis, consider that \mathcal{B} 's execution of the proofs π_1, \dots, π_j and π_{j+2}, \dots, π_n and corresponding randomness is exactly what \mathcal{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 \mathcal{A} expects. If the adSHVZK challenger is playing with bit $b = 0$ and generated (π^*, r^*) according to $\Pi_R^{\text{aFis}}.\text{Prove}$ 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 $\text{com}^* = \text{com}_{j+1}$) will be exactly what \mathcal{A} expects from the $i, j + 1^{\text{st}}$ hybrid. If, on the other hand, π^*, r^* were generated according to $\Pi_R^{\text{aFis}}.\text{SimProve}$ and $\Pi_R^{\text{aFis}}.\text{SimRand}$ respec-

tively, then \mathcal{A} 's view of π_{j+1} and r_{j+1} in Q_j will be according to the hybrid i, j .

For the challenge section of Q_{j+1} , note that the challenge selection process of $\Pi_R^{\text{aFis}}.\text{Prove}$ is identical to that of $\Pi_R^{\text{aFis}}.\text{SimRand}$ up to the challenge in the t^{th} transcript—the **Prove** and **SimRand** algorithms sample challenges and compute responses using the real witness until they find a transcript that causes the RO to return 0^b . When this condition occurs, the **Prove** algorithm returns the accepting transcript computed using the challenge it sampled, while the **SimRand** algorithm inserts the challenge and transcript generated by **SimProve**. For the challenge section of Q_{j+1} , \mathcal{B} samples challenges according to either algorithm and when it finds a transcript that returns 0^b , it inserts the challenge chl^* from the adSHVZK challenger. \mathcal{A} 's view of the challenge selection section in the case that $b = 0$ is therefore identical to hybrid $i, j + 1$, while if $b = 1$, its view is identical to hybrid i, j .

Therefore, if \mathcal{B} outputs 0 when \mathcal{A} outputs “Hybrid $i, j + 1$ ” and 1 when \mathcal{A} outputs “Hybrid i, j ,” it succeeds with the same probability as \mathcal{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 \mathcal{B} 's summed advantage over all of the hybrids is negligible, completing the proof that Π_R^{aFis} must be adNIM-SHVZK.

We now use the fact that Π_R^{aFis} is adNIM-SHVZK to argue that Π_R^{aFis} must also be adaptive non-interactive special simulation-sound (adNI-SSS). Consider a new reduction \mathcal{B} that uses an adversary \mathcal{A} that can win the adNI-SSS game (Definition 19), as well as a second adversary \mathcal{A}' that can win the regular non-interactive special soundness (NI-SS) game (Definition 20 in Appendix ??) with non-negligible advantage as black boxes to win the adNIM-SHVZK game (Definition 18). \mathcal{B} forwards all of \mathcal{A} 's **Prove** queries to and from the adNIM-SHVZK challenger, and all of \mathcal{A} 's random oracle queries to and from H . Similarly, \mathcal{B} forwards all of \mathcal{A}' 's oracle queries to and from H . Whenever \mathcal{A} produces a fresh (non-simulated) proof (x, Φ) such that $\Pi_R^{\text{aFis}}.\text{Verify}(x, \Phi) = 1$, \mathcal{B} computes $w \leftarrow \Pi_R^{\text{aFis}}.\text{Extract}(x, \Phi, \mathcal{Q}_{\mathcal{A}})$ and checks whether $R(x, w) = 0$, which happens with non-negligible probability by assumption. \mathcal{B} does the same for \mathcal{A}' 's proofs (x', Φ') . (The argument that there must be sufficient queries in $\mathcal{Q}_{\mathcal{A}}$ or $\mathcal{Q}_{\mathcal{A}'}$ to invoke $\Pi_R^{\text{aFis}}.\text{Extract}$ in the first place is identical to the arguments given by Kondi and shelat [27]; in brief, the strong special soundness property stops \mathcal{A} from being able to “tweak” an old proof to create a forgery, for instance by changing one response.

If the adNIM-SVHZK challenger is playing with bit $b = 1$ and the proofs and randomness being passed to \mathcal{A} are according to $\Pi_R^{\text{aFis}}.\text{SimProve}$ and $\Pi_R^{\text{aFis}}.\text{SimRand}$, this is exactly what \mathcal{A} expects from the adNI-SSS game, and \mathcal{B} 's advantage is the same as \mathcal{A} 's. If the adNIM-SHVZK challenger is playing with bit $b = 0$, then \mathcal{A} 's advantage reduces to \mathcal{A}' 's advantage in the standard NI-SS game (since \mathcal{A}' can always generate honest proofs and randomness according to $\Pi_R^{\text{aFis}}.\text{Prove}$ itself). Assume for a contradiction that there is a non-negligible difference between the extraction failures produced by \mathcal{A} when $b = 1$ and $\mathcal{A} / \mathcal{A}'$

when $b = 0$. If \mathcal{B} observes a difference in output between \mathcal{A} and \mathcal{A}' , \mathcal{B} knows \mathcal{A} has an extra advantage due to seeing simulated proofs and randomness and outputs 1; otherwise if there is no difference, \mathcal{B} outputs 0. Therefore, the difference must be negligible, implying that as long as Π_R^{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), Π_R^{aFis} must be adNI-SSS.

We have shown that the tuple $\Pi_R^{\text{aFis}} \leftarrow \text{aFis}(\Sigma_{R,\tau})$ is overwhelmingly complete, adNIM-SVHZK, and adNI-SSS, completing the proof that **aFis** is an adaptive straight-line compiler. \square

6.4 Realizing Efficient and Adaptive aUC NIZKPoK from Σ -protocols in the Global ROM(s)

We demonstrated in Section 5.2 that any adaptive SLC is sufficient to convert any Σ -protocol Σ_R into a NISLE proof system Π_R^{SLC} that aUC-realizes $\mathcal{F}_{\text{aNIZK}}$ in the $\mathcal{G}_{\text{rpoRO}}$ -hybrid model. In the previous section, we proved that the adaptive randomized Fischlin transform **aFis** is an adaptive SLC for any Σ -protocol that has either the strong special soundness property. Therefore, we can efficiently aUC-realize $\mathcal{F}_{\text{aNIZK}}$ in the programmable global ROM using **aFis** and any such Σ -protocol.

Corollary 1. *Let Σ_R be any adaptive Σ -protocol for a relation R (Definition 9) with the required properties for **aFis** (Definitions 24 and 25), and **aFis** be the adaptive randomized Fischlin transform (Definition 26). Then the NISLE proof system $\Pi_R^{\text{SLC}} \leftarrow \text{aFis}(\Sigma_R)$ aUC-realizes $\mathcal{F}_{\text{aNIZK}}$ in the $\mathcal{G}_{\text{rpoRO}}$ -hybrid model (Definition 7 where $\mathcal{G}_{\text{RO}} := \mathcal{G}_{\text{rpoRO}}$).*

Proof. The corollary follows directly from Theorems 3 and 5. \square

Similarly, we showed in Section 5.3 that any adaptive SLC is sufficient in conjunction with the adaptive OR-protocol compiler **aguc** from Definition 23 to convert any Σ -protocol into a NISLE proof system $\Pi_{\text{RVS}}^{\text{auc}}$ that aUC-realizes $\mathcal{F}_{\text{aNIZK}}$ in the $\mathcal{G}_{\text{roRO}}\text{-}\mathcal{F}_{\text{CRS}}$ -hybrid model. Therefore, we can efficiently aUC-realize $\mathcal{F}_{\text{aNIZK}}$ in the non-programmable global ROM using **aguc**, **aFis** and any Σ -protocol that is compatible with **aFis**.

Corollary 2. *Let Σ_R be any adaptive Σ -protocol for a relation R (Definition 9) with the required properties for **aFis** (Definitions 24 and 25), **aguc** be the adaptive OR-protocol compiler (Definition 23), and **aFis** be the adaptive randomized Fischlin transform (Definition 26). Then the NISLE proof system $\Pi_{\text{RVS}}^{\text{auc}} \leftarrow \text{aguc}(\Sigma_R, \text{aFis})$ aUC-realizes $\mathcal{F}_{\text{aNIZK}}$ in the $\mathcal{G}_{\text{roRO}}\text{-}\mathcal{F}_{\text{CRS}}$ -hybrid model (Definition 7 where $\mathcal{G} := \mathcal{G}_{\text{roRO}}$ and $*$:= \mathcal{F}_{CRS}).*

Proof. The corollary follows directly from Theorems 4 and 5. \square

7 Practical Adaptive Σ -protocols

In this section, we recall the abstract treatment of identification schemes from linear function families due to Hauck, Kiltz, and Loss [26], and remodel it to capture many simple Σ -protocols (Section 7.1). We then give three satisfying instantiations: proofs of knowledge of a discrete logarithm (Section 7.2) and n equivalent representations of m witnesses (Section 7.3). As proven in Sections 5 and 6.2, any Σ -protocol that qualifies as adaptive can be efficiently bootstrapped into an adaptive UC NIZKPoK in the programmable global ROM, and in the non-programmable global ROM as long as the Σ -protocol is also randomness equivocal. We therefore conclude with an abstract blueprint and three instantiations of efficient, composable, and adaptively-secure NIZKPoK for a variety of real-world applications.

7.1 Simple Adaptive Σ -protocol Abstraction

Hauck et al.’s construction of “canonical identification schemes” from pseudomodules and linear function families [26] 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`, and `Extract` algorithms.

For simplicity and since all of our instantiations are parameterized over modules, we narrow the focus of our abstraction to modules rather than pseudomodules. Briefly, 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 $\text{chl}, \text{chl}' \in \mathcal{C}$ and $x, y \in \mathcal{X}$, there exists a map $\mathcal{C} \times \mathcal{X} \rightarrow \mathcal{X}$ satisfying the following properties: 1) $\text{chl} \cdot (x + y) = \text{chl} \cdot x + \text{chl} \cdot y$, 2) $(\text{chl} + \text{chl}') \cdot x = \text{chl} \cdot x + \text{chl}' \cdot x$, 3) $(\text{chl} \cdot \text{chl}') \cdot x = \text{chl} \cdot (\text{chl}' \cdot x)$, and 4) $1_{\mathcal{C}} \cdot x = x$. In order to guarantee the special soundness property, we additionally require that for any two proofs $\pi = (\text{com}, \text{chl}, \text{res})$ and $\pi' = (\text{com}, \text{chl}', \text{res}')$ for x such that $\text{Verify}(x, \pi) = \text{Verify}(x, \pi') = 1$ and $\text{chl} \neq \text{chl}' \neq 0_{\mathcal{C}}$, the inverse of $(\text{chl} - \text{chl}')$ exists. We write the inversion $(\text{chl} - \text{chl}') \cdot (\text{chl} - \text{chl}')^{-1} = 1_{\mathcal{C}}$.⁶ We highlight the properties of linear function families that are critical to understanding our results as part of the proof below.

Definition 27 (Simple Adaptive Σ -protocol Candidate). *The simple adaptive Σ -protocol candidate Σ_{sim} for relation R is a tuple of efficient procedures $\Sigma_{\text{sim}} = (\text{Setup}, \text{Prove}, \text{Verify}, \text{SimSetup}, \text{SimProve}, \text{SimRand}, \text{RealToSim}, \text{Extract})$ that adhere to the following format:*

⁶ 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 $(\text{chl} - \text{chl}')$ directly (in the case of the RSA inverse, the extractor uses Shamir’s trick [10]). For such constructions, it might be convenient to use the abstraction for the adaptive SHVZK property and treat the special soundness property separately.

- $\text{ppm} \leftarrow \text{Setup}(1^\lambda)$: Define the linear function family $\text{LF} = (\text{PGen}, \text{F})$ (Definition 3.1 in [26]) such that $\text{ppm} \leftarrow \text{PGen}(1^\lambda)$ 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 $\text{F} : \mathcal{W} \rightarrow \mathcal{X}$.
- $\pi \leftarrow \text{Prove}(\text{ppm}, x, w, r)$: P sends V a first message $\text{com} = \text{F}(r)$. V replies with a challenge $\text{chl} \leftarrow_{\S} \mathcal{C}$. P responds with $\text{res} = \text{chl} \cdot w + r$.
- $\{0, 1\} \leftarrow \text{Verify}(\text{ppm}, x, \pi)$: Parse π as $(\text{com}, \text{chl}, \text{res})$. If $\text{F}(\text{res}) = \text{chl} \cdot x + \text{com}$, output 1 (accept). Otherwise, output 0 (reject).
- $\text{ppm} \leftarrow \text{SimSetup}(1^\lambda)$: Invoke $\text{Setup}(1^\lambda)$ and return ppm .
- $\pi \leftarrow \text{SimProve}(\text{ppm}, x, \text{chl}, r')$: Parse $r' = \text{res}$ and compute $\text{com} = \text{F}(\text{res} - \text{chl} \cdot w)$. Return $\pi = (\text{com}, \text{chl}, \text{res})$.
- $r \leftarrow \text{SimRand}(\text{ppm}, x, \pi, w)$: Parse $\pi = (\text{com}, \text{chl}, \text{res})$. Compute and output $r = \text{res} - (\text{chl} \cdot w)$.
- $r' \leftarrow \text{RealToSim}(\text{ppm}, \pi)$: Parse $\pi = (\text{com}, \text{chl}, \text{res})$ and output $r' = \text{res}$.
- $w \leftarrow \text{Extract}(\text{ppm}, x, \pi, \pi')$: Given any two proofs $\pi = (\text{com}, \text{chl}, \text{res})$ and $\pi' = (\text{com}, \text{chl}', \text{res}')$ such that $\text{Verify}(x, \pi) = \text{Verify}(x, \pi') = 1$ and $\text{chl} \neq \text{chl}'$, compute and output $w = (\text{res} - \text{res}') \cdot (\text{chl} - \text{chl}')^{-1}$.

Theorem 6. *The simple adaptive Σ -protocol candidate given above (Definition 27) is an adaptive Σ -protocol (Definition 3) with randomness equivocability (Definition ??).*

Proof. The following proof relies on two core properties of the evaluation function F , described below (for details in context, see Definition 4.1 [26]). First, the *pseudo-module homomorphism* (PMH) property states that for all $y, z \in \mathcal{W}$, $\text{F}(\text{chl} \cdot y + z) = \text{chl} \cdot \text{F}(y) + \text{F}(z)$. Second, the *smoothness property* guarantees that for all $y \in \mathcal{W}$, $\text{F}(y)$ 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.

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

Adaptive SHVZK. As long as $r \leftarrow_{\S} \mathcal{R}$ and $\text{chl} \leftarrow_{\S} \mathcal{C}$, the distribution of $\text{res} = \text{chl} \cdot w + r$ computed in $\Sigma_{\text{sim}}.\text{Prove}$ is uniformly random in \mathcal{B} . The $\text{res} \leftarrow_{\S} \mathcal{B}$ and $\text{chl} \leftarrow_{\S} \mathcal{C}$ sampled by $\Sigma_{\text{sim}}.\text{SimProve}$ are therefore identical to the res and chl in a real proof. By the smoothness of F , $\Sigma_{\text{sim}}.\text{SimProve}$'s $\text{com} = \text{F}(\text{res} - \text{chl} \cdot w)$ is also uniformly random in \mathcal{A} . Since res and chl are uniformly random in \mathcal{B} and \mathcal{C} respectively, the distribution of $r = \text{res} - (\text{chl} \cdot w)$ will be similarly random. Note that both simulations are correct, since

$$\text{F}(r) = \text{F}(\text{res} - \text{chl} \cdot w) = -\text{chl} \cdot \text{F}(w) + \text{F}(\text{res}) = -\text{chl} \cdot x + \text{chl} \cdot x + \text{com} = \text{com}$$

by the PMH property. The outputs of $\Sigma_{\text{sim}}.\text{Prove}$ and r are therefore distributed identically to the outputs of $\Sigma_{\text{sim}}.\text{SimProve}$ and $\Sigma_{\text{sim}}.\text{SimRand}$, respectively. Finally, in order to make a real proof look simulated, the RealToSim algorithm simply pretends that the uniform randomness of res formed the simulator's random tape.

Special Soundness. Solving the two verification equations for com and setting them equal, we get $F(\text{res}) - \text{chl} \cdot x = F(\text{res}') - \text{chl}' \cdot x$, which implies

$$F(w) \cdot \text{chl} - F(w) \cdot \text{chl}' = F(\text{res}) - F(\text{res}') \quad (\text{substitution and commutativity})$$

$$F(w) \cdot (\text{chl} - \text{chl}') = F(\text{res}) - F(\text{res}') \quad (\text{distributivity})$$

$$F(w) = F(\text{res}) - F(\text{res}') \cdot (\text{chl} - \text{chl}')^{-1} \quad (\text{invertability of } \text{chl} - \text{chl}')$$

$$\Rightarrow F(w) = F((\text{res} - \text{res}') \cdot (\text{chl} - \text{chl}')^{-1}) \quad (\text{PMH property})$$

$$\Rightarrow w = (\text{res} - \text{res}') \cdot (\text{chl} - \text{chl}')^{-1}. \quad \square$$

7.2 Adaptive Proof of Knowledge of Discrete Logarithm

Proofs of knowledge of discrete logarithm (PoK DL), also known as Schnorr proofs, are the backbone of many practical constructions of group [12,5], threshold [3], blind [26], and multi- [3,22] signatures.

The public parameters 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 $F : \mathbb{Z}_q \rightarrow \mathbb{G}$ is defined such that $F(w) = w \cdot g$. 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 be 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 $F(w) := w \cdot g \Rightarrow F(\text{chl} \cdot w + r) = (\text{chl} \cdot w + r) \cdot g = \text{chl} \cdot w \cdot g + r \cdot g = \text{chl} \cdot F(w) + F(r)$. Finally, F is smooth, since for $r \leftarrow_{\S} \mathbb{Z}_q$, $F(r) = r \cdot g$ is uniformly distributed over \mathbb{G} .

7.3 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 [36,30], accumulators [2,10], and anonymous credentials [9], 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 proofs of knowledge of representation [13,11] and Okamoto-Schnorr [31], and that the following result holds for those narrower instantiations as well.

Theorem 7 (PoK EqRep is an Adaptive Σ -protocol). *A proof of knowledge of equality of n representations can be described as a simple adaptive Σ -protocol with randomness equivocability (Definition 27).*

Proof. We recall the setup. The public parameters ppm are $(\mathbb{G}, q, \bar{g}_1, \dots, \bar{g}_n)$, where \mathbb{G} is a cyclic group of prime order q , and each \bar{g}_i is a vector of m generators $g_{i,1}, \dots, g_{i,m}$. 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 $F : \mathbb{Z}_q^m \rightarrow \mathbb{G}^n$ is defined such that

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

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

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

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

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

$$\begin{aligned} F(\text{res}) &= F(\text{res}_1, \dots, \text{res}_m) = \prod_{i=1}^m \text{res}_i \cdot g_{1,i}, \dots, \prod_{i=1}^m \text{res}_i \cdot g_{n,i} = \\ &= \prod_{i=1}^m (\text{chl} \cdot w_i + r_i) \cdot g_{1,i}, \dots, \prod_{i=1}^m (\text{chl} \cdot w_i + r_i) \cdot g_{n,i} = \\ &= \text{chl} \cdot F(w) + F(r) = \text{chl} \cdot x + \text{com}. \end{aligned}$$

The `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 $\text{com}_1, \dots, \text{com}_n$ as

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

and each random value r_i output by `SimRand` is computed $\text{res}_i - (\text{chl} \cdot w_i)$ for $1 \leq i \leq m$. Similarly, each witness w_i is extracted by computing $(\text{res}_i - \text{res}'_i) \cdot (\text{chl} - \text{chl}')^{-1}$, where $(\text{chl} - \text{chl}') \neq 0$ 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 ??:

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

$$\begin{aligned}
& \prod_{i=1}^m (\text{chl} \cdot w_i + r_i) \cdot g_{1,i}, \dots, \prod_{i=1}^m (\text{chl} \cdot w_i + r_i) \cdot g_{n,i} = \\
& \prod_{i=1}^m (\text{chl} \cdot w_i) \cdot g_{1,i} + r_i \cdot g_{1,i}, \dots, \prod_{i=1}^m (\text{chl} \cdot w_i) \cdot g_{n,i} + r_i \cdot g_{n,i} = \\
& \text{chl} \cdot \prod_{i=1}^m w_i \cdot g_{1,i} + r_i \cdot g_{1,i}, \dots, \text{chl} \cdot \prod_{i=1}^m w_i \cdot g_{n,i} + r_i \cdot g_{n,i} = \\
& \text{chl} \cdot \mathbf{F}(w) + \mathbf{F}(r).
\end{aligned}$$

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

Acknowledgements

Many thanks to Yashvanth Kondi for ongoing fruitful conversations about the properties of straight-line transformations, and to Ran Cohen for pointing our attention to the universal composability with global subroutines model.

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 Yakubov. Accumulators with applications to anonymity-preserving revocation. In *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 301–315. IEEE, 2017.
3. 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.
4. 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.
5. 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.
6. 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.
7. 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.

8. 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.
9. 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.
10. Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *CRYPTO*, pages 61–76. Springer, 2002.
11. Jan Camenisch and Victor Shoup. Practical verifiable encryption and decryption of discrete logarithms. In *CRYPTO 2003*, pages 126–144. Springer, 2003.
12. Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups. In *CRYPTO 1997*, pages 410–424. Springer, 1997.
13. Jan Camenisch and Markus Stadler. Proof systems for general statements about discrete logarithms. *Technical Report/ETH Zurich, Department of Computer Science*, 260, 1997.
14. 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.
15. 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.
16. Ran Canetti and Marc Fischlin. Universally composable commitments. In *CRYPTO 2001*, pages 19–40. Springer, 2001.
17. 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.
18. Ran Canetti and Tal Rabin. Universal composition with joint state. In *CRYPTO 2003*, pages 265–281. Springer, 2003.
19. Ran Canetti, Pratik Sarkar, and Xiao Wang. Triply adaptive uc nizk. *ePrint Archive*, 2020.
20. 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.
21. 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.
22. Elizabeth Crites, Chelsea Komlo, and Mary Maller. How to prove schnorr assuming schnorr: Security of multi-and threshold signatures. *ePrint Archive*, 2021.
23. Ivan Damgård. On σ -protocols. University of Aarhus, Department of Computer Science, 2002.
24. 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.
25. Chaya Ganesh, Yashvanth Kondi, Claudio Orlandi, Mahak Pancholi, Akira Takahashi, and Daniel Tschudi. Witness-succinct universally-composable snarks. In *Advances in Cryptology—EUROCRYPT 2023: 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23–27, 2023, Proceedings, Part II*, pages 315–346. Springer, 2023.

26. 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.
27. Yashvanth Kondi and abhi shelat. Improved straight-line extraction in the random oracle model with applications to signature aggregation. *ePrint Archive*, 2022.
28. Anna Lysyanskaya and Leah Namisa Rosenbloom. Efficient and universally composable non-interactive zero-knowledge proofs of knowledge with security against adaptive corruptions. *Cryptology ePrint Archive*, 2022.
29. Anna Lysyanskaya and Leah Namisa Rosenbloom. Universally composable sigma-protocols in the global random-oracle model. *ePrint Archive*, 2022.
30. 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.
31. Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In *CRYPTO 1992*, pages 31–53. Springer, 1992.
32. Somnath Panja and Bimal Kumar Roy. A secure end-to-end verifiable e-voting system using zero knowledge based blockchain. *ePrint Archive*, 2018.
33. Rafael Pass. On deniability in the common reference string and random oracle model. In *CRYPTO 2003*, pages 316–337, 2003.
34. Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of cryptology*, 4(3):161–174, 1991.
35. Berry Schoenmakers. A simple publicly verifiable secret sharing scheme and its application to electronic voting. In *CRYPTO 1999*, pages 148–164. Springer, 1999.
36. Douglas Wikström. A commitment-consistent proof of a shuffle. In Colin Boyd and Juan Manuel González Nieto, editors, *ACISP*, pages 407–421. Springer, 2009.