Resettable Non-Interactive Zero-Knowledge: Attacks and Defenses

Behzad Abdolmaleki[1], Matteo Campanelli[2], Quang Dao[3], Shadman Mohammadi[4], and Nahid Roustaeifar[1]

[1] University of Sheffield
{behzad.abdolmaleki, nroustaeifar1}@sheffield.ac.uk
[2] Offchain Labs and University of Tartu
binarywhalesinternaryseas@gmail.com
[3] Carnegie Mellon University
qvd@andrew.cmu.edu
[4] Independent Researcher
shadman.mohammadi1996@gmail.com

Abstract. As zero-knowledge proofs are used in real-world systems, they face threats such as resetting attacks, where an attacker manipulates the prover's randomness. While resettable security has been studied for interactive protocols, whether modern non-interactive systems such as zkSNARKs remain secure under resets is still unclear. This work provides the first systematic study of resettable security for NIZK arguments with three main contributions:

- **New Definition:** We define *strong resettable NIZK* (srNIZK), which captures adversaries that can selectively reset the prover's randomness, so that some parts are reused while others are generated afresh. This goes beyond the whole-tape resets considered in standard rZK and is motivated by fault injection on secure hardware and incomplete state restoration in virtualized environments.
- **Concrete Attacks:** We demonstrate that widely used NIZK constructions are vulnerable to resetting attacks. We show witness-recovery attacks against (i) Fiat–Shamir-compiled Σ-protocols (e.g., Schnorr), (ii) succinct proof systems using independently sampled masks (e.g., PLONK and Bulletproofs), and (iii) salted Fiat–Shamir compilations of a broad class of interactive arguments.
- **Generic Defense:** Our simple compiler transforms any NIZK into an srNIZK by modifying only randomness generation, using a pseudorandom function (PRF) secure under partial key resets. This prevents resetting attacks without changing proof size or verification.

Our results demonstrate that resetting attacks must be considered in NIZK systems, and provide a simple defense with minor prover overhead.

1 Introduction

Non-Interactive Zero-Knowledge (NIZK) proofs are a fundamental component in both the theory and practice of cryptography. A NIZK scheme enables a prover

to convince a verifier of a statement's validity through a single message, without revealing any information beyond the truth of the statement itself. NIZKs have numerous applications, including the construction of advanced cryptographic protocols [12,35] and core blockchain technologies [14].

As NIZKs see increasing deployment in real-world systems, they face novel attacks that are not covered by standard security definitions, such as knowledge soundness and zero-knowledge. A particularly concerning threat model emerges when adversaries can *manipulate* or *reset* the prover's randomness—a capability that is becoming increasingly realistic. Consider the following scenarios:

- *Trusted Execution Environments (TEEs)* offering proving-as-a-service, where a malicious server might compromise the TEE's input-output operations, including randomness generation.
- *Virtual machines* in cloud environments, where VM snapshots enable adversaries to reset the prover to a previous state with the same randomness [60].
- *Smartcards and hardware tokens*, which now run zero-knowledge provers [67] and where physical access enables attacks such as electromagnetic fault injection that can alter randomness [57].

The concept of *resettable zero-knowledge* (rZK) [21] was introduced for interactive protocols to address this threat, ensuring the zero-knowledge property holds even if a malicious verifier can reset the prover and force the reuse of randomness. While significant progress has been made in constructing resettable *interactive* zero-knowledge systems (see Section 1.3), the security of NIZK proofs[5] under resetting attacks remains less understood. Prior work studied reset attacks on interactive identification and faults in Fiat–Shamir signatures [2,10]. Here we examine zero-knowledge under resets for Fiat–Shamir NIZKs and succinct arguments such as PLONK [37] and Bulletproofs [19]. Thus, we first ask:

Q: Are existing NIZKs secure against resetting attacks?
If not, can we secure them generically with minimal overhead?

A Broader Class of Resetting Attacks. Additionally, we consider a new, practically motivated strengthening of resetting attacks. The standard model assumes an adversary can reset the prover's *entire* randomness tape. In contrast, realistic attack scenarios often grant the adversary more fine-grained control:

- **Partial Randomness Manipulation:** Fault injection attacks, such as voltage glitching or laser injection, can affect selected parts of a device's state while leaving others intact [6,57], and faults are known to compromise cryptographic computations [16]. This motivates a model in which an adversary can keep some portions of the prover's randomness fixed while others are freshly generated. Our model isolates resets of randomness; it does not cover leakage or faults in the subsequent computation.

[5] For simplicity we occasionally talk about "proofs" but we focus exclusively on computationally sound NIZKs [50], i.e., *arguments*.

- **State Restoration:** VM snapshots motivate randomness reuse [60]. Selective resets also model executions that combine restored prover state with fresh external entropy.

Therefore, a second question this work seeks to answer is:

Q: How to protect NIZKs against resetting adversaries that can manipulate even only part of the randomness?

1.1 Our Results

1) New Definition We introduce *strong resettable NIZK (srNIZK)*, a security notion that extends the whole-tape resetting capability of classical resettable zero-knowledge (rZK) [21]. While classical rZK allows an adversary to reset the prover's entire randomness tape, our model allows selective resets of arbitrary portions of the tape.

2) Attacks on Practical NIZKs in the ROM While our definitions are general, our attacks focus on NIZKs in the random oracle model (ROM), identifying vulnerabilities in both standard and strong resettable settings.

Standard Resetting Attacks. For Fiat–Shamir-compiled NIZKs, we give witness-recovering attacks on Maurer's proof of homomorphism preimage and show failure of zero-knowledge for homomorphisms with nontrivial kernels and images, including the Okamoto instance. We further show that PIOP-based schemes such as PLONK admit witness-recovering attacks with auxiliary input, under standard linear-response structure and deterministic commitments.

Strong Resetting Attacks. In the strong resetting model, salts allow an adversary to reproduce rewinding with fresh verifier challenges. This yields witness-recovering attacks on a broad class of Fiat–Shamir arguments, including Σ-protocols and multi-round arguments with polynomial-size challenge trees. As an illustration of the latter, the replay recovers both committed vectors from Bulletproofs' inner-product argument; that argument is not zero-knowledge on its own, so the example shows the extraction mechanism, not a privacy violation. We also give a general attack on PIOP-based schemes using extraction from their prover polynomials. For PLONK and Bulletproofs range provers with independently sampled masks, even salts are unnecessary: refreshing selected masks suffices to recover the witness. Unsalted Schnorr, whose only randomness is a single mask, is exposed as well once a reset can refresh one bit of that mask (Proposition 4.7).

3) Generic Compiler for NIZKs We propose a simple compiler that defends any NIZK against resetting attacks by modifying only the prover's randomness generation. The prover derives a short seed from $(\mathsf{pp}, \mathsf{x}, \mathsf{w})$ using one PRF evaluation and expands it into its randomness tape; setup, verification, and proof size

remain unchanged. The added prover work is this one PRF evaluation on the prover's inputs and one PRG expansion to the length of its tape (Section 5.2). Selective resets act on the seed before derivation; the derivation and subsequent prover computation execute correctly.

Ordinary PRFs give rNIZK in the standard model. For selective resets, we formalize *strong resettable-key PRF* security (sRK-PRF) and prove it equivalent to full XOR-related-key security [9, 11], up to polynomial overhead. We give an efficient ROM instantiation and a standard-model compiler conditional on this assumption.

1.2 Technical Overview

We use the Okamoto Σ-protocol [55] and its Fiat–Shamir non-interactive version as a running example to provide a high-level overview of our results.

1. **Public Parameters:** A group $\mathbb{G}$ of prime order p with generators g, h. The relation is $\mathcal{R}_{\mathsf{Oka}} = \{(\mathsf{x}, (\mathsf{w}_0, \mathsf{w}_1)) \mid \mathsf{x} = g^{\mathsf{w}_0} \cdot h^{\mathsf{w}_1}\}$.
2. **Prover $\mathcal{P}$:** Samples $r_0, r_1 \leftarrow \mathbb{F}_p$, computes the commitment $a = g^{r_0} \cdot h^{r_1}$, the challenge $c = \mathsf{H}(\mathsf{x}, a)$, and the responses $z_0 = r_0 + c \cdot \mathsf{w}_0$, $z_1 = r_1 + c \cdot \mathsf{w}_1$. The proof is $\pi = (a, z_0, z_1)$.
3. **Verifier $\mathcal{V}$:** Given x and $\pi = (a, z_0, z_1)$, recomputes $c = \mathsf{H}(\mathsf{x}, a)$ and checks $g^{z_0} \cdot h^{z_1} \stackrel{?}{=} a \cdot \mathsf{x}^c$.

Resetting vs. Rewinding: Rewinding allows an extractor to return to a fixed protocol state and obtain two accepting transcripts with the same (x, a) but distinct challenges $c \neq c'$, from which it can recover the witness. In contrast, resetting the whole prover tape on fixed inputs reproduces the same challenge, since $c = \mathsf{H}(\mathsf{x}, a)$ is deterministic. Thus, whole-tape resetting on fixed inputs does not directly emulate rewinding.

The Resetting Threat to NIZKs: Although NIZKs are naturally secure against concurrent attacks [36], resetting allows an adversary to reuse all or part of the randomness across proofs, breaking independence between executions.

Breaking Zero-Knowledge: Reusing randomness across statements preserves the first message a and correlates the responses. In Okamoto, these correlations reveal the particular witness used by the prover, as we show below. Given x, a simulator must account for p equally likely witness pairs without knowing which pair the prover used. This yields a distinguishing attack even when the simulator knows the reset pattern; Section 4.1.1 gives the formal argument.

Witness Recovery: Resetting can also enable direct witness recovery. With an auxiliary pair $(\mathsf{x}', \mathsf{w}')$, the adversary obtains a proof, uses the known w' to recover the prover's randomness $r = (r_0, r_1)$, and then resets the prover to reuse r on a target statement x. The responses reveal the target witness w. We extend this auxiliary-input attack to PIOP-based NARGs in Section 4.1.2.

Under strong resetting, auxiliary inputs are unnecessary. By selectively fixing r_0 while varying r_1, the adversary extracts w_0; similarly, fixing r_1 while varying r_0 extracts w_1. Only three proofs are needed: one fresh proof and one reset for

each mask block (Lemma A.4). Combining both parts recovers $\mathsf{w} = (\mathsf{w}_0, \mathsf{w}_1)$. We give strong resetting attacks on PIOP-based NARGs in Section 4.2.2.

Salted Fiat–Shamir provides another route. If $c = \mathsf{H}(\mathsf{x}, a; \tau)$ and τ is part of the randomness, a strong resetting adversary can fix $r = (r_0, r_1)$ while varying τ. This gives fresh challenges for the same (x, a), so two different challenges reveal both witness coordinates. Section 4.2 extends this approach to multi-round protocols.

Generic Compiler: Our compiler prevents these attacks by enforcing two properties on the prover's randomness. First, the derived tapes remain pseudorandom even when the adversary resets only part of the seed. Second, the tape is bound to the instance and witness, so reusing a seed across different inputs produces computationally independent tapes. These properties allow us to reduce strong resettable security to the zero-knowledge of the underlying proof system.

We first derive a short seed from $(\mathsf{pp}, \mathsf{x}, \mathsf{w})$ using F_r, then expand it into the prover's random tape. A standard PRF suffices for rNIZK. For srNIZK we use an sRK-PRF, which remains pseudorandom under partial key fixing. Details appear in Section 5.

1.3 Related Work

Resettable Zero-Knowledge in the Interactive Setting. The notions of *concurrent zero-knowledge* [36] and *resettable zero-knowledge* [21], our primary focus, were proposed for interactive protocols. Concurrent Zero-Knowledge (cZK) guarantees that the zero-knowledge property holds even when multiple protocol sessions are arbitrarily interleaved.

Canetti et al. [21] also proposed a transformation that converts an *admissible proof system*—zero-knowledge in a hybrid model—into a resettable zero-knowledge system. Since then, there has been significant progress in constructing proof systems with this type of security. Barak et al. [7] introduced resettable soundness and constructed resettably-sound zero-knowledge arguments. Building on this, Deng, Goyal, and Sahai [31] resolved the simultaneous resettability conjecture by constructing an argument for NP that is both resettable zero-knowledge and resettably sound. Bellare et al. [10] studied reset attacks on interactive identification, using verifier challenge commitments and PRF-derived prover coins to prevent impersonation.

A substantial body of literature explores resettable security in the relaxed *bare public key* model and related public-key models, which assume all verifiers deposit a public key in a public file before any interaction occurs [7, 21, 24, 30, 32, 33, 51, 52, 74]. Other works have proposed resettable *statistical* zero-knowledge [39, 44], in which the view of every polynomial-time resetting verifier is simulated up to statistical distance, so the simulation withstands unbounded distinguishers even when the prover reuses randomness.

However, these efforts target interactive protocols. Closer to our setting, zaps, which are two-message protocols with a reusable first message, achieve resettable witness indistinguishability when the prover derives its coins with a PRF [35],

and hedged Fiat–Shamir signatures have been analyzed under adversarial nonce inputs and faults [2].

Deterministic, Leaked, and Correlated Prover Randomness. Several lines of work study zero knowledge when the prover's coins are absent, exposed, or correlated. In the plain model, deterministic-prover zero knowledge has been obtained for honest verifiers and for verifiers with bounded auxiliary input [15, 26]. In the common reference string model, Sahai and Waters [62] construct a NIZK with a deterministic prover from indistinguishability obfuscation. A prover without coins is unaffected by resets; their result, however, is a new proof system, whereas our compiler keeps an existing NIZK and changes only how it derives its coins. Garg, Jain, and Sahai [38] define leakage-resilient NIZK, in which the adversary obtains leakage on the witness and on the prover's coins, and the simulator obtains leakage on the witness. Austrin et al. [5] show that zero-knowledge protocols can be broken with probability p by an adversary that tampers online with each bit of an honest party's tape with probability p. A resetting adversary cannot choose tape bits: each bit is either copied from an earlier tape or sampled afresh.

Closest to our model is correlated-tape zero knowledge, introduced by Goyal and Song [41]. In their model, the adversary fixes tampering functions $f_1, \ldots, f_t$ before the executions. The prover holds one uniform tape X and runs the j-th execution on the tape $f_j(X)$, which must have sufficient min-entropy; resettable zero knowledge is the case of equal tapes. Their protocols apply a seeded correlated-source extractor to the tape and then run a protocol that is secure under resets. The seed is part of a uniform reference string that the simulator cannot program, and they show that the tampering functions must not depend on this string. Our resets are a different kind of correlation: each tape copies chosen positions of an earlier tape and is fresh elsewhere, and the adversary chooses every request adaptively, after seeing the public parameters and all earlier proofs. We also treat non-interactive arguments whose setup, proofs, and verifier the defense leaves unchanged, and we give attacks on deployed proof systems. To our knowledge, none of these works defines or analyzes zero knowledge of Fiat–Shamir NIZKs and SNARKs under resets of the prover's randomness, whether of the whole tape or of selected positions.

Security Against Malicious Provers. Simulation soundness [61] and simulation extractability [29] preserve soundness and knowledge soundness after a malicious prover obtains simulated proofs. Our focus is the privacy of honest proofs when the prover's randomness is reset.

Hedging Cryptography Against Randomness Failures. Since randomness failures are widespread and unlikely to be completely eliminated, a fundamental approach is to design cryptographic primitives that hedge against such failures. A simple and elegant method for hedged public-key encryption (PKE) is the *Randomized-Encrypt-With-Hash* (REwH) technique introduced by Bellare et al. [8]. This approach hashes the encryption randomness together with the public

key and the message, and uses the hash output as the coins of the underlying PKE scheme. While their goal of mitigating bad randomness aligns with ours, their definitions do not capture the full adversarial control of the resettable setting.

In subsequent work aligned with our goal, Yilek [73] addressed the case of reused (but not adversarially chosen) randomness, suggesting a construction where the hash is replaced by a keyed pseudorandom function (PRF). Building on this, Ristenpart and Yilek [60] proposed a backwards-compatible framework for hedging deployed cryptography, protecting primitives like PKE, symmetric encryption, digital signatures, and key exchange against a wide range of randomness failures. Similar techniques were later applied to authenticated key exchange (AKE) protocols [72].

Motivated by the history of randomness failures, Paterson, Schuldt, and Sibborn [58] discuss hedging in the context of PKE to protect against *related randomness attacks*, where the randomness used after a reset is correlated with previously used values.

Bellare and Tackmann [13] develop nonce-based public-key encryption and signatures using hedged extractors, which derive coins from a secret seed, a message, and a nonce. Deterministic signing provides another defense: RFC 6979 and EdDSA derive signing nonces from secret key material and the message [42, 59]. BIP-340 additionally mixes auxiliary randomness into this derivation [70]. MuSig-DN [54] derives multi-signature nonces with a PRF and has each signer prove the derivation in zero knowledge, since naively deterministic nonces let an adversary obtain two partial signatures under the same nonce; its authors note that deterministic signing also protects against virtual-machine rewinding. Breitner and Heninger [17] demonstrate the practical consequences of weak nonces by recovering ECDSA private keys from cryptocurrency signatures and Internet-wide scans. These works motivate deriving coins from secret inputs, while addressing encryption or signature security rather than zero-knowledge under selective resets. Unlike Aranha et al. [2], we assume that computation after the seed reset executes correctly. Faults in the computation itself have been used to recover signing keys of zero-knowledge-based signatures [53] and, through Rowhammer bit flips, to leak witness bits from a zk-SNARK prover [47]; such faults are outside our model.

Our compiler follows the same general approach of deriving randomness from the prover's inputs and a random seed. Deriving prover coins from a PRF of the input is also the classical route to resettability [21,35]; our compiler applies it to NIZKs. We study the security of this approach when an adversary can preserve selected seed coordinates across executions while the remaining coordinates are freshly sampled.

Transcript-Based Randomness in Merlin. Merlin introduced a closely related defense against bad prover randomness in 2018 [69]. Its transcript RNG [68] combines a private copy of the public transcript state, the prover's secret witness data, and entropy from an external RNG to derive prover randomness. Its documentation already warns that using the same blinding factor in two proofs

reveals the secret. Thus, binding randomness to the proof context, witness, and external entropy is an existing practical design. Our focus is to formulate and analyze security under an explicit selective-reset interface, and our compiler analyzes input-derived randomness for NIZKs under adaptive selective seed resets. Establishing that Merlin itself satisfies our definition would require a separate analysis of its concrete construction and the placement of resets; we do not claim that result here.

Missing Blinding in PLONK. Sefranek [64] shows that the original PLONK specification is not statistically witness indistinguishable and proves statistical zero knowledge for the patched protocol, which is the version we analyze. Implementations have also omitted blinders: an issue in Dusk's PLONK library [34] and an audit of Linea's prover [56] each report prover polynomials that were committed without the specified blinding, and a public challenge recovers a witness from a PLONK proof generated without blinding [46]. These reports concern masks that are absent. Our attacks apply to provers that sample every mask as specified, and exploit masks that repeat across proofs.

Related-Key Pseudorandomness. Bellare and Kohno [11] formalized related-key security; Bellare and Cash [9] studied PRFs for algebraic key relations. Our strong resettable-key notion captures coordinate copying and resampling. We show that it is equivalent to security under the full family of XOR key offsets, up to polynomial overhead (Section 5.3).

2 Preliminaries

We fix notation and random-oracle conventions, then recall interactive arguments, non-interactive arguments and their zero-knowledge experiments, and the Fiat–Shamir transformation connecting the two. We then introduce the polynomial IOPs, polynomial commitments, and PIOP compilation used in our attacks.

Let $\mathbb{N}$ denote the set of positive integers. For a list L, we write $L[i]$ for its i-th element (indexing starting at 1), and $|L|$ for its length; each element is an ℓ-bit string. For an ℓ-bit string r, we write $r[i]$ for its i-th bit.

In the ROM, all parties share the public oracle $\mathsf{H}_{\mathsf{proof}}$. In ideal experiments, its query handler and proof simulator share state: answered inputs are fixed, while fresh inputs are uniform unless programmed. The compiler's derivation oracle $\mathsf{H}_{\mathsf{rnd}}$ is independent and unprogrammed. We suppress fixed public parameters and domain tags.

2.1 Interactive Arguments

Definition 2.1 (Interactive Argument). *An interactive argument (*IARG*) for a relation $\mathcal{R}$ is a tuple of PPT algorithms $\Pi = (\mathcal{G}, \mathcal{P}, \mathcal{V})$ with the following syntax:*

- $\mathcal{G}(1^\lambda) \to \mathsf{pp}$: *outputs public parameters* pp *given security parameter* 1^λ.

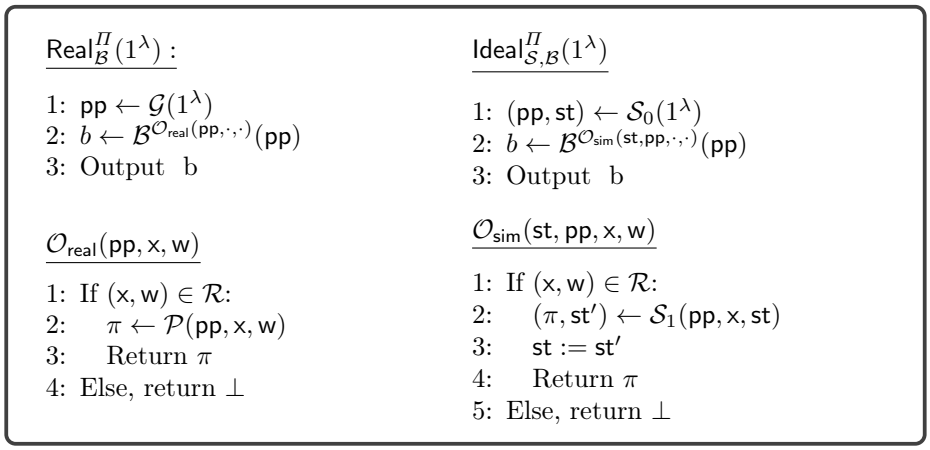


Fig. 1: Experiments for NIZK.

- $\langle \mathcal{P}(\mathsf{w}), \mathcal{V} \rangle(\mathsf{pp},\mathsf{x}) \to \{0,1\}$: *an interactive protocol whereby the prover $\mathcal{P}$, holding a witness w, interacts with the verifier $\mathcal{V}$ on common input (pp,x) to convince $\mathcal{V}$ that $(\mathsf{pp},\mathsf{x},\mathsf{w}) \in \mathcal{R}$. At the end, $\mathcal{V}$ outputs a bit to accept or reject the proof.*

We define the following properties for interactive arguments:

- **Completeness.** For any adversary $\mathcal{A}$:

$$\Pr\left[\begin{matrix}(\mathsf{pp},\mathsf{x},\mathsf{w}) \notin \mathcal{R} \vee \\ \langle \mathcal{P}(\mathsf{w}), \mathcal{V} \rangle(\mathsf{pp},\mathsf{x}) = 1\end{matrix} \;\middle|\; \begin{matrix}\mathsf{pp} \leftarrow \mathcal{G}(1^\lambda) \\ (\mathsf{x},\mathsf{w}) \leftarrow \mathcal{A}(\mathsf{pp})\end{matrix}\right] = 1.$$

- **Knowledge Soundness.** There exists an expected-polynomial-time extractor $\mathcal{E}$ such that for any stateful PPT adversary $\mathcal{P}^*$, the following probability is negligible in λ:[6]

$$\Pr\left[\begin{matrix}\\ b = 1 \wedge \\ (\mathsf{pp},\mathsf{x},\mathsf{w}) \notin \mathcal{R}\end{matrix} : \begin{matrix}\mathsf{pp} \leftarrow \mathcal{G}(1^\lambda) \\ (\mathsf{x},\mathsf{st}_{\mathcal{P}^*}) \leftarrow \mathcal{P}^*(\mathsf{pp}) \\ b \leftarrow \langle \mathcal{P}^*(\mathsf{st}_{\mathcal{P}^*}), \mathcal{V} \rangle(\mathsf{pp},\mathsf{x}) \\ \mathsf{w} \leftarrow \mathcal{E}^{\mathcal{P}^*}(\mathsf{pp},\mathsf{x})\end{matrix}\right]$$

  Here $\mathcal{E}$ gets black-box access to each of the next-message functions of $\mathcal{P}^*$ in the interactive protocol, and can rewind $\mathcal{P}^*$ to any point in the interaction.
- **Public Coin.** In each round i, the verifier $\mathcal{V}$ samples its message uniformly at random from some challenge space Ch_i.

[6] We do not define soundness, but this implies soundness with the same advantage.

2.2 Non-Interactive Arguments

Definition 2.2 (Non-Interactive Argument Syntax). *The syntax of a* NARG $\Pi = (\mathcal{G}, \mathcal{P}, \mathcal{V})$ *for an NP relation* $\mathcal{R}$ *consists of three PPT algorithms:*

- $\mathsf{pp} \leftarrow \mathcal{G}(1^\lambda)$*: outputs public parameters.*
- $\pi \leftarrow \mathcal{P}(\mathsf{pp}, \mathsf{x}, \mathsf{w})$*: on input* $(\mathsf{x}, \mathsf{w}) \in \mathcal{R}$*, outputs a proof* π*.*
- $b \leftarrow \mathcal{V}(\mathsf{pp}, \mathsf{x}, \pi)$*: deterministically outputs* $b \in \{0, 1\}$*.*

We include any circuit index in the public statement x. When setup determines the relation, $(\mathsf{x}, \mathsf{w}) \in \mathcal{R}$ means $(\mathsf{x}, \mathsf{w}) \in \mathcal{R}_{\mathsf{pp}}$. Externally fixed parameters are common immutable inputs to all algorithms, including the setup simulator; we suppress them in the notation.

We state completeness and soundness assumptions where needed. Our focus is adaptive multi-theorem zero-knowledge.

Definition 2.3 (Non-Interactive Zero-Knowledge (NIZK)). *Let* $\Pi = (\mathcal{G}, \mathsf{P}, \mathsf{V})$ *be a* NARG *for an NP relation* $\mathcal{R}$*. We say* Π *is a NIZK if it satisfies* computational adaptive multi-theorem zero-knowledge*. Specifically, there exists a PPT simulator* $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1)$ *such that for any PPT adversary* $\mathcal{B}$*:*

$$\mathsf{Adv}^{\mathit{NIZK}}_{\mathcal{B},\Pi}(\lambda) := \left| \Pr\left[\mathsf{Real}^{\Pi}_{\mathcal{B}}(1^\lambda) = 1\right] - \Pr\left[\mathsf{Ideal}^{\Pi}_{\mathcal{S},\mathcal{B}}(1^\lambda) = 1\right] \right| \leq \mathrm{negl}(\lambda)$$

In both experiments, $\mathcal{B}$ *receives public parameters and makes polynomially many adaptive proof queries* (x, w)*; invalid pairs are rejected. The real experiment uses honest setup and returns* $\mathcal{P}(\mathsf{pp}, \mathsf{x}, \mathsf{w})$ *on each valid query. The ideal experiment generates* $(\mathsf{pp}, \mathsf{st}) \leftarrow \mathcal{S}_0(1^\lambda)$ *and answers each valid query using* $(\pi, \mathsf{st}') \leftarrow \mathcal{S}_1(\mathsf{pp}, \mathsf{x}, \mathsf{st})$*, updating its state without revealing* w *to the simulator. Both experiments output* $\mathcal{B}$*'s bit. Figure 1 gives the pseudocode.*

2.3 Fiat–Shamir Transformation

We use the Fiat–Shamir transformation with the full transcript [23]. Each challenge is derived from the public context and all preceding messages. The salted variant also includes independent public salts, which become part of the transcript.

Definition 2.4 ((Salted) Fiat–Shamir Transformation). *Let* IARG *be a public-coin interactive argument* $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ *with* R *verifier challenges and transcript* $(a_1, c_1, \ldots, a_R, c_R, a_{R+1})$*. We say that* $\mathsf{NARG} := \mathsf{FS}^*_{\mathit{salt}}[\mathsf{IARG}, \lambda]$ *is the non-interactive argument in the ROM obtained from* IARG *via the* (salted) Fiat–Shamir transformation *if* NARG *is defined as follows. All hash inputs use injective encodings and a reserved domain tag* $\mathsf{tag}_{\mathrm{FS}}$*;* H_i *denotes the round-*i *oracle domain with uniform output in that round's challenge space:*

- $\mathsf{pp} \leftarrow \mathsf{NARG}.\mathcal{G}(1^\lambda)$*: Return* $\mathsf{pp} \leftarrow \mathsf{IARG}.\mathcal{G}(1^\lambda)$*.*
- $\pi \leftarrow \mathsf{NARG}.\mathcal{P}^{\mathsf{H}}(\mathsf{pp}, \mathsf{x}, \mathsf{w})$*:*
  1. *For* $i \in [R]$*:*

(a) *Compute the i-th message a_i from the prover state and the preceding challenges.*
(b) *Sample a random salt $\tau_i \in \{0,1\}^\lambda$.*
(c) *Let $T_{i-1} = ((a_j, c_j, \tau_j))_{j<i}$ and derive*

$$c_i := \mathsf{H}_i(\mathsf{enc}(\mathsf{tag}_{\mathrm{FS}}, \mathsf{pp}, \mathsf{x}, T_{i-1}, a_i, \tau_i)) .$$

2. *Compute the final response a_{R+1} using c_R and the prover state.*
3. *Output the proof $\pi := \big((a_i)_{i\in[R+1]}, (\tau_i)_{i\in[R]}\big)$.*

- $b \leftarrow \mathsf{NARG}.\mathcal{V}^{\mathsf{H}}(\mathsf{pp}, \mathsf{x}, \pi)$*:*
  1. *Parse the proof $\pi = \big((a_i)_{i\in[R+1]}, (\tau_i)_{i\in[R]}\big)$.*
  2. *For each round $i \in [R]$, derive the challenge c_i exactly as in Step 1(c) of the prover.*
  3. *Return $b \leftarrow \mathsf{IARG}.\mathcal{V}\big(\mathsf{pp}, \mathsf{x}, (a_i)_{i\in[R+1]}, (c_i)_{i\in[R]}\big)$.*

Clearly, if the parts highlighted in orange are not executed, the transformation corresponds to the standard (non-salted) Fiat–Shamir.

Lemma 2.5 (Completeness of Fiat–Shamir). *If the underlying public-coin argument has perfect completeness, its salted and unsalted Fiat–Shamir compilations have perfect completeness.*

Proof. Perfect completeness holds for every honest prover tape and every verifier challenge sequence in the support of the finite uniform challenge distributions. The compiled prover computes the same messages for the challenges supplied by the random oracle, including the final response. The compiled verifier reconstructs those challenges and therefore accepts.

2.4 Polynomial Interactive Oracle Proofs

For the attacks on polynomial-based arguments, we need to specify which polynomials the prover sends and which evaluations the verifier obtains. We use polynomial interactive oracle proofs (PIOPs) with preprocessing, following [45], which builds on [20, 22].

Definition 2.6 (PIOP). *A Polynomial Interactive Oracle Proof (*PIOP*) for an indexed relation $\mathcal{R}_{\mathsf{i}}$ is a tuple of algorithms $\mathsf{PIOP} = (\mathbf{I}, \mathbf{P}, \mathbf{V})$. The protocol execution proceeds as follows:*

- ***Offline phase.*** *Given an index i for $\mathcal{R}_{\mathsf{i}}$, the indexer $\mathbf{I}$ outputs $\mathsf{s}(0)$ preprocessed polynomials $p_{0,1}, \ldots, p_{0,\mathsf{s}(0)} \in \mathbb{F}^{\le d(|\mathsf{i}|)}[X]$.*
- ***Online phase.*** *Given an instance x and witness w such that $(\mathsf{i}, \mathsf{x}, \mathsf{w}) \in \mathcal{R}_{\mathsf{i}}$, the prover $\mathbf{P}$ receives $(\mathsf{i}, \mathsf{x}, \mathsf{w})$, and the verifier $\mathbf{V}$ receives x and has oracle access to the polynomials output by $\mathbf{I}(\mathsf{i})$. The parties engage in a R-round interaction. In round r, the prover $\mathbf{P}$ sends $\mathsf{s}(r)$ oracle polynomials $p_{r,1}, \ldots, p_{r,\mathsf{s}(r)} \in \mathbb{F}^{\le d(|\mathsf{i}|)}[X]$ to the verifier $\mathbf{V}$, who then replies with a challenge $c_r \in \mathsf{Ch}$. The protocol is* public-coin*, meaning all challenges c_r are public and uniformly sampled from the challenge space Ch.*

Query phase. *Let* $\mathbf{p} = (p_{r,t})_{r\in\{0\}\cup[R],t\in[\mathsf{s}(r)]}$ *denote the preprocessed and prover polynomials. On input* $(\mathsf{x}; c_1, \ldots, c_R)$*, the verifier's query subroutine* $\mathbf{Q_v}$ *outputs evaluation points* $\mathbf{z} = (z_{r,t,u})_{r\in\{0\}\cup[R],\, t\in[\mathsf{s}(r)],\, u\in[\mathsf{s}(r,t)]}$*. Here* $\mathsf{s}(r,t)$ *is the number of points at which* $p_{r,t}$ *is queried. The verifier obtains the corresponding evaluation vector* $\mathbf{y} = \mathbf{p}(\mathbf{z})$*.*

- ***Decision phase.*** *The verifier executes a decision subroutine* $\mathbf{D_V}$ *that receives* $(\mathsf{x}, \mathbf{p}(\mathbf{z}); c_1, \ldots, c_R)$ *as input and outputs a decision bit* b *indicating acceptance or rejection.*

Remark 2.7 (PIOP with Multilinear Polynomials). Definition 2.6 assumes that the polynomials sent are univariate. There is a growing body of work on SNARKs built from PIOPs with *multilinear* polynomials [3,40,65,66,71]. To accommodate this, we simply need to replace "univariate" with "multilinear" or "multivariate" in our definition above, with the degree restrictions extended properly.

We require PIOP to satisfy the following properties:

Definition 2.8 (Completeness). *A* PIOP *is complete if for any* $(\mathsf{i}, \mathsf{x}, \mathsf{w}) \in \mathcal{R}_\mathsf{i}$*,*

$$\Pr\left[b \leftarrow \langle \mathbf{P}(\mathsf{i}, \mathsf{x}, \mathsf{w}), \mathbf{V}^{\mathbf{I}(\mathsf{i})}(\mathsf{x})\rangle : b = 1\right] = 1.$$

Definition 2.9 (Knowledge Extraction from Prover Polynomials). *We use the following coefficient-oracle knowledge guarantee. There exists a strict-PPT black-box extractor* E *such that, for every admissible PPT prover* $\mathbf{P}^*$ *and every PPT sampler of its index, statement, and auxiliary state, the probability of acceptance together with extraction failure is negligible:*

$$\Pr\left[b = 1 \ \wedge \ (\mathsf{i}, \mathsf{x}, \mathrm{E}^{\mathbf{P}^*}(\mathsf{i}, \mathsf{x})) \notin \mathcal{R}_\mathsf{i}\right] \le \mathsf{negl}(\lambda).$$

The sampler chooses the index before preprocessing, then chooses the statement and auxiliary state from the resulting public context. Here b *is the decision in a reference interaction with the honest verifier. An admissible prover sends only polynomials within the declared degree bounds [45], including any smaller bound that the protocol prescribes for an individual polynomial. The extractor receives next-message access to the complete polynomial coefficient vectors, may choose any challenge in the prescribed spaces, and may rewind the prover to a previous state. One sampled prover initialization, including its tape and auxiliary state, is cloned into the reference interaction and the extractor's rewinding interaction. Their subsequent mutable states are separate. The extractor is not given the reference transcript or the auxiliary state. Sampled preprocessing is fixed throughout, and its coefficient vectors are available to the extractor.*

This guarantee asks more than extraction from an accepting transcript, because the extractor does not see the reference challenges. It nevertheless holds for every PIOP in which the witness can be read from the first prover message.

Lemma 2.10 (First-message decoders). *Let* Dec *be a deterministic algorithm that runs in polynomial time and maps the index, the statement, the preprocessing coefficients, and the coefficient vectors* $\mathbf{p}_1$ *of the first prover message*

to a candidate witness. Suppose that, for every admissible PPT prover and every PPT sampler as in Definition 2.9,

$$\Pr\big[b = 1 \ \wedge \ (\mathsf{i}, \mathsf{x}, \mathsf{Dec}(\mathsf{i}, \mathsf{x}, \mathbf{p}_0, \mathbf{p}_1)) \notin \mathcal{R}_{\mathsf{i}}\big] \leq \mathsf{negl}(\lambda),$$

where b and $\mathbf{p}_1$ are the decision and the first prover message of the reference interaction. Then the PIOP satisfies Definition 2.9.

Proof. The extractor requests the first prover message and applies Dec to it. The prover sends this message before it receives any challenge, so the message is determined by the prover's initialization, which the reference interaction and the extractor's interaction share. The extractor therefore decodes the same $\mathbf{p}_1$ that appears in the reference interaction, and its failure event is the one bounded in the hypothesis. It makes one oracle call and runs in strict polynomial time.

The hypothesis of Lemma 2.10 is what the usual soundness analysis of such a PIOP proves: once the first message is fixed and decodes to an invalid witness, every continuation is accepted with negligible probability. In PLONK [37], Dec evaluates the three wire polynomials on the evaluation domain. If the resulting assignment violates a gate or a copy constraint, then for arbitrary wire polynomials and later messages of degree at most D, the gate, permutation, and quotient checks accept with probability $O(D/|\mathbb{F}|)$ over the permutation, quotient-combination, and evaluation challenges. The R1CS protocol of Marlin [22] and the basic protocol of Sonic [48] also place the witness in the first prover polynomial. Their soundness analyses use a separate degree bound for each polynomial and, for Sonic, a prescribed coefficient support, so the lemma applies to them only when admissibility includes these restrictions. When the witness is determined only by later prover messages, Definition 2.9 needs a separate argument; decoding the accepting transcripts of the honest prover does not suffice.

2.5 Polynomial Commitments

To compile a PIOP into an argument, we replace polynomial oracles by commitments and proofs of evaluation. The commitment randomness matters for resetting attacks, so we include it explicitly in the syntax.

Definition 2.11 (Polynomial Commitment Scheme). *A polynomial commitment scheme* PCOM *consists of four algorithms,* $(\mathsf{KGen}, \mathsf{Com}, \mathsf{Eval}, \mathsf{Check})$*:*

1. $\mathsf{ck} \leftarrow \mathsf{KGen}(1^\lambda, D)$*: It takes as input the security parameter λ and the maximum degree bound D and generates commitment key ck as output. We assume ck to include a description of the finite field $\mathbb{F}$.*
2. $C \leftarrow \mathsf{Com}(\mathsf{ck}, f; \rho)$*: It takes as input ck, the polynomial $f \in \mathbb{F}^{\leq D}[X]$, and using randomness ρ, outputs a commitment C. In case the commitment scheme is deterministic, then $\rho = \perp$.*

1. $\mathcal{G}(1^\lambda)$: Output $\mathsf{srs} \leftarrow \mathsf{PCOM.KGen}(1^\lambda, D_\lambda)$.
2. $\mathcal{I}(\mathsf{i}, \mathsf{srs})$: Sample $\mathbf{p}_0 = (p_{0,t})_{t\in[\mathsf{s}(0)]} \leftarrow \mathsf{PIOP.I}(\mathsf{i})$. Independently sample commitment coins $\rho_{0,t}$ and set $C_{0,t} \leftarrow \mathsf{PCOM.Com}(\mathsf{srs}, p_{0,t}; \rho_{0,t})$. Output $\mathsf{ipk} = (\mathsf{i}, \mathsf{srs}, \mathbf{C}_0, \mathbf{p}_0, \boldsymbol{\rho}_0)$ and $\mathsf{ivk} = (\mathsf{i}, \mathsf{srs}, \mathbf{C}_0)$.
3. $\mathcal{P}^{\mathsf{H}}(\mathsf{ipk}, \mathsf{x}, \mathsf{w})$:
   (a) Set $\mathsf{st}_0 = (\mathsf{i}, \mathsf{x}, \mathsf{w})$, $c_0 = \perp$, and retain $\mathbf{p}_0, \boldsymbol{\rho}_0, \mathbf{C}_0$.
   (b) For $r = 1, \ldots, R$, compute $(\mathbf{p}_r, \mathsf{st}_r) \leftarrow \mathsf{PIOP.P}(\mathsf{st}_{r-1}, c_{r-1}; \sigma_r)$. For each $t \in [\mathsf{s}(r)]$, independently sample $\rho_{r,t}$ and set $C_{r,t} \leftarrow \mathsf{PCOM.Com}(\mathsf{srs}, p_{r,t}; \rho_{r,t})$. Then set $c_r := \mathsf{Ch}_r(\mathsf{srs}, \mathsf{i}, \mathsf{x}, \mathbf{C}_{\le r}, \mathbf{c}_{<r})$ and pass it to the next PIOP transition.
   (c) Compute $\mathbf{z} \leftarrow \mathsf{PIOP.Q}_{\mathbf{v}}(\mathsf{x}; c_1, \ldots, c_R)$. For every (r, t, u) with $r \in \{0\} \cup [R]$, $t \in [\mathsf{s}(r)]$, and $u \in [\mathsf{s}(r, t)]$, set $y_{r,t,u} := p_{r,t}(z_{r,t,u})$ and $\pi_{r,t,u} \leftarrow \mathsf{PCOM.Eval}(\mathsf{srs}, C_{r,t}, z_{r,t,u}, p_{r,t}; \rho_{r,t})$. Output $\Pi = (\mathbf{C}_1, \ldots, \mathbf{C}_R, \mathbf{y}, (\pi_{r,t,u})_{r,t,u})$.
4. $\mathcal{V}^{\mathsf{H}}(\mathsf{ivk}, \mathsf{x}, \Pi)$:
   (a) Parse $\Pi = (\mathbf{C}_1, \ldots, \mathbf{C}_R, \mathbf{y}, \boldsymbol{\pi})$ with exactly the prescribed commitment, evaluation, and proof indices; reject malformed encodings. Set $\mathbf{C}_0$ from ivk.
   (b) For $r = 1, \ldots, R$, compute $c_r := \mathsf{Ch}_r(\mathsf{srs}, \mathsf{i}, \mathsf{x}, \mathbf{C}_{\le r}, \mathbf{c}_{<r})$.
   (c) Compute $\mathbf{z} \leftarrow \mathsf{PIOP.Q}_{\mathbf{v}}(\mathsf{x}; c_1, \ldots, c_R)$. Accept iff $\mathbf{D}_{\mathbf{V}}(\mathsf{x}, \mathbf{y}; c_1, \ldots, c_R) = 1$ and, for every (r, t, u) in the prescribed index set, $\mathsf{PCOM.Check}(\mathsf{srs}, C_{r,t}, z_{r,t,u}, y_{r,t,u}, \pi_{r,t,u}) = 1$.

Fig. 2: Fiat–Shamir compilation of a PIOP using a polynomial commitment scheme. Each commitment's coins $\rho_{r,t}$ are retained and reused for every opening of that commitment. The salted compilation, defined in Section 2.6, adds a salt to every challenge derivation.

3. $\pi \leftarrow \mathsf{Eval}(\mathsf{ck}, C, \mathbf{z}, f; \rho)$*: It takes as input* ck*, the commitment* C*, evaluation point* $\mathbf{z} \in \mathbb{F}$*, the polynomial* f*, and outputs a non-interactive proof of evaluation* π*. The randomness* ρ *must equal the one previously used in* Com*.*
4. $b \leftarrow \mathsf{Check}(\mathsf{ck}, C, \mathbf{z}, y, \pi)$*: It takes as input the statement* $(\mathsf{ck}, C, \mathbf{z}, y)$*, where* $y \in \mathbb{F}$ *is a claimed polynomial evaluation, and the proof of evaluation* π*, and outputs a bit* b*.*

Typically, PCOM should satisfy *correctness*, *extractability*, and *hiding*. For formal definitions, we refer to, e.g., [22, 43]. Additionally, milder properties such as *weak hiding* and *weak evaluation binding* are discussed in [45].

2.6 Compiling Polynomial IOPs

We give the Fiat–Shamir compilation of a PIOP using a polynomial commitment scheme PCOM. For each security parameter λ, let $\mathcal{I}_\lambda$ be the set of supported

indices and set

$$D_\lambda = \max_{\mathsf{i} \in \mathcal{I}_\lambda} d(|\mathsf{i}|).$$

We assume this universal degree bound is efficiently computable and polynomial in λ. The PIOP indexer may be randomized; its sampled preprocessing polynomials are fixed once by $\mathcal{I}$ and stored in the proving key. For the reset games, fix the public index i before indexing, generate its key pair once, and set $\mathsf{pp} = (\mathsf{ipk}, \mathsf{ivk})$. Both keys are public; the prover and verifier use their respective components. The target sampler then chooses a valid statement and witness for this indexed relation. This convention also applies to PPT ensembles of indices chosen before indexing, and gives the attacker the same preprocessing coefficients as the PIOP extractor.

For $r \in [R]$, let $\mathbf{C}_r = (C_{r,t})_{t \in [\mathsf{s}(r)]}$, and let $\mathbf{C}_0$ denote the commitments to the preprocessing polynomials. Write $\mathbf{C}_{\le r} = (\mathbf{C}_0, \ldots, \mathbf{C}_r)$ and $\mathbf{c}_{<r} = (c_1, \ldots, c_{r-1})$. Both algorithms use the same challenge routine

$$\mathsf{Ch}_r(\mathsf{srs}, \mathsf{i}, \mathsf{x}, \mathbf{C}_{\le r}, \mathbf{c}_{<r}) := \mathsf{H}(\mathsf{enc}(\mathsf{tag}_{\mathrm{ch}}, \mathsf{srs}, \mathsf{i}, \mathsf{x}, r, \mathbf{C}_{\le r}, \mathbf{c}_{<r}))\,,$$

where $\mathsf{tag}_{\mathrm{ch}}$ is a reserved domain tag, enc is an injective, length-delimited encoding and H maps to Ch. The round number is included in the encoding, so the honest prover's challenge queries are distinct. The PIOP prover is run as a stateful sequence of transitions: on round r, it receives c_{r-1} and its current state, with $c_0 = \bot$. Each transition uses fresh independent coins σ_r, retaining any earlier randomness in its state. For deterministic commitments, take $\rho_{r,t} = \bot$.

Salted compilation. The *salted* compilation applies the salted variant of Definition 2.4 to the same interactive argument. In each round $r \in [R]$, after computing $\mathbf{C}_r$, the prover samples an independent salt $\tau_r \leftarrow \{0,1\}^\lambda$ and derives

$$c_r := \mathsf{H}(\mathsf{enc}(\mathsf{tag}_{\mathrm{ch}}, \mathsf{srs}, \mathsf{i}, \mathsf{x}, r, \mathbf{C}_{\le r}, \mathbf{c}_{<r}, \tau_1, \ldots, \tau_r))\,.$$

The proof additionally contains $(\tau_r)_{r \in [R]}$, and the verifier recomputes the challenges with these salts. The other steps of Figure 2 do not change, and the proof of Lemma 2.12 applies to the salted compilation as well. The general strong resetting attack on PIOP-based arguments (Theorem 4.6) concerns this salted compilation; the PLONK attacks that need no salt concern the unsalted one.

Lemma 2.12 (Completeness of the compilation). *Suppose that* PIOP *has perfect completeness,* PCOM *has perfect commitment and evaluation correctness, and* H *is a random oracle to the PIOP challenge space* Ch*. Then the compiled argument in Figure 2 has perfect completeness.*

Proof. The honest prover and verifier use the same preprocessing commitments and the same sequence of prover commitments. Applying Ch_r to these identical inputs gives identical challenges in every round, hence identical query points. The prover passes each challenge to the next PIOP transition and opens the resulting polynomials using their original commitment coins. Perfect PIOP completeness

and the full support of uniform challenges on the finite space Ch imply acceptance for every challenge sequence and every honest random choice in its support, including preprocessing randomness. Thus $\mathbf{D_V}$ accepts this execution, even though its challenges depend on commitments. Perfect polynomial-commitment correctness makes each componentwise opening check accept as well.

3 Strong Resettable NIZK

To model selective randomness resets, we define a stateful randomness generator $\mathsf{RandGen}$ that reuses selected bits of a previous randomness string while independently sampling the remaining bits uniformly. The model does not give the adversary the sampled bits or allow it to choose their values.

Definition 3.1 (Randomness Generator for Strong Resetting Attacks). *Let $\ell = \ell(\lambda)$ be the length of the prover's randomness string. The stateful algorithm $\mathsf{RandGen}$ maintains a list L, initially empty, of ℓ-bit strings. On input (idx, I), where $\mathsf{idx} \in \mathbb{N}$ and $I \subseteq \{1, \ldots, \ell\}$, malformed inputs return $\perp$ without changing L. On well-formed inputs it operates as follows:*

- *If L is empty, ignore idx and I, sample $r_1 \leftarrow \{0,1\}^\ell$, set $L := [r_1]$, and return r_1.*
- *If $\mathsf{idx} > |L|$, return $\perp$ without changing L.*
- *Otherwise, let $r := L[\mathsf{idx}]$. Sample $r'_{\mathsf{fresh}} \leftarrow \{0,1\}^{\ell-|I|}$, and construct $r' \in \{0,1\}^\ell$ by scanning positions $i = 1, \ldots, \ell$:*
  - *if $i \in I$, set $r'[i] := r[i]$;*
  - *otherwise, set $r'[i]$ to the next unused bit of r'_{fresh}.*

  Append r' to L and return r'.

For (strong) resettable NIZK, as in the standard NIZK definition, we require indistinguishability between real and simulated proofs. Here the real proofs use randomness supplied by $\mathsf{RandGen}$. To reproduce the resulting correlations, the simulator receives the reset pattern and a simulated randomness tape, while the witness remains hidden.

The ideal oracle also enforces repeatability: identical inputs $(\mathsf{pp}, \mathsf{x}, \mathsf{w})$ and tapes must give identical proofs. It assigns a handle h to each distinct tuple $(\mathsf{pp}, \mathsf{x}, \mathsf{w}, r'_{\mathsf{sim}})$ and caches its proof. On every successful query, the simulator receives $(\mathsf{idx}, I, r'_{\mathsf{sim}}, h, \mathsf{hit})$, where hit indicates a cache hit; on a hit, the oracle returns the cached proof. This exposes whether the same witness was used for an earlier query with the same statement and tape. Both oracles reject invalid pairs before advancing $\mathsf{RandGen}$. We compare this reset-aware interface with other reset models in Section 3.1.

Definition 3.2 (Strong Resettable NIZK). *Let $\Pi = (\mathcal{G}, \mathcal{P}, \mathcal{V})$ be a* NARG *for an NP relation $\mathcal{R}$. Π is* strong resettable non-interactive zero-knowledge (srNIZK) *if there exists a PPT simulator $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_{\mathsf{step}})$ such that for every PPT strong resetting adversary $\mathcal{A}$, it holds that:*

$$\mathsf{Adv}^{\mathit{srNIZK}}_{\mathcal{A},\Pi}(\lambda) := \left| \Pr\left[\mathsf{srReal}^{\Pi}_{\mathcal{A}}(1^\lambda) = 1\right] - \Pr\left[\mathsf{srIdeal}^{\Pi}_{\mathcal{S},\mathcal{A}}(1^\lambda) = 1\right] \right| \leq \mathsf{negl}(\lambda)$$

$\underline{\mathsf{srReal}^{\Pi}_{\mathcal{A}}(1^\lambda):}$

1: $\mathsf{pp} \leftarrow \mathcal{G}(1^\lambda)$
2: Initialize $\mathsf{RandGen}$
3: $b \leftarrow \mathcal{A}^{\mathcal{O}_{\mathcal{P}}(\mathsf{pp},\cdot,\cdot,\cdot,\cdot)}(\mathsf{pp})$
4: Output b

$\underline{\mathcal{O}_{\mathcal{P}}(\mathsf{pp},\mathsf{x},\mathsf{w},\mathsf{idx},I):}$

1. If $(\mathsf{x},\mathsf{w}) \notin \mathcal{R}$, return $\perp$
2. $r' \leftarrow \mathsf{RandGen}(\mathsf{idx},I)$; if $r' = \perp$, return $\perp$
3. $\pi \leftarrow \mathcal{P}(\mathsf{pp},\mathsf{x},\mathsf{w};r')$
4. Return π

$\underline{\mathsf{srIdeal}^{\Pi}_{\mathcal{S},\mathcal{A}}(1^\lambda):}$

1: $(\mathsf{pp},\mathsf{st}) \leftarrow \mathcal{S}_0(1^\lambda)$
2: Initialize $\mathsf{RandGen}_{\mathsf{sim}}$ and empty dictionary $\mathcal{D} := \{\}$
3: $\mathsf{st}_t := (\mathcal{D},\mathsf{st})$
4: $b \leftarrow \mathcal{A}^{\mathcal{O}_{\mathcal{S}}(\mathsf{st}_t,\mathsf{pp},\cdot,\cdot,\cdot,\cdot)}(\mathsf{pp})$
5: Output b

$\underline{\mathcal{O}_{\mathcal{S}}(\mathsf{st}_t,\mathsf{pp},\mathsf{x},\mathsf{w},\mathsf{idx},I):}$

1. If $(\mathsf{x},\mathsf{w}) \notin \mathcal{R}$, return $\perp$
2. $r'_{\mathsf{sim}} \leftarrow \mathsf{RandGen}_{\mathsf{sim}}(\mathsf{idx},I)$; if $r'_{\mathsf{sim}} = \perp$, return $\perp$
3. Parse st_t as $(\mathcal{D},\mathsf{st})$
4. Let $\mathsf{key} := (\mathsf{pp},\mathsf{x},\mathsf{w},r'_{\mathsf{sim}})$ and $\mathsf{hit} := \mathbf{1}[\mathsf{key} \in \mathcal{D}]$
5. If $\mathsf{hit} = 1$, read $(h,\pi) \leftarrow \mathcal{D}[\mathsf{key}]$; otherwise set $h := |\mathcal{D}| + 1$
6. Set $\mu := (\mathsf{idx},I,r'_{\mathsf{sim}},h,\mathsf{hit})$
7. $(\widehat{\pi},\mathsf{st}') \leftarrow \mathcal{S}_{\mathsf{step}}(\mathsf{pp},\mathsf{x},\mu,\mathsf{st})$
8. If $\mathsf{hit} = 0$, set $\pi := \widehat{\pi}$ and $\mathcal{D}[\mathsf{key}] := (h,\pi)$
9. Set $\mathsf{st}_t := (\mathcal{D},\mathsf{st}')$; return π

Fig. 3: Experiments for srNIZK. The real and ideal experiments use independently initialized instances of $\mathsf{RandGen}$ and $\mathsf{RandGen}_{\mathsf{sim}}$, respectively.

where the experiments $\mathsf{srReal}^{\Pi}_{\mathcal{A}}$ *and* $\mathsf{srIdeal}^{\Pi}_{\mathcal{S},\mathcal{A}}$ *are defined in Figure 3. The oracle conventions of Section 2 apply.*

Resettable NIZK (rNIZK) is the restriction of this definition to whole-tape resets: every query uses $I = \emptyset$ or $I = [\ell]$. Section 3.1 spells out the restricted experiments and compares our model with classical interactive rZK.

3.1 Schedule-Oblivious Simulation and Reset-Aware Simulation

For rNIZK, use the experiments in Figure 3 with queries restricted to $I \in \{\emptyset,[\ell]\}$. A query with $I = \emptyset$ samples a fresh independent tape, while $I = [\ell]$ copies the referenced tape in full. The first well-formed query samples a fresh tape in either case. All other rules, including setup, validity checks, caching, and the simulator's interface, are unchanged. The definition requires a PPT simulator for which the distinguishing advantage is negligible against every PPT adversary making only these queries.

The reset-aware ideal oracle of Figure 3 gives the simulator the reset pattern and cache information on each successful query. We compare this definition with a schedule-oblivious interface, in which the simulator receives only the statement on each cache miss.

Definition 3.3 (Schedule-Oblivious Simulation). *We define the schedule-oblivious experiment as follows. It has the same real oracle, setup, simulated randomness generator, and dictionary as Figure 3. On each successful query, let* $\mathsf{key} = (\mathsf{pp}, \mathsf{x}, \mathsf{w}, r'_{\mathsf{sim}})$ *be the cache key. If* key *is already in* $\mathcal{D}$*, return its cached proof; the simulator is not called and its state does not change. Otherwise call*

$$(\pi, \mathsf{st}') \leftarrow \mathcal{S}_{\mathsf{step}}(\mathsf{pp}, \mathsf{x}, \mathsf{st}),$$

store π *at* key*, and set* $\mathsf{st} := \mathsf{st}'$*. Thus the simulator is called only on cache misses and receives only* $(\mathsf{pp}, \mathsf{x})$ *on each call; it receives no notification for skipped hits and no reset request* (idx, I) *or* r'_{sim}*. A protocol is schedule-oblivious reset zero knowledge if one PPT simulator makes the real and ideal experiments computationally indistinguishable for every PPT resetting adversary.*

Lemma 3.4 (Relation between the two reset interfaces). *If a protocol is schedule-oblivious reset zero knowledge, then it is reset-aware zero knowledge with no loss.*

Proof. Given a schedule-oblivious simulator, define a reset-aware simulator that ignores the metadata on a miss and makes the schedule-oblivious simulator's call. On a hit it leaves its state unchanged and returns an arbitrary string; the outer oracle discards that string and returns the cached proof. The two ideal experiments are identical.

Classical resettable zero knowledge is formulated for interactive executions in which a verifier can reset the prover and the simulator must reproduce the verifier's view [21]. Our definition instead gives an adversary a relation-checked NIZK proof oracle with an explicit selective-reset generator and cache. Relating the two definitions requires matching the setup, public inputs, auxiliary information, registration pattern, reset schedule, and simulated view. We leave such a comparison with the general interactive definition outside our scope.

The cache key determines the witness-equality information available to the simulator. Since it contains w, the bit hit tells the simulator whether the current $(\mathsf{pp}, \mathsf{x}, r'_{\mathsf{sim}})$ has appeared with the same witness; the handle h identifies the corresponding prior entry. The simulator can therefore use this equality pattern when generating proofs. For example, with two witnesses for one statement, a prover that returns $H(r \| \langle \mathsf{x}, \mathsf{w} \rangle)$ for a random function H gives equal proofs on a repeated tape exactly when the witness is the same, except for a random-function collision. Concealing witness-registration equalities would require an ideal interface that also hides this pattern.

3.2 Witness-Recovering Attack

We now define a class of attacks in which an adversary can recover the witness of a target statement.

Attack Model. After setup, we sample $(\mathsf{x}^*, \mathsf{w}^*, \mathsf{pub}, \mathsf{aux}) \leftarrow \mathcal{J}(1^\lambda, \mathsf{pp})$ using a fixed PPT algorithm $\mathcal{J}$. Here $(\mathsf{x}^*, \mathsf{w}^*) \in \mathcal{R}$, pub is public auxiliary information, and aux is a polynomial-size list of valid statement–witness pairs. The adversary receives $(\mathsf{pp}, \mathsf{x}^*, \mathsf{pub}, \mathsf{aux})$ and can request target proofs using the hidden witness w^* and reset patterns (idx, I). Its goal is to output any valid witness for x^*.

We also allow an auxiliary-proof oracle on arbitrary valid pairs supplied by the adversary. This oracle shares its $\mathsf{RandGen}$ state with the target oracle, so the adversary can reuse randomness across target and auxiliary proofs. The sampler is fixed as part of the experiment; it specifies the target distribution and any correlations with the auxiliary input. Experiments return 0 if it outputs a malformed sample or an invalid pair. Witness recovery alone need not violate zero-knowledge when simulated setup retains a trapdoor. Section 3.3 states the corresponding hardness condition under simulated setup.

Definition 3.5 (Witness-Recovering Attack). *Let Π be a non-interactive argument for $\mathcal{R}$, and fix a joint sampler $\mathcal{J}$ as above. A PPT adversary $\mathcal{A}$ performs a* witness-recovering attack *against Π if*

$$\Pr\left[\mathsf{WRA}^{\Pi}_{\mathcal{A}}(1^\lambda) = 1\right] \geq 1 - \mathsf{negl}(\lambda).$$

The variant with auxiliary input *uses $\mathsf{WRAaux}^{\Pi}_{\mathcal{A}}$ in place of $\mathsf{WRA}^{\Pi}_{\mathcal{A}}$. Both experiments use the fixed sampler $\mathcal{J}$ and appear in Figure 4.*

The same whole-tape restriction defines standard witness-recovering attacks.

3.3 Setup-Aware Recovery Baselines

To relate zero-knowledge to witness recovery, we also account for the simulator's setup state. As fixed in Section 2, an external relation parameter is supplied unchanged to all algorithms, including $\mathcal{S}_0$. If setup determines the relation $\mathcal{R}_{\mathsf{pp}}$, however, the simulated setup may retain a trapdoor. Thus a hardness assumption for honestly generated setup alone cannot serve as a baseline for every zero-knowledge simulator.

Fix the joint sampler $\mathcal{J}$ from Definition 3.5. It is run after setup and returns $(\mathsf{x}^*, \mathsf{w}^*, \mathsf{pub}, \mathsf{aux})$; the target and every pair in aux must be valid for the public parameters. The adversary and baseline receive the same public data $(\mathsf{pp}, \mathsf{x}^*, \mathsf{pub}, \mathsf{aux})$. The target oracle always uses the sampled w^*, and the auxiliary oracle accepts any valid pair supplied by the adversary. We use *any-witness recovery* as the default success event, namely $(\mathsf{x}^*, \widehat{\mathsf{w}}) \in \mathcal{R}_{\mathsf{pp}}$. The experiment's fixed abort rule for malformed or invalid sampler outputs applies under both honest and simulated setup.

For a reset-aware simulator $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_{\mathsf{step}})$, define the *simulator-assisted baseline* as follows. It samples $(\mathsf{pp}, \mathsf{st}) \leftarrow \mathcal{S}_0(1^\lambda)$, runs $\mathcal{J}(1^\lambda, \mathsf{pp})$, and gives the public outputs and st to a PPT baseline algorithm. The algorithm receives no real target proofs or w^*; it may use $\mathcal{S}_{\mathsf{step}}$, the simulated reset generator, and the public auxiliary data. In the ROM it also runs the full simulated oracle environment,

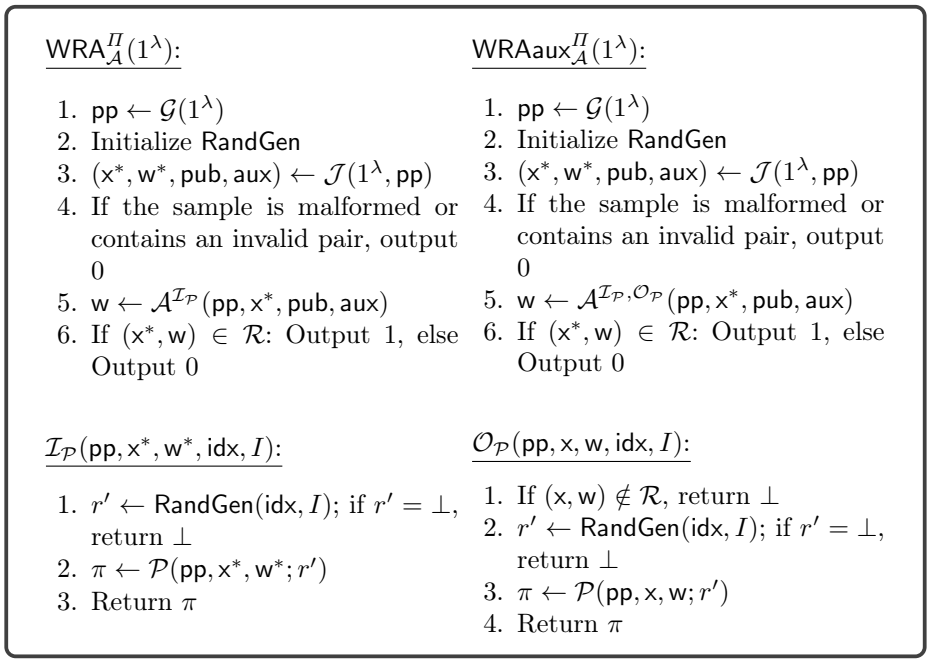


Fig. 4: Experiments defining witness-recovering attacks. The oracles $\mathcal{I}_\mathcal{P}$ and $\mathcal{O}_\mathcal{P}$ share the same $\mathsf{RandGen}$ instance. Both oracles use the displayed pp, and $\mathcal{I}_\mathcal{P}$ is bound to $(\mathsf{x}^*, \mathsf{w}^*)$. The oracle $\mathcal{I}_\mathcal{P}$ answers only for the target statement x^* using w^*, while $\mathcal{O}_\mathcal{P}$ accepts arbitrary valid statement-witness pairs supplied by the adversary.

including public queries and the programming allowed by the underlying zero-knowledge definition. Invalid sampler output makes the baseline return 0 before this interaction. Thus the baseline measures recovery using simulated setup and the resources available to a witness-free simulator.

Theorem 3.6 (Recovery is bounded by the simulator-assisted baseline). *Suppose Π is strong reset-aware zero knowledge with simulator $\mathcal{S}$. For every PPT any-witness recovery adversary A in the auxiliary-input experiment, there are a PPT baseline algorithm $B_\mathcal{S}$ and a PPT distinguisher D such that*

$$\Pr\left[\mathsf{WRAaux}^{\Pi,\mathcal{J}}_{A} = 1\right] \leq \Pr\left[\mathsf{Base}^{\mathcal{S},\mathcal{J}}_{B_\mathcal{S}} = 1\right] + \mathsf{Adv}^{\mathsf{srNIZK}}_{D,\Pi}(\lambda).$$

The baseline has the setup and resources just specified. Consequently, hardness against this simulator-assisted baseline implies negligible witness-recovery probability. The baseline includes the simulator's state and simulated-oracle resources under simulated setup.

Proof. The distinguisher samples $\mathcal{J}$ using its challenge public parameters, runs A, forwards target queries using w^*, and forwards auxiliary queries. It outputs

one exactly when A outputs a valid witness. Its real-world success is the left-hand side.

The baseline algorithm runs the ideal-world interaction without knowing w^*. It represents the fixed target witness by a formal symbol in its cache and otherwise uses $\mathcal{S}_{\mathsf{step}}$ and the simulated reset generator to answer queries. If A supplies a valid auxiliary pair for the target statement, the baseline returns that pair's witness and succeeds immediately. On executions with no such query, the symbolic target entry gives exactly the ideal cache behavior: every target query uses the same hidden witness, and no auxiliary query has the target public context. Thus baseline success is at least the distinguisher's ideal-world success. The indistinguishability bound gives the theorem.

For example, let setup generate a trapdoor permutation key pair $(\mathsf{pk}, \mathsf{sk})$, publish pk, and define $\mathcal{R}_{\mathsf{pp}}(x, w)$ by $f_{\mathsf{pk}}(w) = x$. A prover that outputs w reveals the unique witness, which is hard to recover from a random image without the trapdoor. Yet a simulator can generate setup while retaining sk and reproduce every proof by inversion. The simulator-assisted baseline has this same ability, so Theorem 3.6 correctly permits witness recovery.

The theorem concerns any-witness recovery. For exact recovery, where success requires $\widehat{\mathsf{w}} = \mathsf{w}^*$, a valid auxiliary witness for x^* need not be the sampled witness. If the auxiliary oracle permits target-context queries, an exact-recovery comparison must give the baseline the equality test $[v = \mathsf{w}^*]$ for each such candidate, or otherwise bound the information exposed by these tests. If target-context auxiliary queries are disallowed, the symbolic-cache simulation above applies without this extra equality information. Any resulting hardness claim must also specify a distribution in which guessing the exact witness is hard.

4 Resetting Attacks on Non-Interactive Arguments

We present resetting attacks on Fiat–Shamir NIZKs in both models. Throughout, target samplers are independent of the Fiat–Shamir oracle.

The mask structures we analyze occur in dalek's Bulletproofs range prover and gnark's PLONK prover [25,28]; realizing our selective-reset interface in either implementation requires separate analysis.

4.1 Standard Resetting Attacks

We begin by demonstrating a standard resetting attack on the Fiat–Shamir compilation of Maurer's proof of homomorphism preimage [49].

4.1.1 Maurer's Proof of Homomorphism Preimage We consider a generator $\mathsf{GroupHomGen}$ that samples finite groups $(\mathbb{G}, \star)$ and $(\mathbb{H}, \otimes)$ and an efficiently computable homomorphism $\phi : \mathbb{G} \to \mathbb{H}$. We assume efficient group operations and uniform sampling. Maurer's protocol proves knowledge of a preimage under ϕ; its Fiat–Shamir compilation appears in Figure 5.

Relation: $\mathcal{R}_{\mathsf{hom\text{-}preim}} := \{(\mathsf{x}, \mathsf{w}) \mid \mathsf{x} = \phi(\mathsf{w})\}$.

Parameters. Finite challenge set $\mathcal{C} \subseteq \mathbb{Z}_{\geq 0}$ with $|\mathcal{C}| \geq 2$ and repetition parameter $\kappa \geq 1$; the random oracle H outputs values in $\mathcal{C}^\kappa$. Knowledge soundness requires Maurer's extraction condition on $\mathcal{C}$ [49, Theorem 3] and negligible $|\mathcal{C}|^{-\kappa}$; our results use $\mathcal{C} = \{0, 1\}$ and $\kappa \geq \lambda$, which satisfy both.

Non-Interactive Protocol Π_{hom}:

- $\mathcal{G}(1^\lambda)$: Return $\mathsf{pp} := (\mathbb{G}, \mathbb{H}, \phi) \leftarrow \mathsf{GroupHomGen}(1^\lambda)$.
- $\mathcal{P}(\mathsf{pp}, \mathsf{x}, \mathsf{w})$:
  1. For $i \in \{1, \ldots, \kappa\}$, sample $r_i \leftarrow \mathbb{G}$ and compute $a_i := \phi(r_i)$.
  2. Compute $(c_1, \ldots, c_\kappa) := \mathsf{H}(\mathsf{pp}, \mathsf{x}, a_1, \ldots, a_\kappa) \in \mathcal{C}^\kappa$.
  3. For $i \in \{1, \ldots, \kappa\}$, compute $z_i := r_i \star \mathsf{w}^{c_i}$.
  4. Output $\pi = \{(a_i, z_i)\}_{i=1}^{\kappa}$.
- $\mathcal{V}(\mathsf{pp}, \mathsf{x}, \pi)$:
  1. Parse $\pi = \{(a_i, z_i)\}_{i=1}^{\kappa}$.
  2. Compute $(c_1, \ldots, c_\kappa) := \mathsf{H}(\mathsf{pp}, \mathsf{x}, a_1, \ldots, a_\kappa)$.
  3. For each $i \in \{1, \ldots, \kappa\}$, check $\phi(z_i) = a_i \otimes \mathsf{x}^{c_i}$.
  4. Accept if all checks pass; otherwise reject.

Fig. 5: Fiat–Shamir compilation of Maurer's protocol for knowledge of a homomorphism preimage.

Theorem 4.1. *Maurer's protocol with $\mathcal{C} = \{0, 1\}$ and $\kappa \geq \lambda$ is not rNIZK if its homomorphism has nontrivial kernel and image. The homomorphism may be externally fixed, or generated by setup in an efficiently checked parameter format that guarantees these properties. Two proof queries give distinguishing advantage at least $1/2 - 2^{-\lambda} - 2^{-\kappa}$.*

Proof sketch. The adversary samples a witness w uniformly, resampling if $\phi(\mathsf{w})$ is the identity. A proof for the identity with its identity witness reveals the prover's masks. Reusing those masks on $\mathsf{x} = \phi(\mathsf{w})$ then reveals the sampled witness whenever a target challenge is 1. Every preimage of x is equally likely, even given the simulator's setup state and reset information. Since there are at least two preimages, the simulator's answers identify the sampled witness with probability at most $1/2$. Bounding the resampling and challenge failures gives the claim. Section A.1 gives the full proof, the sharper bound in terms of kernel size, and the corresponding attack on externally fixed one-way homomorphisms.

For the field-challenge Okamoto instance in the overview, the same argument gives advantage at least $1 - 3/p$ (Corollary A.2).

Witness Recovery. Resetting also allows an adversary to recover the witness from two honest proofs. Take $\mathcal{C} = \{0, 1\}$ and $\kappa \geq \lambda$. First, the adversary requests an

auxiliary proof for $(e_{\mathbb{H}}, e_{\mathbb{G}})$, whose responses reveal $z'_i = r_i$. It then resets the prover to the same tape and requests a proof of the target (x, w). The new responses satisfy $z_i = r_i \star \mathsf{w}^{c_i}$, so any challenge $c_i = 1$ gives

$$\mathsf{w} = (z'_i)^{-1} \star z_i.$$

This two-query attack also works in noncommutative groups. For $\mathsf{x} \neq e_{\mathbb{H}}$, it succeeds with probability at least $1 - 2^{-\kappa}$; the identity target already has a known witness. Section A.1 also analyzes extraction from accepting transcripts.

The witness-recovery attack applies to every homomorphism in Figure 5 with these challenge parameters. When the kernel and image are nontrivial, it reveals which preimage the prover used and therefore breaks zero-knowledge. For externally fixed one-way permutations, witness recovery also gives a privacy attack. Section A.1 proves this extension; Section 3.3 explains the role of simulated setup.

4.1.2 PIOP-Based Arguments with Auxiliary Statements We next give witness-recovering attacks against Fiat–Shamir NARGs compiled from PIOPs. These attacks use resetting to obtain enough polynomial evaluations to recover the witness.

Attack Intuition. In a PIOP, the prover outputs polynomials and the verifier queries them at evaluation points derived via a random oracle. If a subset of polynomials encodes the witness, the adversary can force the prover to generate multiple proofs with different evaluation points by resetting randomness. With enough evaluations, we solve the resulting equations to recover the witness.

The challenge is that identical inputs yield identical evaluation points. We address this via two techniques: (1) standard resetting with auxiliary inputs, and (2) strong resetting with partial randomness control.

The attack removes a common mask before recovering the witness. More generally, the auxiliary evaluations need only determine the mask contribution to the target evaluations. This gives the following condition, which also allows several polynomials and different query maps.

Theorem 4.2. *Let Π_{FS} be a Fiat–Shamir NARG compiled from a PIOP. Suppose that in each session i, k known auxiliary proofs and one target proof on a shared tape expose*

$$y_{i,j} = b_{i,j} + L_{i,j}\theta_i, \qquad y_i^* = A_i a + B_i \theta_i + d_i,$$

where $a \in \mathbb{F}^D$ admits an efficient decoder for a valid target witness, θ_i is the common mask, and the matrices and offsets are computable from the public transcripts and auxiliary witnesses. Let L_i stack the auxiliary maps $L_{i,j}$. If $\ker L_i \subseteq \ker B_i$ in every session and the stacked target maps A_i have rank D, except with negligible probability, then n sessions give a standard witness-recovering attack using $n(k+1)$ proof queries. All dimensions and query counts are polynomially bounded; abort and decoding probabilities are included in the failure bound.

Input: Let $\mathcal{A}$ be a resetting adversary with auxiliary statement-witness pairs $\mathsf{aux} = \{(\mathsf{x}_1, \mathsf{w}_1), \ldots, (\mathsf{x}_k, \mathsf{w}_k)\}$ and public parameters pp, where $(\mathsf{x}_j, \mathsf{w}_j) \in \mathcal{R}$ for all $j \in \{1, \ldots, k\}$. Let x_{k+1} be the target statement, which is provided to $\mathcal{A}$ by the challenger as per Definition 3.5.

Output: Witness w_{k+1} such that $(\mathsf{x}_{k+1}, \mathsf{w}_{k+1}) \in \mathcal{R}$.

Attack Procedure: Assume $k \geq 1$ and make exactly $k+1$ oracle queries per session, with no intervening queries.

1. For $i \in \{1, \ldots, n\}$:
   (a) Let $t_i := (i-1)(k+1)+1$ index this session's first tape.
   (b) For $j \in \{1, \ldots, k\}$:
      - Query $\mathcal{O}_{\mathcal{P}}$ with $(\mathsf{x}_j, \mathsf{w}_j, \mathsf{idx} = 1, I = \varnothing)$ if $j = 1$; otherwise use $(\mathsf{x}_j, \mathsf{w}_j, \mathsf{idx} = t_i, I = \{1, \ldots, \ell\})$
      - Receive $\pi_{i,j} \leftarrow \mathcal{P}(\mathsf{pp}, \mathsf{x}_j, \mathsf{w}_j; r_i)$ containing evaluation $\Phi_{i,j}(z_{i,j})$
   (c) Compute $r_i(z_{i,j}) = \Phi_{i,j}(z_{i,j}) - \hat{\Phi}_j(z_{i,j})$ for $j \in \{1, \ldots, k\}$, and interpolate $r_i(X)$ in its known mask space
   (d) Query $\mathcal{I}_{\mathcal{P}}$ with $(\mathsf{idx} = t_i, I = \{1, \ldots, \ell\})$
   (e) Receive $\pi_{i,k+1} \leftarrow \mathcal{P}(\mathsf{pp}, \mathsf{x}_{k+1}, \mathsf{w}_{k+1}; r_i)$ containing evaluation $\Phi_{i,k+1}(z_{i,k+1})$
2. Compute $\hat{\Phi}_{k+1}(z_{i,k+1}) = \Phi_{i,k+1}(z_{i,k+1}) - r_i(z_{i,k+1})$ for $i \in \{1, \ldots, n\}$, and recover w_{k+1} from these evaluations
3. Return w_{k+1}

Fig. 6: Resetting attack with auxiliary input and a common polynomial mask. This instance recovers the full mask within its known polynomial space.

We prove the theorem in Section A.2 and apply the attack in Figure 6 to PLONK [37].

4.1.3 Instantiation with PLONK In PLONK, two known auxiliary assignments with suitable evaluation points determine the two coefficients of each wire mask on a shared tape. Removing these masks from the target proof gives one unmasked evaluation of each wire polynomial. Repeating this procedure across d sessions recovers the assignment by interpolation, using $3d$ proof queries.

We analyze the patched protocol specified in Construction 1 of [64], which incorporates the June 29, 2022 correction to [37]. Let $\mathbb{F}$ have prime order p, let $H \subset \mathbb{F}^*$ be a subgroup of size d, and let s denote the KZG setup exponent. Assume d is polynomially bounded and $1/p$ is negligible. For a circuit index i, let the public statement be $\mathsf{x} = (w_t)_{t \in [d']}$ and the private witness be $\mathsf{w} = (w_t)_{t \in [d'+1, 3d]}$. Write L_t for the Lagrange basis polynomials on the ordered domain H and $Z_H(X) = X^d - 1$ for its vanishing polynomial. The three un-

masked wire polynomials are

$$\widehat{A}(X) = \sum_{t\in[d]} w_t L_t(X),$$
$$\widehat{B}(X) = \sum_{t\in[d]} w_{t+d} L_t(X),$$
$$\widehat{C}(X) = \sum_{t\in[d]} w_{t+2d} L_t(X).$$

The public proof includes the evaluations of all three wire polynomials at its evaluation challenge. Statements and the circuit index are included in the transcript, and the different challenge queries have unambiguous encodings.

Randomness. Write the masked wire polynomials as $A = \Phi_L$, $B = \Phi_R$, and $C = \Phi_O$, with

$$A = \widehat{A} + (\rho_1 X + \rho_2) Z_H, \quad B = \widehat{B} + (\rho_3 X + \rho_4) Z_H, \quad C = \widehat{C} + (\rho_5 X + \rho_6) Z_H,$$

where the unmasked polynomials have degree less than d. Each scalar is sampled using a separate, publicly identified block of the prover tape. The remaining blinders likewise have separate blocks. Uniform field sampling may be idealized directly; a fixed-block sampler whose joint distribution is within statistical distance ϵ_{samp} of the ideal execution adds at most ϵ_{samp} to each success bound below. This block convention is necessary to implement field-level resets with Definition 3.1.

The prover first commits to A, B, C, then derives the permutation challenges β, γ. Its permutation polynomial has the form

$$P = \widehat{P} + (\rho_7 X^2 + \rho_8 X + \rho_9) Z_H.$$

It then commits to P and to the quotient pieces, and derives the evaluation challenge ζ from the statement and all commitments.

Proposition 4.3. *Fix a circuit and two known valid auxiliary assignments, chosen independently of the random oracle, whose public statements differ from each other and from the target statement. With the randomness layout above, the attack of Figure 6, run for d sessions with two auxiliary proofs each, recovers the complete wire assignment of the target from $3d$ proof queries. Its failure probability is at most*

$$\frac{16d^2 + 6d}{p} + \epsilon_{\text{samp}}.$$

Section A.3 gives the proof. The bound is for the specified transcript and independent field blinders, rather than every implementation called PLONK. The BN254 PLONK prover in gnark samples degree-one wire masks and a degree-two permutation mask from external randomness [25]. Its concurrent prover steps and entropy interface still require a separate analysis to realize our fixed tape-block resets.

4.2 Strong Resetting Attacks

4.2.1 Rewinding through Salted Fiat–Shamir Strong resets can reproduce the rewinding used by many knowledge extractors. In salted Fiat–Shamir, the adversary preserves the prover's randomness while changing the salt, obtaining a fresh challenge from the same prover state. If extraction uses only such randomly sampled challenges, the adversary can use the resulting transcripts to recover the witness.

The attack need not extract from an arbitrary malicious prover: it suffices to recover a witness from the honest prover by replaying saved prefixes with fresh challenges. We state the required success and running-time bounds conditionally on the target, since repetitions retain the same target.

Definition 4.4 (Honest-Prover Replay Extraction). *Fix a valid PPT target sampler $\mathcal{J}$ from Definition 3.5, independent of the compilation oracle. Write $\zeta = (\mathsf{pp}, \mathsf{x}^*, \mathsf{w}^*, \mathsf{pub}, \mathsf{aux})$ for its full sampled context, including the hidden witness. An extractor E receives only $(1^\lambda, \mathsf{pp}, \mathsf{x}^*, \mathsf{pub}, \mathsf{aux})$ and accesses the honest prover on $(\mathsf{pp}, \mathsf{x}^*, \mathsf{w}^*)$ through the following interface:*

- *Start a fresh root with an independent prover tape and receive its first message.*
- *Save and restore any previously reached prefix, preserving that root's tape and state.*
- *Extend a saved prefix with an independent uniform verifier challenge and receive the next prover message.*

Only protocol messages and sampled challenges are revealed; saved states are opaque handles. Choices of roots and prefixes may be adaptive. The extractor may discard sampled challenges and try again, but every draw counts toward its running time, and it cannot prescribe a challenge. The extractor receives no witness, secret state, setup trapdoor, or access to the compilation oracle.

The protocol admits efficient honest-prover replay extraction *for $\mathcal{J}$ if there are such an E, polynomials $B, \psi \geq 1$, and a negligible function ν such that*

$$\begin{gathered} s(\zeta) := \Pr[E \text{ returns a valid witness within } B(\lambda) \text{ steps} \mid \zeta], \\ \Pr_{\zeta}[s(\zeta) < 1/\psi(\lambda)] \leq \nu(\lambda). \end{gathered} \tag{1}$$

The first probability averages fresh root tapes, challenges, and extractor coins. Running time counts local computation and all interface calls, including discarded samples. Input lengths and the cost of one honest execution are polynomially bounded for the fixed sampler; the polynomials may depend on these bounds and on $\mathcal{J}$, but not on the sampled context ζ. We say the protocol admits this extraction without specifying a sampler when the property holds for every valid PPT target sampler.

Theorem 4.5. *Let Π be a public-coin argument with efficient extraction from its honest prover by rewinding with fresh uniform challenges (Definition 4.4). Its*

salted Fiat–Shamir compilation, with independent salt blocks at fixed public tape positions and negligible joint abort and sampling errors over the replay execution, admits a witness-recovering attack in the strong resettable model.

Proof sketch. To return to round i of a saved execution, the adversary preserves the prover's coins and the salts before that round. Refreshing the remaining salts then reproduces the earlier messages while supplying new random challenges, except when salts collide. The adversary can therefore provide the extractor with the interactions it requests. The extraction guarantee gives witness recovery after polynomially many repetitions. The prefix-replay analysis proves the simulation and bounds the running time and failure probability.

For Σ-protocols such as Schnorr and Okamoto, two accepting transcripts with the same first message and distinct challenges suffice. Multi-round extraction may require a tree of transcripts; the attack constructs this tree when its size is polynomial (Corollaries A.9 and A.10). Section A.4 proves Theorem 4.5 and these corollaries.

Bulletproofs. Bulletproofs' inner-product argument also permits efficient extraction of this form (Proposition A.11). Bulletproofs range proofs expose an evaluation of a quadratic polynomial whose constant term encodes the committed value. Refreshing the blinder of one coefficient commitment changes the evaluation challenge while preserving the polynomial. Three proofs then recover the value by interpolation. If the blinder of the masking-vector commitment can be refreshed independently as well, five proofs recover both the value and its commitment blinding. For m aggregated values, $3m + 2$ proofs suffice, independent of bit length. These attacks need no salts (Theorem A.12 and Corollary A.13); Section A.5 gives the proofs.

4.2.2 PIOP-Based Arguments under Strong Resets

We now revisit the attack from Figure 6 in the strong resettable setting. Here the adversary obtains new evaluations without auxiliary inputs: it fixes the prover randomness and the prefix defining a witness polynomial, then varies the later salts.

Theorem 4.6. *Let Π_{sFS} be a salted Fiat–Shamir NARG compiled from a perfectly complete PIOP with the coefficient-oracle knowledge guarantee of Definition 2.9. With public proving keys and independently resettable salt blocks, Π_{sFS} admits a witness-recovering attack in the **strong** resettable model.*

The knowledge guarantee requires one extractor with a fixed polynomial time bound for all admissible provers, including malicious ones, given access to their polynomial coefficients. It holds whenever the witness can be decoded from the first prover message, as in PLONK (Lemma 2.10). We learn coefficients consistent with evaluations at saved prefixes and run this extractor on the resulting table (Section A.7). The PLONK attack uses direct interpolation (Figure 7) and does not require this extraction assumption.

Input: Strong resetting adversary $\mathcal{A}$ with public parameters pp and target statement x^* (provided by the challenger as per Definition 3.5)

Output: Witness w^* such that $(\mathsf{x}^*, \mathsf{w}^*) \in \mathcal{R}$

Attack Procedure:

1. Request a fresh target proof. Save its tape index v, challenge transcript, and evaluations.
2. For each selected polynomial group g, sent before challenge t_g:
   (a) Let I_g contain the prover-coin positions and the salt blocks before round t_g.
   (b) Make $N_g - 1$ target queries with $(\mathsf{idx} = v, I = I_g)$, collecting evaluations of the fixed group.
   (c) Combine these with the base proof's evaluations and recover the group's coefficients.
3. Decode a valid witness from the recovered groups and the base challenge transcript.

Fig. 7: Strong resetting attack against a PIOP-based NARG. Each group is recovered from independent continuations of its defining prefix in one saved base execution. The stacked evaluation maps must have full coefficient rank.

Concrete Example with PLONK. Unlike the standard resetting attack, the strong variant requires no auxiliary inputs: the adversary fixes the main prover randomness and refreshes salts after the wire commitments, obtaining evaluations of the same masked polynomials Φ_L, Φ_R, Φ_O at distinct points.

Since the masked wire polynomials remain fixed and have degree at most $d+1$, the adversary can interpolate them from $\eta = d+2$ distinct evaluations. It then evaluates the recovered polynomials on H, where Z_H vanishes, to remove the masks and recover the wire assignment.

4.2.3 Generality of the Salted Fiat–Shamir Modeling The strong resetting attack also applies to non-salted Fiat–Shamir NARGs when another randomness component plays the role of the salt. In unsalted PLONK, the adversary refreshes only the constant blinder ρ_9 of the permutation polynomial. This keeps the wire polynomials fixed while changing the permutation commitment and hence the evaluation-challenge transcript. The attack then proceeds as in the salted case, recovering the witness from $d+2$ target proofs with failure probability at most $(2d^2 + 11d + 6)/p + \epsilon_{\text{samp}}$ (Section A.8).

Randomized polynomial commitments can play the same role in the PIOP compiler of [20,45] (Section 2.6) when resampling commitment coins yields fresh encodings except with negligible probability and preserves the prefix defining

the witness polynomials. For multi-round PIOPs, this includes their dependence on earlier challenges.

Bit-Level Resets of a Single Mask. The attacks above refresh a whole block of the tape: a salt, or an independently sampled blinder. A protocol whose tape consists of a single block is still exposed, because a reset may preserve all but one bit of that block. Consider unsalted Fiat–Shamir Schnorr [63] for $\mathsf{x} = g^{\mathsf{w}}$ in a group of odd prime order p with generator g, where $1/p$ is negligible. The prover computes $a := g^r$, $c := \mathsf{H}(\mathsf{pp}, \mathsf{x}, a) \in \mathbb{Z}_p$, and $z := r + c\mathsf{w}$, and outputs (a, z). Its only randomness is the mask r, so a whole-tape reset of a target proof returns either the same proof or an independent one. Suppose the prover derives the mask from its tape $\rho \in \{0,1\}^\ell$ as $r := \sum_{i \in [\ell]} \rho[i]\, 2^{i-1} \bmod p$; for $\ell \geq \lceil \log_2 p \rceil + \lambda$ this mask is within statistical distance $2^{-\lambda}$ of uniform.

Proposition 4.7. *With this tape encoding, unsalted Fiat–Shamir Schnorr admits a witness-recovering attack in the strong resettable model. The attack uses at most $\lambda + 1$ target proofs and no auxiliary proofs, and fails with probability at most $2^{-\lambda} + 1/p$.*

Proof. Fix a position $i \in [\ell]$ and let $\Delta := 2^{i-1}$. The adversary requests a target proof (a, z) on a fresh tape ρ. It then queries $\mathcal{I}_{\mathcal{P}}$ with $(\mathsf{idx} = 1, I = [\ell] \setminus \{i\})$ until a returned proof (a', z') has $a' \neq a$, making at most λ such queries. Each of these proofs uses a tape that agrees with ρ outside position i and has an independent uniform bit at position i. If that bit equals $\rho[i]$, the proof is (a, z) again. Otherwise the mask is $r' = r + s\Delta$, where $s = 1$ if $\rho[i] = 0$ and $s = -1$ if $\rho[i] = 1$, and $a' = a \cdot g^{s\Delta}$. Since p is odd, neither Δ nor 2Δ is divisible by p. Hence $a' \neq a$, and the adversary learns s by comparing a' with $a \cdot g^{\Delta}$. All λ queries repeat the bit $\rho[i]$ with probability $2^{-\lambda}$.

The tapes and the target are independent of the random oracle, and $c' = \mathsf{H}(\mathsf{pp}, \mathsf{x}, a')$ is its value at an input different from that of c, so $c' = c$ with probability $1/p$. When $c' \neq c$, subtracting $z = r + c\mathsf{w}$ from $z' = r + s\Delta + c'\mathsf{w}$ gives

$$\mathsf{w} = \frac{z' - z - s\Delta}{c' - c}.$$

The attack needs no salt, no second randomness block, and no auxiliary statement; it uses only the fact that the two masks differ by a value the adversary can compute. The same argument applies to any Σ-protocol whose response has the form $z = r + c\,\mathsf{w}$ over a field of superpolynomial size, under any tape encoding in which flipping one bit shifts the mask by a known amount and changes the first message.

5 Generic Compiler for Strong Resettable NIZK

We give a generic compiler from any NIZK to a strong resettable one (Figure 8). The prover derives a seed from $(\mathsf{pp}, \mathsf{x}, \mathsf{w})$ and expands it to the full randomness tape before running the original prover.

Input: A NIZK Π, an sRK-PRF family $\mathcal{F}$, and a pseudorandom generator (PRG) G.
Compiled algorithms:

1. $\tilde{\mathcal{G}}(1^\lambda)$: Return $\mathcal{G}(1^\lambda)$.
2. $\tilde{\mathcal{V}}(\mathsf{pp}, \mathsf{x}, \pi)$: Return $\mathcal{V}(\mathsf{pp}, \mathsf{x}, \pi)$.
3. $\tilde{\mathcal{P}}(\mathsf{pp}, \mathsf{x}, \mathsf{w}; r)$, for $r \in \{0,1\}^\ell$: Set $m := m(\lambda, \mathsf{pp}, \mathsf{x})$, $u := \langle \mathtt{rnd}, \mathsf{pp}, \mathsf{x}, \mathsf{w}, m \rangle$, and $\tilde{r} := \mathsf{F}^*_r(u, m)$. Return $\mathcal{P}(\mathsf{pp}, \mathsf{x}, \mathsf{w}; \tilde{r})$.

Fig. 8: Generic compiler for strong resettable NIZK. The public polynomial bound $m = m(\lambda, \mathsf{pp}, \mathsf{x}) \geq 1$ bounds the prover's tape length for every valid witness; unused bits are ignored. Here $\mathsf{F}^*_r(u, m) := \mathsf{Expand}_{\mathsf{G}}(\mathsf{F}_r(u), m)$, the first m bits of a pseudorandom stream.

Partial resets act on the seed before derivation, so we require a PRF secure under them, a form of related-key security [11].

Definition 5.1 (Strong Resettable-Key PRF). *A PRF family $\mathcal{F} = \{\mathsf{F}_k : \{0,1\}^* \to \{0,1\}^\lambda\}_{k \in \{0,1\}^\ell}$ is a* strong resettable-key PRF (sRK-PRF) *if for all PPT adversaries $\mathcal{C}$:*

$$\mathsf{Adv}^{\mathsf{sRK-PRF}}_{\mathcal{C},\mathcal{F}}(\lambda) := \left|\Pr\left[\mathcal{C}^{\mathcal{O}_{\mathsf{real}}}(1^\lambda) = 1\right] - \Pr\left[\mathcal{C}^{\mathcal{O}_{\mathsf{rand}}}(1^\lambda) = 1\right]\right| \leq \mathsf{negl}(\lambda)$$

Each experiment initializes its own empty RandGen *state. The oracles for each query $(x, (\mathsf{idx}, I))$ with $x \in \{0,1\}^*$, $\mathsf{idx} \in \mathbb{N}$, $I \subseteq [\ell]$ operate as follows:*

- $\mathcal{O}_{\mathsf{real}}$*:* $k' \leftarrow \mathsf{RandGen}(\mathsf{idx}, I)$*; if* $k' = \bot$*, return* $\bot$*; otherwise return* $\mathsf{F}_{k'}(x)$
- $\mathcal{O}_{\mathsf{rand}}$*:* $k' \leftarrow \mathsf{RandGen}(\mathsf{idx}, I)$*; if* $k' = \bot$*, return* $\bot$*; otherwise return* $\mathsf{R}(k', x)$ *for random* $\mathsf{R} : \{0,1\}^\ell \times \{0,1\}^* \to \{0,1\}^\lambda$

We instantiate sRK-PRF in the ROM as $\mathsf{F}_k(x) = \mathsf{H}(k \| x)$ for $\ell \geq \lambda$. This security notion is equivalent to full XOR-related-key security, up to polynomial overhead. Sections 5.1 and 5.3 give the precise assumptions and reductions.

Theorem 5.2. *If Π is a NIZK, $\mathcal{F}$ is a secure* sRK-PRF*, and $\mathsf{G} : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ is a secure PRG, then the compiled system $\tilde{\Pi}$ in Figure 8 is a srNIZK.*

Perfect completeness and soundness are preserved; setup, verification, and proof size are unchanged. Each proof uses one sRK-PRF evaluation and $\lceil m/\lambda \rceil$ calls to a λ-to-2λ PRG. In the ROM, the added work is one context hash and expansion to m bits; hash-based expansion adds no group operations or polynomial arithmetic. Section 5.2 gives the precise cost ratio.
Proof sketch. By sRK-PRF security, replace seed derivation by a random function of the reset seed and proof context. A PRG hybrid makes the tape for each

distinct pair uniform; repetitions reuse the same tape and proof. Adaptive multi-theorem zero-knowledge then simulates fresh proofs. The simulator updates its state only on cache misses and ignores the reset schedule. Section 5.2 proves the transitions for adaptive tape lengths and gives the PRF and ROM corollaries.

5.1 Random-Oracle Instantiation of an sRK-PRF

Theorem 5.3. *Let $\ell \geq \lambda$ and let $\mathsf{H} : \{0,1\}^* \to \{0,1\}^\lambda$ be a random oracle. Define the function family $\mathcal{F} = \{\mathsf{F}_k : \{0,1\}^* \to \{0,1\}^\lambda\}_{k \in \{0,1\}^\ell}$ where $\mathsf{F}_k(x) = \mathsf{H}(k\|x)$. Then $\mathcal{F}$ is an* sRK-PRF.

This hidden-key hit argument parallels Bellare and Kohno's ideal-cipher analysis of XOR-related-key attacks [11]; here the public primitive is a random oracle, and the keys arise through adaptive partial resets.

We prove Theorem 5.3 with public random-oracle access. Let $\mathcal{C}$ make at most q_F queries to the reset oracle and q_H queries to H, in any adaptive order. The key length ℓ is fixed, so $k\|x$ uniquely encodes (k, x). Write Real for the experiment with oracles $(\mathcal{O}_{\mathsf{real}}, \mathsf{H})$ and Ideal for the experiment with $(\mathcal{O}_{\mathsf{rand}}, \mathsf{H})$, where R and H are independent random functions. We show

$$|\Pr[\mathsf{Real} = 1] - \Pr[\mathsf{Ideal} = 1]| \leq \frac{q_F q_H}{2^\ell}.$$

Proof. **The ideal transcript hides absolute key values.** Let $K_1, \ldots, K_N$, with $N \leq q_F$, be the keys appended to the ideal experiment's $\mathsf{RandGen}$ list. These keys need not be independent: partial resets correlate them, and repeated function values can reveal key equalities. We instead use a symmetry of the entire experiment.

Fix any $\Delta \in \{0,1\}^\ell$. Transform every fresh bit sampled at coordinate j by XORing it with $\Delta[j]$, including the bits of the initial key. Copying a coordinate from an earlier key commutes with this transformation, so every generated key becomes $K_i \oplus \Delta$. At the same time, replace R with

$$\mathsf{R}_\Delta(k, x) := \mathsf{R}(k \oplus \Delta, x),$$

while leaving H and the adversary's coins unchanged. Fresh bits remain independent and uniform, and R_Δ is again a uniform random function independent of H. Thus the transformed experiment has the same distribution as the original one.

Moreover, its visible transcript is identical, point by point. Public hash answers are unchanged, and reset-oracle answers agree since

$$\mathsf{R}_\Delta(K_i \oplus \Delta, x) = \mathsf{R}(K_i, x).$$

Inductively, the adversary makes the same queries and reset choices, receives the same rejections, and stops at the same time. Consequently, conditional on any visible ideal transcript and on K_i existing, K_i is uniform in $\{0,1\}^\ell$: the transformations act transitively on its possible values while preserving that transcript.

This statement holds for the complete transcript, including public queries made before or after K_i is generated.

Public queries rarely reach a hidden key. Let Hit be the event that some public query to H has length at least ℓ and its first ℓ bits equal some K_i. Conditional on a complete ideal transcript, all public query strings and N are fixed. The preceding symmetry gives probability $2^{-\ell}$ for each key–query pair to match. A union bound, followed by averaging over transcripts, yields

$$\Pr_{\mathsf{Ideal}}[\mathsf{Hit}] \leq \frac{q_F q_H}{2^\ell}.$$

No conditioning on the absence of earlier hits is needed. In particular, the bound covers a public query that guesses a key before that key is generated.

Coupling the two experiments. Implement both experiments by lazy sampling, using the same adversary coins and fresh key bits. In the ideal experiment, keep separate tables for $\mathsf{H}(s)$ and $\mathsf{R}(k, x)$; in the real experiment, use one table indexed by $s = k\|x$. Until an input has been queried through both interfaces, couple each newly sampled value across experiments. Repeated queries within either interface use the same stored value, including when different reset histories produce the same key. Invalid reset queries are rejected identically.

The coupled transcripts therefore agree until a public hash query and a valid reset query address the same string $k\|x$, regardless of which query came first. This cross-interface intersection implies Hit in the ideal experiment. The probability that the coupled outputs differ, and hence the distinguishing advantage, is at most $\Pr_{\mathsf{Ideal}}[\mathsf{Hit}]$. This proves the bound. For polynomial q_F, q_H and $\ell \geq \lambda$, it is negligible.

The argument applies unchanged to a random oracle with any polynomial output length. More generally, the bound is negligible whenever $\ell = \omega(\log \lambda)$.

5.2 Proof of Theorem 5.2

The ordinary PRG adds no existential assumption: $\mathsf{sRK\text{-}PRF}$ security implies ordinary PRF security under whole-key copying, which implies one-way functions and hence pseudorandom generators. We specify a λ-to-2λ generator as an explicit ingredient to count its cost.

For each public input $(\lambda, \mathsf{pp}, \mathsf{x})$, let $m = m(\lambda, \mathsf{pp}, \mathsf{x}) \geq 1$ be a publicly computable polynomial bound on the prover's random tape length, for every valid witness. Unused bits are ignored. We use a standard PRG $\mathsf{G} : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$. To expand a seed s, parse $\mathsf{G}(s_j) = s_{j+1}\|z_j$ into two λ-bit strings, starting with $s_0 = s$, and concatenate the output blocks $z_0, z_1, \ldots$. Define $\mathsf{Expand}_{\mathsf{G}}(s, m)$ as the first m bits of this stream. It is prefix-consistent, and for every polynomial M, its first M bits are pseudorandom by a hybrid over $\lceil M/\lambda \rceil$ calls to G. For a proof query set $u = \langle \mathtt{rnd}, \mathsf{pp}, \mathsf{x}, \mathsf{w}, m \rangle$, using an injective, efficiently decodable encoding; the tag $\mathtt{rnd}$ is reserved for this compiler.

The tape bound is a worst-case requirement; it excludes unbounded rejection sampling. A capped sampler fits this requirement, but its overflow behavior

must be included in the underlying prover. Aborting on overflow adds the overflow probability to the completeness error, so a negligible-error implementation requires a negligible tail bound uniform over valid witnesses.

Write $\mathsf{Adv}^{\mathsf{PRG},M}_{\mathcal{D},\mathsf{G}}$ for the advantage of distinguishing $\mathsf{Expand}_{\mathsf{G}}(U_\lambda, M)$ from U_M. The blockwise expansion above reduces this advantage to at most $\lceil M/\lambda \rceil$ advantages against the ordinary generator G.

For a ROM NIZK, we use the independent proof and randomness domains specified in Section 2. A reduction forwards proof-domain queries to the underlying NIZK experiment and samples the randomness domain independently. For a standard-model NIZK, omit the proof oracle.

Proof (Proof of Theorem 5.2). Let $\mathcal{A}$ make at most q valid proof queries. Let Q be a polynomial upper bound on the number of distinct pairs (r, u) arising in these queries, and let M be a polynomial upper bound on their tape lengths. We use three transitions.

First, replace the sRK-PRF by its ideal random function. A reduction $\mathcal{C}$ runs setup and $\mathcal{A}$, and makes one sRK-PRF query $(u, (\mathsf{idx}, I))$ for each valid proof query $(\mathsf{x}, \mathsf{w}, \mathsf{idx}, I)$. The oracle's $\mathsf{RandGen}$ state is exactly the compiled prover's seed state, since each valid proof query makes one call and invalid pairs are rejected before either state advances. If the reset request is invalid, the reduction returns $\perp$ without expanding a seed. Otherwise, the real oracle gives the compiled prover seed $\mathsf{F}_r(u)$; its ideal oracle gives $\mathsf{R}(r, u)$ for a random function R. Repeated pairs receive the same seed in both experiments. This transition costs at most $\mathsf{Adv}^{\mathsf{sRK\text{-}PRF}}_{\mathcal{C},\mathcal{F}}(\lambda)$, with q oracle queries.

Next, replace the expanded tapes one distinct pair (r, u) at a time. In the first hybrid, each such pair uses $\mathsf{Expand}_{\mathsf{G}}(\mathsf{R}(r, u), M)$; repeated pairs reuse the same stream. In the final hybrid, each distinct pair instead uses an independent uniform M-bit string. A distinguisher for the ith transition receives one M-bit PRG challenge and runs $\mathcal{A}$. It samples reset seeds and the random-function table R itself. On the ith new pair, it returns the first m bits of its challenge, where m is determined by that query. For every other new pair, it samples a fresh uniform PRG seed and expands it, or samples a uniform M-bit string, according to the hybrid; it caches the resulting tape and proof for repeated pairs. The distinguisher's challenge length M is fixed before the experiment, so the proof applies even when query lengths are chosen adaptively. Therefore this transition is bounded by $\sum_{i=1}^{Q} \mathsf{Adv}^{\mathsf{PRG},M}_{\mathcal{D}_i,\mathsf{G}}(\lambda)$. Prefix consistency makes every returned tape the correct prefix. With the block expansion above, each M-bit PRG advantage is at most $\lceil M/\lambda \rceil$ ordinary PRG advantages by the standard hybrid.

Finally, replace the fresh uniform-tape proofs by simulated proofs. A reduction $\mathcal{B}$ runs $\mathcal{A}$, samples its own reset seeds, and caches under $(r, \mathsf{pp}, \mathsf{x}, \mathsf{w})$, the same key used by the ideal compiled experiment. It rejects invalid pairs before advancing $\mathsf{RandGen}$. On a cache miss, it queries its underlying NIZK challenger on (x, w) and stores the response; on a hit, it returns the stored proof. In the real NIZK experiment, each cache miss uses an independent uniform tape, as in the preceding hybrid. In the simulated experiment, use the underlying setup simulator $\mathcal{S}_0$ and run $\mathcal{S}_1(\mathsf{pp}, \mathsf{x}, \mathsf{st})$ only on cache misses, updating its state there. On

hits, leave the simulator state unchanged; the compiled ideal oracle returns its cached proof. This gives the required ideal distribution, and the NIZK transition costs at most $\mathsf{Adv}^{\mathsf{NIZK}}_{\mathcal{B},\Pi}(\lambda)$. The simulator ignores the reset schedule.

Combining the transitions gives

$$\mathsf{Adv}^{\mathsf{srNIZK}}_{\mathcal{A},\tilde{\Pi}}(\lambda) \le \mathsf{Adv}^{\mathsf{sRK\text{-}PRF}}_{\mathcal{C},\mathcal{F}}(\lambda) + \sum_{i=1}^{Q} \mathsf{Adv}^{\mathsf{PRG},M}_{\mathcal{D}_i,\mathsf{G}}(\lambda) + \mathsf{Adv}^{\mathsf{NIZK}}_{\mathcal{B},\Pi}(\lambda).$$

All terms are negligible. Setup and verification are unchanged, so soundness is preserved whenever the underlying argument has it. Perfect completeness is also preserved: every expanded tape is an admissible tape for the original prover. The compiler's simulator is schedule-oblivious because its proof-generation step uses only the statement and its state on cache misses.

Proving cost. Let $T_H(|u| + \ell)$ be the cost of hashing the seed and encoded context, and let $T_G(m)$ be the cost of the $\lceil m/\lambda \rceil$ generator calls. If T_P is the original proving cost, the relative overhead in the ROM is

$$\frac{T_H(|u| + \ell) + T_G(m)}{T_P}.$$

For implementations whose hashing and expansion costs are linear in their input and output lengths, the added work is linear in $|u|+\ell+m$, with the corresponding primitive costs. The relative overhead tends to zero when that work is lower-order than T_P; this is an asymptotic cost comparison, not a cryptographic negligibility bound. This is a cost condition on the underlying prover and the chosen hash and PRG, rather than a consequence of reset security alone.

The context u contains pp, which for preprocessing arguments includes the proving key. Since pp is fixed throughout each experiment, the proof above is unchanged when u carries, in place of pp, any fixed-length string determined by pp. An implementation can therefore hash pp once with a collision-resistant function and use the digest in every proof; collision resistance keeps the contexts of different parameter sets distinct when one seed serves several of them. The per-proof input to the seed derivation then has length linear in $|\mathsf{x}| + |\mathsf{w}|$, which the prover reads in any case.

Corollary 5.4 (Random-oracle defense). *Let $\ell \ge \lambda$. Instantiate $\mathcal{F}$ by $\mathsf{F}_r(v) = \mathsf{H}_{\mathsf{rnd}}(r\|v)$, where $\mathsf{H}_{\mathsf{rnd}}$ outputs λ bits and is independent of the underlying proof oracle, and let G be a secure ordinary PRG. Then Figure 8 achieves srNIZK. For an adversary making at most q valid proof queries and q_H public queries to $\mathsf{H}_{\mathsf{rnd}}$, with at most Q distinct pairs (r, u) and tape lengths at most M,*

$$\mathsf{Adv}^{\mathit{srNIZK}}_{\mathcal{A},\tilde{\Pi}}(\lambda) \le q q_H\, 2^{-\ell} + \sum_{i=1}^{Q} \mathsf{Adv}^{\mathsf{PRG},M}_{\mathcal{D}_i,\mathsf{G}}(\lambda) + \mathsf{Adv}^{\mathit{NIZK}}_{\mathcal{B},\Pi}(\lambda).$$

Proof. Apply Theorem 5.3 to the first transition above, with q reset-oracle queries. The independent proof oracle does not affect its XOR symmetry argument. The PRG and underlying NIZK terms are as in the proof of Theorem 5.2.

Corollary 5.5 (Ordinary-PRF defense). *If $\mathcal{F}$ is an ordinary secure PRF with $\ell \geq \lambda$ and G is a secure ordinary PRG, then Figure 8 achieves rNIZK under whole-tape resets. In particular, it is a standard-model construction when the underlying NIZK, PRF, and PRG are in the standard model.*

Proof. Under whole-tape resetting, every successful query either samples a fresh independent seed or copies an earlier seed in full. Assign an origin index to each fresh sampling event; copied entries retain that index. Let $n \leq q$ bound the number of origins. A standard adaptive multi-key hybrid replaces the PRF for each origin, one at a time, by an independent random function. Each transition is an ordinary PRF reduction: embed its challenge key at that origin, answer all copies through the same oracle, and sample the other origins' keys locally. After these replacements, the functions are independent by origin. The probability that two fresh origin seeds coincide is at most $\binom{n}{2}2^{-\ell}$. Outside this event, the origin-indexed random functions have exactly the distribution of a random function indexed by the actual seed. The derived λ-bit seeds are then independent for every new (r, u) pair, with repeats receiving the same value, so the PRG hybrid from Section 5.2 applies.

Combining these reductions gives ordinary PRF adversaries $\mathcal{C}_i$, PRG adversaries $\mathcal{D}_i$, and a NIZK adversary $\mathcal{B}$ satisfying

$$\mathsf{Adv}^{\mathsf{rNIZK}}_{\mathcal{A},\tilde{\Pi}}(\lambda) \leq \sum_{i=1}^{n} \mathsf{Adv}^{\mathsf{PRF}}_{\mathcal{C}_i,\mathcal{F}}(\lambda) + \binom{n}{2}2^{-\ell} + \sum_{i=1}^{Q} \mathsf{Adv}^{\mathsf{PRG},M}_{\mathcal{D}_i,\mathsf{G}}(\lambda) + \mathsf{Adv}^{\mathsf{NIZK}}_{\mathcal{B},\Pi}(\lambda).$$

Polynomially many negligible terms and the seed-collision term are negligible.

5.3 XOR-Related-Key PRFs

Partial-reset and XOR-related-key security give equivalent PRF notions. Bellare and Kohno [11] formalized related-key security; we use the PRF formulation [9] for the class

$$\Phi_{\oplus} = \{k \mapsto k \oplus \delta : \delta \in \{0,1\}^{\ell}\}.$$

For an ℓ-bit root key K, the XOR-related-key experiment answers queries (δ, x) with either $\mathsf{F}_{K\oplus\delta}(x)$ or $\mathsf{R}(K\oplus\delta, x)$, where R is one random function. Queries are adaptive, and the ideal oracle uses the same value whenever the derived key and input repeat. We use this full related-key notion, with no restriction against repeating an input under different keys.

Our standard-model defense is conditional on a PRF with this full security notion and the domain and range of Definition 5.1. Abdalla, Benhamouda, and Passelègue [1] construct XOR-related-key PRFs with bit-string keys for fixed-length inputs under multilinear-map assumptions. Their result establishes feasibility under those assumptions; it does not supply a low-cost instantiation of our compiler. The restriction to fixed-length inputs is removed by hashing: if F is XOR-related-key secure on inputs of a fixed length n_0 and $\mathsf{h}_{\mathsf{hk}} : \{0,1\}^* \to \{0,1\}^{n_0}$ is collision-resistant for a public hash key hk included in the public parameters, then $(k, x) \mapsto \mathsf{F}_k(\mathsf{h}_{\mathsf{hk}}(x))$ is XOR-related-key secure on $\{0,1\}^*$. The

hash does not depend on the PRF key, so a reduction forwards each query (δ, x) as $(\delta, \mathsf{h}_{\mathsf{hk}}(x))$, and the two ideal experiments differ only if the adversary's queries contain a collision.

Lemma 5.6 (Partial resets and XOR-related keys). *Let $\ell = \ell(\lambda)$ be polynomial, and let $\mathcal{F}$ have ℓ-bit keys, domain $\{0,1\}^*$, and range $\{0,1\}^\lambda$. Then $\mathcal{F}$ is XOR-related-key secure if and only if it is an* sRK-PRF*. The reduction from partial resets to XOR-related keys preserves the number of oracle queries and the distinguishing advantage. The reverse reduction has polynomial query overhead and negligible additive loss.*

Section B proves the lemma.

Corollary 5.7 (Defense under XOR-related-key security). *If Π is a standard-model NIZK, $\mathcal{F}$ is an XOR-related-key-secure PRF with the domain and range of Definition 5.1, and* G *is a secure PRG, then Figure 8 achieves* srNIZK *in the standard model.*

Proof. Apply Lemma 5.6 and Theorem 5.2. For at most q proof queries, the XOR-related-key reduction makes at most q oracle queries, without additional distinguishing loss. The PRG and zero-knowledge transitions are those of Section 5.2.

6 Conclusion

We defined zero knowledge for non-interactive arguments under selective resets of the prover's randomness, gave witness-recovering attacks on Fiat–Shamir Σ-protocols, PLONK, Bulletproofs range proofs, and salted Fiat–Shamir compilations, and gave a compiler that turns any NIZK into an srNIZK by deriving the prover's tape with a PRF secure under partial key resets. We close with the limits of these results and the questions they leave open.

Limitations. The attacks on PLONK and on Bulletproofs range proofs are stated for provers that sample their masks independently from fixed positions of the tape. Realizing the reset interface in a deployed prover, or attacking a prover that derives its masks from a transcript-based generator, requires a separate analysis (Section 4 and Section A.5). The general attack on PIOP-based arguments (Theorem 4.6) relies on the coefficient-oracle knowledge guarantee of Definition 2.9, with one strict-PPT extractor for every admissible prover. It does not cover extractors that run in expected polynomial time or whose error depends on the prover's acceptance probability. Finally, we do not relate our definition to classical resettable zero knowledge for interactive protocols (Section 3.1).

Open Problems. Our standard-model defense assumes a PRF with full XOR-related-key security (Section 5.3). Such PRFs exist under multilinear-map assumptions [1]; this establishes feasibility but not an efficient instantiation. A

construction from standard assumptions, or a compiler that needs a weaker primitive, would give a practical defense against selective resets without random oracles. A second question is whether an sRK-PRF also gives public-key encryption that remains secure under selective resets of the encryption randomness, using the techniques of resettable and related-randomness encryption [58,73]. A third is experimental: our model abstracts faults and incomplete state restoration as resets of chosen tape positions. Carrying out such resets against deployed provers, for example by fault injection with bit-level control, would show how closely the model matches practice.

Acknowledgments. Matteo Campanelli was supported by the Estonian Research Council grant PRG2531.

References

1. Abdalla, M., Benhamouda, F., Passelègue, A.: Algebraic XOR-RKA-secure pseudorandom functions from post-zeroizing multilinear maps. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part II. LNCS, vol. 11922, pp. 386–412. Springer, Heidelberg (Dec 2019). https://doi.org/10.1007/978-3-030-34621-8_14
2. Aranha, D.F., Orlandi, C., Takahashi, A., Zaverucha, G.: Security of hedged Fiat-Shamir signatures under fault attacks. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part I. LNCS, vol. 12105, pp. 644–674. Springer, Heidelberg (May 2020). https://doi.org/10.1007/978-3-030-45721-1_23
3. Arun, A., Setty, S., Thaler, J.: Jolt: SNARKs for virtual machines via lookups. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 3–33. Springer (2024)
4. Attema, T., Fehr, S., Klooß, M.: Fiat-shamir transformation of multi-round interactive proofs (extended version). Journal of Cryptology **36**(4), 36 (Oct 2023). https://doi.org/10.1007/s00145-023-09478-y
5. Austrin, P., Chung, K.M., Mahmoody, M., Pass, R., Seth, K.: On the impossibility of cryptography with tamperable randomness. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 462–479. Springer, Heidelberg (Aug 2014). https://doi.org/10.1007/978-3-662-44371-2_26
6. Bar-El, H., Choukri, H., Naccache, D., Tunstall, M., Whelan, C.: The sorcerer's apprentice guide to fault attacks. Cryptology ePrint Archive, Report 2004/100 (2004), https://eprint.iacr.org/2004/100
7. Barak, B., Goldreich, O., Goldwasser, S., Lindell, Y.: Resettably-sound zero-knowledge and its applications. In: Proceedings 42nd IEEE Symposium on Foundations of Computer Science. pp. 116–125. IEEE (2001)
8. Bellare, M., Brakerski, Z., Naor, M., Ristenpart, T., Segev, G., Shacham, H., Yilek, S.: Hedged public-key encryption: How to protect against bad randomness. In: Advances in Cryptology–ASIACRYPT 2009: 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings. pp. 232–249. Springer (2009)
9. Bellare, M., Cash, D.: Pseudorandom functions and permutations provably secure against related-key attacks. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 666–684. Springer, Heidelberg (Aug 2010). https://doi.org/10.1007/978-3-642-14623-7_36

10. Bellare, M., Fischlin, M., Goldwasser, S., Micali, S.: Identification protocols secure against reset attacks. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 495–511. Springer, Heidelberg (May 2001). https://doi.org/10.1007/3-540-44987-6_30
11. Bellare, M., Kohno, T.: A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications. In: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 491–506. Springer (2003)
12. Bellare, M., Micciancio, D., Warinschi, B.: Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In: Advances in Cryptology—EUROCRYPT 2003: International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4–8, 2003 Proceedings. pp. 614–629. Springer (2003)
13. Bellare, M., Tackmann, B.: Nonce-based cryptography: Retaining security when randomness fails. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016, Part I. LNCS, vol. 9665, pp. 729–757. Springer, Heidelberg (May 2016). https://doi.org/10.1007/978-3-662-49890-3_28
14. Ben-Sasson, E., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., Virza, M.: Zerocash: Decentralized anonymous payments from bitcoin. In: 2014 IEEE symposium on security and privacy. pp. 459–474. IEEE (2014)
15. Bitansky, N., Choudhuri, A.R.: Characterizing deterministic-prover zero knowledge. In: Pass, R., Pietrzak, K. (eds.) TCC 2020, Part I. LNCS, vol. 12550, pp. 535–566. Springer, Heidelberg (Nov 2020). https://doi.org/10.1007/978-3-030-64375-1_19
16. Boneh, D., DeMillo, R.A., Lipton, R.J.: On the importance of eliminating errors in cryptographic computations. Journal of cryptology **14**, 101–119 (2001)
17. Breitner, J., Heninger, N.: Biased nonce sense: Lattice attacks against weak ECDSA signatures in cryptocurrencies. In: Goldberg, I., Moore, T. (eds.) FC 2019. LNCS, vol. 11598, pp. 3–20. Springer, Heidelberg (Feb 2019). https://doi.org/10.1007/978-3-030-32101-7_1
18. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: Short proofs for confidential transactions and more. Cryptology ePrint Archive, Report 2017/1066 (2017), https://eprint.iacr.org/2017/1066
19. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: Short proofs for confidential transactions and more. In: 2018 IEEE Symposium on Security and Privacy. pp. 315–334. IEEE Computer Society Press (May 2018). https://doi.org/10.1109/SP.2018.00020
20. Bünz, B., Fisch, B., Szepieniec, A.: Transparent SNARKs from DARK compilers. In: Advances in Cryptology–EUROCRYPT 2020: 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10–14, 2020, Proceedings, Part I. pp. 677–706. Springer (2020)
21. Canetti, R., Goldreich, O., Goldwasser, S., Micali, S.: Resettable zero-knowledge. In: Proceedings of the thirty-second annual ACM symposium on Theory of computing. pp. 235–244 (2000)
22. Chiesa, A., Hu, Y., Maller, M., Mishra, P., Vesely, N., Ward, N.: Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In: Advances in Cryptology–EUROCRYPT 2020: 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10–14, 2020, Proceedings, Part I. pp. 738–768. Springer (2020)
23. Chiesa, A., Yogev, E.: Building cryptographic proofs from hash functions (2024), https://github.com/hash-based-snargs-book

24. Cho, C., Ostrovsky, R., Scafuro, A., Visconti, I.: Simultaneously resettable arguments of knowledge. In: Theory of Cryptography Conference. pp. 530–547. Springer (2012)
25. Consensys: gnark PLONK prover for BN254, https://github.com/Consensys/gnark/blob/19e21741d0e71f12767adae8285080360b31ace8/backend/plonk/bn254/prove.go, source code, commit 19e21741; accessed September 25, 2026
26. Dahari, H., Lindell, Y.: Deterministic-prover zero-knowledge proofs. Cryptology ePrint Archive, Report 2020/141 (2020), https://eprint.iacr.org/2020/141
27. dalek-cryptography: Bulletproofs R1CS prover, https://github.com/dalek-cryptography/bulletproofs/blob/be67b6d5f5ad1c1f54d5511b52e6d645a1313d07/src/r1cs/prover.rs, source code, commit be67b6d5; accessed September 25, 2026
28. dalek-cryptography: Bulletproofs range prover, https://github.com/dalek-cryptography/bulletproofs/blob/be67b6d5f5ad1c1f54d5511b52e6d645a1313d07/src/range_proof/party.rs, source code, commit be67b6d5; accessed September 25, 2026
29. De Santis, A., Di Crescenzo, G., Ostrovsky, R., Persiano, G., Sahai, A.: Robust non-interactive zero knowledge. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 566–598. Springer, Heidelberg (Aug 2001). https://doi.org/10.1007/3-540-44647-8_33
30. Deng, Y., Feng, D., Goyal, V., Lin, D., Sahai, A., Yung, M.: Resettable cryptography in constant rounds–the case of zero knowledge. In: Advances in Cryptology–ASIACRYPT 2011: 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings. pp. 390–406. Springer (2011)
31. Deng, Y., Goyal, V., Sahai, A.: Resolving the simultaneous resettability conjecture and a new non-black-box simulation strategy. In: 2009 50th Annual IEEE Symposium on Foundations of Computer Science. pp. 251–260. IEEE (2009)
32. Deng, Y., Lin, D.: Resettable zero knowledge with concurrent soundness in the bare public-key model under standard assumption. In: International Conference on Information Security and Cryptology. pp. 123–137. Springer (2007)
33. Di Crescenzo, G., Persiano, G., Visconti, I.: Constant-round resettable zero knowledge with concurrent soundness in the bare public-key model. In: Annual International Cryptology Conference. pp. 237–253. Springer (2004)
34. Dusk Network: Add blinding scalars. Issue 650 of the dusk-network/plonk repository (Dec 2021), https://github.com/dusk-network/plonk/issues/650
35. Dwork, C., Naor, M.: Zaps and their applications. In: Proceedings 41st Annual Symposium on Foundations of Computer Science. pp. 283–293. IEEE (2000)
36. Dwork, C., Naor, M., Sahai, A.: Concurrent zero-knowledge. Journal of the ACM (JACM) **51**(6), 851–898 (2004)
37. Gabizon, A., Williamson, Z.J., Ciobotaru, O.: PLONK: Permutations over Lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Paper 2019/953 (2019), https://eprint.iacr.org/2019/953
38. Garg, S., Jain, A., Sahai, A.: Leakage-resilient zero knowledge. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 297–315. Springer, Heidelberg (Aug 2011). https://doi.org/10.1007/978-3-642-22792-9_17
39. Garg, S., Ostrovsky, R., Visconti, I., Wadia, A.: Resettable statistical zero knowledge. In: Theory of Cryptography Conference. pp. 494–511. Springer (2012)
40. Golovnev, A., Lee, J., Setty, S.T.V., Thaler, J., Wahby, R.S.: Brakedown: Linear-time and field-agnostic SNARKs for R1CS. In: Handschuh, H., Lysyanskaya, A.

(eds.) CRYPTO 2023, Part II. LNCS, vol. 14082, pp. 193–226. Springer, Heidelberg (Aug 2023). https://doi.org/10.1007/978-3-031-38545-2_7
41. Goyal, V., Song, Y.: Correlated-source extractors and cryptography with correlated-random tapes. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part I. LNCS, vol. 11476, pp. 562–592. Springer, Heidelberg (May 2019). https://doi.org/10.1007/978-3-030-17653-2_19
42. Josefsson, S., Liusvaara, I.: Edwards-curve digital signature algorithm (EdDSA). RFC 8032 (2017). https://doi.org/10.17487/RFC8032, https://www.rfc-editor.org/rfc/rfc8032
43. Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-size commitments to polynomials and their applications. In: Advances in Cryptology-ASIACRYPT 2010: 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings. pp. 177–194. Springer (2010)
44. Kiyoshima, S.: Resettable statistical zero-knowledge for NP. In: Annual International Cryptology Conference. pp. 288–320. Springer (2024)
45. Kohlweiss, M., Pancholi, M., Takahashi, A.: How to compile polynomial IOP into simulation-extractable SNARKs: a modular approach. In: Theory of Cryptography Conference. pp. 486–512. Springer (2023)
46. LambdaClass: First Lambda-Ingo ZK CTF: ZK challenges using LambdaWorks. Challenge write-up (Jul 2023), https://blog.lambdaclass.com/first-lambda-ingo-zk-ctf-zk-challenges-using-lambdaworks/
47. Liang, J., Zhang, X., Hu, D., Shen, Q., Fang, Y., Wu, Z.: ZK-Hammer: Leaking secrets from zero-knowledge proofs via Rowhammer. In: 62nd ACM/IEEE Design Automation Conference (DAC). pp. 1–7. IEEE (2025). https://doi.org/10.1109/DAC63849.2025.11133021
48. Maller, M., Bowe, S., Kohlweiss, M., Meiklejohn, S.: Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) ACM CCS 2019. pp. 2111–2128. ACM Press (Nov 2019). https://doi.org/10.1145/3319535.3339817
49. Maurer, U.: Unifying zero-knowledge proofs of knowledge. In: Preneel, B. (ed.) Advances in Cryptology – AfricaCrypt 2009. Lecture Notes in Computer Science, Springer-Verlag (Jun 2009), https://crypto.ethz.ch/publications/Maurer09.html
50. Micali, S.: CS proofs (extended abstracts). In: 35th FOCS. pp. 436–453. IEEE Computer Society Press (Nov 1994). https://doi.org/10.1109/SFCS.1994.365746
51. Micali, S., Reyzin, L.: Min-round resettable zero-knowledge in the public-key model. In: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 373–393. Springer (2001)
52. Micali, S., Reyzin, L.: Soundness in the public-key model. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 542–565. Springer, Heidelberg (Aug 2001). https://doi.org/10.1007/3-540-44647-8_32
53. Mondal, P., Adhikary, S., Kundu, S., Karmakar, A.: ZKFault: Fault attack analysis on zero-knowledge based post-quantum digital signature schemes. In: ASIACRYPT 2024, Part VIII. pp. 132–167. LNCS, Springer (2024). https://doi.org/10.1007/978-981-96-0944-4_5
54. Nick, J., Ruffing, T., Seurin, Y., Wuille, P.: MuSig-DN: Schnorr multi-signatures with verifiably deterministic nonces. In: Ligatti, J., Ou, X., Katz, J., Vigna, G. (eds.) ACM CCS 2020. pp. 1717–1731. ACM Press (Nov 2020). https://doi.org/10.1145/3372297.3417236

55. Okamoto, T.: Provably secure and practical identification schemes and corresponding signature schemes. In: Annual international cryptology conference. pp. 31–53. Springer (1992)
56. OpenZeppelin: Linea Plonk prover (backend) and Plonk verifier audit. Security audit report (Jun 2024), https://www.openzeppelin.com/news/linea-prover-audit
57. Otto, M.: Fault attacks and countermeasures. Ph.D. thesis, University of Paderborn, Germany (2005)
58. Paterson, K.G., Schuldt, J.C., Sibborn, D.L.: Related randomness attacks for public key encryption. In: Public-Key Cryptography–PKC 2014: 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014. Proceedings. pp. 465–482. Springer (2014)
59. Pornin, T.: Deterministic usage of the digital signature algorithm (DSA) and elliptic curve digital signature algorithm (ECDSA). RFC 6979 (2013). https://doi.org/10.17487/RFC6979, https://www.rfc-editor.org/rfc/rfc6979
60. Ristenpart, T., Yilek, S.: When good randomness goes bad: Virtual machine reset vulnerabilities and hedging deployed cryptography. In: NDSS (2010)
61. Sahai, A.: Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In: 40th FOCS. pp. 543–553. IEEE Computer Society Press (Oct 1999). https://doi.org/10.1109/SFFCS.1999.814628
62. Sahai, A., Waters, B.: How to use indistinguishability obfuscation: deniable encryption, and more. In: Shmoys, D.B. (ed.) 46th ACM STOC. pp. 475–484. ACM Press (May / Jun 2014). https://doi.org/10.1145/2591796.2591825
63. Schnorr, C.P.: Efficient identification and signatures for smart cards. In: Brassard, G. (ed.) CRYPTO'89. LNCS, vol. 435, pp. 239–252. Springer, Heidelberg (Aug 1990). https://doi.org/10.1007/0-387-34805-0_22
64. Sefranek, M.: How (not) to simulate PLONK. Cryptology ePrint Archive, Paper 2024/848 (2024), https://eprint.iacr.org/2024/848, conference version in SCN 2024, LNCS 14973, pp. 96–117
65. Setty, S.: Spartan: Efficient and general-purpose zkSNARKs without trusted setup. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part III. LNCS, vol. 12172, pp. 704–737. Springer, Heidelberg (Aug 2020). https://doi.org/10.1007/978-3-030-56877-1_25
66. Setty, S., Thaler, J., Wahby, R.: Unlocking the lookup singularity with lasso. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 180–209. Springer (2024)
67. Silur: Zk11: Zk in restricted environments - can state-of-the-art systems run on smartcards? https://www.youtube.com/watch?v=s2BVwde--AM (April 2024), accessed: 2024-05-24
68. de Valence, H.: Merlin transcripts: Generating randomness. https://merlin.cool/transcript/rng.html, accessed September 24, 2026
69. de Valence, H.: Merlin: Flexible, composable transcripts for zero-knowledge proofs. https://medium.com/@hdevalence/merlin-flexible-composable-transcripts-for-zero-knowledge-proofs-28d9fda22d9a (Sep 2018)
70. Wuille, P., Nick, J., Ruffing, T.: Schnorr signatures for secp256k1. Bitcoin Improvement Proposal 340 (2020), https://github.com/bitcoin/bips/blob/master/bip-0340.mediawiki
71. Xie, T., Zhang, Y., Song, D.: Orion: Zero knowledge proof with linear prover time. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part IV. LNCS, vol. 13510, pp. 299–328. Springer, Heidelberg (Aug 2022). https://doi.org/10.1007/978-3-031-15985-5_11

72. Yang, G., Duan, S., Wong, D.S., Tan, C.H., Wang, H.: Authenticated key exchange under bad randomness. In: Financial Cryptography and Data Security: 15th International Conference, FC 2011, Gros Islet, St. Lucia, February 28-March 4, 2011, Revised Selected Papers 15. pp. 113–126. Springer (2012)
73. Yilek, S.: Resettable public-key encryption: How to encrypt on a virtual machine. In: Cryptographers' Track at the RSA Conference. pp. 41–56. Springer (2010)
74. Yung, M., Zhao, Y.: Generic and practical resettable zero-knowledge in the bare public-key model. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 129–147. Springer (2007)

A Proofs of the Resetting Attacks

A.1 Analysis of the Resetting Attack Against Maurer's Protocol

We first analyze witness recovery and use it to prove Theorem 4.1. We then give the field-challenge Okamoto instance and the extension to externally fixed one-way homomorphisms.

Definition A.1 (One-Way Group Homomorphism). *We define a* group-and-homomorphism generator *as an efficient algorithm* $\mathsf{GroupHomGen}$ *that, on input the security parameter* 1^λ*, outputs the description of two groups* $(\mathbb{G}, \star)$ *and* $(\mathbb{H}, \otimes)$*, whose elements are efficiently representable and operations efficiently computable, along with an efficiently computable group homomorphism* $\phi : \mathbb{G} \to \mathbb{H}$*. We assume that the groups are finite and admit efficient uniform sampling.*

We say that $\mathsf{GroupHomGen}$ *is* (average-case) one-way *if, for all PPT adversaries* $\mathcal{A}$*,*

$$\Pr\left[\phi(\mathsf{w}) = \mathsf{x} \;\middle|\; \begin{array}{l}(\mathbb{G}, \mathbb{H}, \phi) \leftarrow \mathsf{GroupHomGen}(1^\lambda) \\ \mathsf{w}_0 \leftarrow \mathbb{G}, \quad \mathsf{x} := \phi(\mathsf{w}_0) \\ \mathsf{w} \leftarrow \mathcal{A}(\mathbb{G}, \mathbb{H}, \phi, \mathsf{x})\end{array}\right] \leq \mathsf{negl}(\lambda).$$

Direct Witness Recovery. Let $\mathcal{C} = \{0, 1\}$ and $\kappa \geq \lambda$. The target is sampled from a valid pair for the public homomorphism; when setup determines the relation, this sampling is conditional on pp. If $x = e_{\mathbb{H}}$, return $e_{\mathbb{G}}$. Otherwise, make exactly the following two oracle calls:

$$\mathcal{O}_{\mathcal{P}}(e_{\mathbb{H}}, e_{\mathbb{G}}, 1, \varnothing), \qquad \mathcal{I}_{\mathcal{P}}(1, [\ell]).$$

The first call creates tape 1 and returns $z_i' = r_i \star e_{\mathbb{G}}^{c_i'} = r_i$. The second call reuses that tape and returns $z_i = r_i \star w^{c_i}$. If any target challenge is 1, return $(z_i')^{-1} \star z_i = w$. The attack makes no other hash queries before these proofs; since $x \neq e_{\mathbb{H}}$, the target hash input is distinct from the auxiliary input and its challenge vector is uniform. The only failure event is the all-zero target vector, of probability $2^{-\kappa}$.

The extraction algebra also applies to arbitrary accepted transcripts with a common first message $a_i = a_i'$ and target challenge $c_i = 1$. Acceptance of the

identity proof gives $\phi(z'_i) = a_i$, while acceptance of the target gives $\phi(z_i) = a_i \otimes x$. Consequently,

$$\phi((z'_i)^{-1} \star z_i) = a_i^{-1} \otimes a_i \otimes x = x.$$

Thus accepting transcripts with a common first message suffice to extract a valid witness. In our privacy attack, tape reuse gives this common first message because the prover follows the specified honest algorithm. An arbitrary malicious prover may instead choose its first message as a function of the statement or refuse to answer. For $x \neq e_{\mathbb{H}}$, the recovered witness is exactly the one used; for $x = e_{\mathbb{H}}$, returning the identity only guarantees a valid witness.

Nontrivial Kernels under Reset-Aware Simulation. We now prove Theorem 4.1. For fixed parameters write $M = |\operatorname{im}\phi|$ and $K = |\ker\phi|$, and assume $M, K \geq 2$. The adversary need not compute these cardinalities. In the setup-generated case, assume an efficient public format check that accepts every honest output and whose accepted descriptions define finite groups with efficient uniform sampling and an efficient homomorphism having $M, K \geq 2$. The distinguisher rejects a failed check. For externally fixed parameters, it simply checks that setup returned the prescribed parameters. In either case, every accepted parameter choice has at least two preimages for each element of the image.

Proof (Proof of Theorem 4.1). After receiving the parameters, the distinguisher samples independent uniform $W \leftarrow \mathbb{G}$ until $X = \phi(W) \neq e_{\mathbb{H}}$, stopping after λ trials and outputting 0 if all trials fail. It then requests an identity proof with witness $e_{\mathbb{G}}$ on a fresh tape and a proof for (X, W) using that same tape. It outputs 1 if a target challenge $c_i = 1$ and the extracted value $(z'_i)^{-1} \star z_i$ equals its privately sampled W; otherwise it outputs 0.

A uniform group element maps to a uniform image, so all sampling trials fail with probability $M^{-\lambda} \leq 2^{-\lambda}$. On success, the two statements differ. The target hash input is therefore fresh in the real execution, and its challenge vector is uniform. The direct extraction above gives

$$\Pr[D_{\mathsf{real}} = 1] = (1 - M^{-\lambda})(1 - 2^{-\kappa}) \geq 1 - 2^{-\lambda} - 2^{-\kappa}.$$

For setup-generated parameters, the equality holds conditioned on each honest parameter choice and the lower bound holds after averaging.

In the ideal execution, condition on accepted public parameters, the simulator's setup state, and the sampled nonidentity X. Rejection sampling preserves the uniform distribution of W on $\phi^{-1}(X)$, a coset with exactly K elements. The auxiliary query is fixed, and the target query is a cache miss because its statement differs. Both handles and all reset metadata are independent of W given X. Consequently, the simulator's entire view, including its random-oracle programming, gives no information about which element of this fiber was sampled. Any extracted candidate equals W with probability at most $1/K \leq 1/2$. Rejected parameters or exhausted sampling trials contribute zero to ideal success. Thus

$$\mathsf{Adv}^{\mathsf{rNIZK}}_{D,\Pi_{\mathsf{hom}}} \geq \frac{1}{2} - 2^{-\lambda} - 2^{-\kappa}.$$

For fixed parameters the sharper bound is $(1 - M^{-\lambda})(1 - 2^{-\kappa} - 1/K)$. Only the two displayed proof queries are needed; rejection sampling is local computation. Neither the argument nor its bound requires one-wayness.

The Field-Challenge Okamoto Instance.

Corollary A.2. *For fixed cyclic-group parameters of prime order p and nonidentity generators g, h, Okamoto's Fiat–Shamir protocol with challenges in $\mathbb{F}_p$ admits a two-query reset-aware distinguishing attack of advantage at least $1-3/p$.*

Fix a cyclic group of prime order p and nonidentity generators g, h, with $1/p$ negligible. For the Okamoto instance, $\mathbb{G} = \mathbb{F}_p^2$, $\phi(u, v) = g^u h^v$, and the challenge lies in $\mathbb{F}_p$, as in Section 1.2. The parameters are fixed common inputs; the simulator may retain arbitrary setup state, including $\log_g h$. We allow it to observe the reset schedule, but not the witness or the adversary's private coins. It answers oracle calls online, as in Figure 3.

The distinguisher rejects any setup output inconsistent with these fixed parameters. It samples $W = (U, V) \leftarrow \mathbb{F}_p^2$, sets $X = g^U h^V$, and outputs 0 if $X = 1$. Otherwise, it requests a fresh proof for $(1, (0, 0))$ and a proof for (X, W) using the same tape. Write their responses as (z_0', z_1') and (z_0, z_1). It checks the transcripts and their common first message. If the target challenge c is nonzero, it computes

$$\widehat{W} = \big(c^{-1}(z_0 - z_0'),\ c^{-1}(z_1 - z_1')\big)$$

and outputs 1 exactly when $\widehat{W} = W$. Malformed transcripts, a failed check, or $c = 0$ cause output 0.

In the real experiment, the auxiliary responses equal the two masks. Since $X \neq 1$, the target hash input is fresh and c is uniform. Thus

$$\Pr[D_{\mathsf{real}} = 1] = (1 - 1/p)^2.$$

For any fixed $X \neq 1$, exactly p pairs satisfy $X = g^U h^V$, and W is uniform among them, even conditioned on the simulator's setup state. The auxiliary query is fixed, and the target query is a new cache entry because its statement differs. The simulator therefore receives no information about the particular preimage W beyond X. This remains true if it receives the reset pattern, independently sampled simulated tapes, or permission to program its random oracle. Its answer and the resulting $\widehat{W}$ identify the sampled pair with probability at most $1/p$. Hence

$$\Pr[D_{\mathsf{ideal}} = 1] \le 1/p, \qquad \mathsf{Adv}_D \ge (1 - 1/p)^2 - 1/p \ge 1 - 3/p.$$

The distinguisher samples W itself and supplies it to the relation-checking oracle, so both proof queries are legal. The distinguishing gap follows from recovering this particular witness among the p preimages of X.

Externally Fixed One-Way Homomorphisms.

Proposition A.3. *Let the homomorphism be an immutable external parameter sampled by a one-way* GroupHomGen*, with no generation trapdoor supplied to the simulator. For $\mathcal{C} = \{0,1\}$ and $\kappa \geq \lambda$, Maurer's protocol is not reset-aware* rNIZK*, including when the kernel is trivial.*

Proof. Run the direct witness-recovery attack and distinguish by testing whether extraction yields any valid witness. The distinguisher samples its target witness itself, so every oracle query is legal. Real success is at least $1 - 2^{-\kappa}$, including returning the identity witness on the identity target. Reset-aware simulation would give ideal success $1 - 2^{-\kappa} - \mathsf{negl}(\lambda)$.

This ideal attack gives an inverter for the external homomorphism. Given a sampled image x, return the identity if $x = e_{\mathbb{H}}$. Otherwise, run $\mathcal{S}_0$ on the immutable external parameters, generate the simulated tape, and invoke $\mathcal{S}_{\mathsf{step}}$ for the identity query and then the target query. Their statements differ, so both are cache misses; their handles and reset metadata can be supplied without knowing a preimage of x. Extract and check the returned candidate. This reproduces the ideal attack on the sampled-image distribution and contradicts one-wayness. For ROM simulators the inverter also runs their simulated public-oracle environment.

Three-Query Strong Resetting for Okamoto. We now analyze the partial-reset attack from the overview. By resetting each of the two scalar masks separately, we recover both witness coordinates from three proofs.

Lemma A.4. *Fix the Okamoto parameters above and a valid target pair independently of the proof-generation random oracle. Suppose the masks r_0, r_1 use disjoint, fixed tape blocks B_0, B_1. Starting with no prior queries in this oracle domain, three target proofs suffice to recover the honest witness with probability at least*

$$1 - 4/p + 2/p^2 - \epsilon_{\mathsf{samp}},$$

where ϵ_{samp} bounds the statistical distance of the joint scalar sampling from four independent uniform field elements.

Proof. Make exactly the queries $(1, \varnothing)$, $(1, B_0)$, and $(1, B_1)$ to the target oracle. The first creates tape 1 with masks (r_0, r_1). The second preserves r_0 and refreshes r_1; the third preserves the original r_1 and refreshes r_0. Write the resulting challenges as c, c', c'' and responses as $(z_0, z_1), (z_0', z_1'), (z_0'', z_1'')$. If both denominators are nonzero, output

$$w_0 = \frac{z_0' - z_0}{c' - c}, \qquad w_1 = \frac{z_1'' - z_1}{c'' - c}.$$

For either reset, the new commitment equals the first with probability $1/p$, since the refreshed mask is uniform and its generator has order p. Otherwise, the two distinct hash inputs have independent uniform challenges, which coincide with

probability $1/p$. Thus either denominator vanishes with probability $2/p - 1/p^2$. A union bound gives failure at most $4/p - 2/p^2$; no independence between the two failures is needed. Coupling the actual scalar sampler with this ideal execution adds at most ϵ_{samp}.

For example, each mask block can contain $m = \lceil \log_2 p \rceil + t$ uniform bits interpreted modulo p. Each scalar is within 2^{-t} of uniform, so the four independent draws above have $\epsilon_{\mathsf{samp}} \leq 4 \cdot 2^{-t}$. Taking $t = \lambda$ makes the error negligible while keeping the block positions fixed.

A.2 Proof of Theorem 4.2

The attack in Figure 6 proceeds in two steps. We first use the known auxiliary witnesses to remove the target's mask, then recover the target witness from the unmasked observations. For the first step, it suffices to determine the mask contribution at the target query.

Fix the setup-aware target sampler of Definition 3.5. Let $a \in \mathbb{F}^D$ encode its target witness, with an efficient decoder that recovers a valid witness from $(\mathsf{pp}, \mathsf{x}^*, a)$. In session i, write the exposed auxiliary and target evaluations as

$$y_{i,j} = b_{i,j} + L_{i,j}\theta_i, \qquad y_i^* = A_i a + B_i \theta_i + d_i.$$

Here $\theta_i \in \mathbb{F}^\mu$ is the common mask induced by tape reuse. The offsets $b_{i,j}$ are computable from the auxiliary witnesses and their transcripts; all matrices and d_i are publicly computable. Their dimensions, and the numbers n, k of sessions and auxiliary proofs per session, are polynomially bounded. Stack the auxiliary matrices and residuals as

$$L_i = \begin{pmatrix} L_{i,1} \\ \vdots \\ L_{i,k} \end{pmatrix}, \qquad v_i = \begin{pmatrix} y_{i,1} - b_{i,1} \\ \vdots \\ y_{i,k} - b_{i,k} \end{pmatrix} = L_i \theta_i.$$

Lemma A.5 (Determining the target mask). *For fixed matrices L, B, the value $B\theta$ is determined by $L\theta$ for every $\theta \in \mathbb{F}^\mu$ if and only if $\ker L \subseteq \ker B$. When this condition holds, Gaussian elimination finds a matrix C with $CL = B$.*

Proof. Two masks give the same auxiliary observations precisely when their difference belongs to $\ker L$. Their target contributions agree precisely when this difference belongs to $\ker B$. This proves the equivalence. The kernel inclusion is equivalent to $\mathrm{rowspan}(B) \subseteq \mathrm{rowspan}(L)$, so expressing each row of B in the row span of L gives C.

Proof (Proof of Theorem 4.2). The first proof in session i uses fresh randomness and creates tape $t_i = (i-1)(k+1)+1$. The remaining k queries copy that tape in full. Thus the k auxiliary proofs and target proof have the stated common mask θ_i.

For each session, compute v_i and a matrix C_i satisfying $C_i L_i = B_i$. Subtracting the resulting mask contribution gives

$$y_i^* - d_i - C_i v_i = A_i a.$$

Stack these equations across the n sessions. If the stacked matrix with blocks $A_1, \ldots, A_n$ has rank D, Gaussian elimination recovers a, and the decoder returns a valid target witness. The adversary verifies this witness before returning it. It makes exactly $n(k+1)$ proof queries and otherwise performs polynomial-time linear algebra.

Let ϵ bound the probability that a proof aborts, the displayed response equations or kernel inclusions fail, the stacked target matrix has insufficient rank, or decoding fails. On the complement of these events the attack succeeds, so its failure probability is at most ϵ. Any statistical error in implementing the stated mask distribution is added to this bound by coupling.

Fresh Evaluation Points. To obtain enough independent observations for recovery, we analyze the inputs to the random oracle. Suppose all challenges treated as fresh use previously unqueried, domain-separated inputs, except with probability ϵ_{ctx}. If each evaluation point has conditional maximum probability at most α, then among N such points a collision occurs with probability at most

$$\epsilon_{\mathsf{ctx}} + \binom{N}{2}\alpha.$$

For a fixed bad set B, the probability of any point lying in B is at most $\epsilon_{\mathsf{ctx}} + N|B|\alpha$. These bounds follow by exposing the oracle answers in order and taking a union bound. For uniform field points, $\alpha = 1/|\mathbb{F}|$. More general rank conditions can be checked by a nonzero determinant polynomial: if its total degree is Δ in independent uniform field challenges, the Schwartz–Zippel bound is $\Delta/|\mathbb{F}|$, with denominator zeros counted separately.

The PLONK analysis bounds both commitment collisions between distinct polynomials and evaluation-point collisions between distinct hash inputs, using the protocol's challenge transcript. For its linear wire masks, the two auxiliary evaluations determine all six masking coefficients, which implies Lemma A.5; the d target evaluations then determine the three wire polynomials, each of degree less than d.

A.3 Concrete PLONK Witness-Recovery Bounds

We prove Proposition 4.3, using the protocol, notation, and randomness layout of Section 4.1.3.

Transcript. After committing to P, the prover derives the quotient-combination challenge, constructs and commits to the quotient pieces, and derives the evaluation challenge ζ. The hash input for ζ contains the public statement and all these commitments, including $[P(s)]_1$. Later quotient-piece blinders are independent

of the six wire blinders. The permutation denominators contain $3d$ factors that are affine with nonzero coefficient in the uniform challenge γ; their vanishing probability is at most $3d/p$ per fresh challenge pair. We count such a prover abort as attack failure. We also count evaluation points in $H \cup \{s, s/\omega\}$ as bad, where $H = \langle\omega\rangle$, so the bounds do not rely on exceptional evaluation, opening, or verification cases.

Standard Resetting with Auxiliary Inputs. Fix a circuit and two known valid auxiliary assignments whose public statements are distinct from each other and from the target statement. The assignments are fixed independently of the proof-generation random oracle. Execute d sessions. In each session, request a fresh tape for the first auxiliary proof, reuse it for the second auxiliary proof, and reuse it again for the target proof, as in Figure 6. Thus each session uses three queries and its first tape has index $3(i-1)+1$.

Here z_j denotes the evaluation point derived from the random oracle for the j-th statement in the current session, and $A_j, \widehat{A}_j$ are the corresponding wire polynomials; the target statement is $j = 3$. For either auxiliary evaluation point $z_j \notin H$, compute

$$u_j = \frac{A_j(z_j) - \widehat{A}_j(z_j)}{Z_H(z_j)} = \rho_1 z_j + \rho_2.$$

If $z_1 \neq z_2$, then

$$\rho_1 = \frac{u_1 - u_2}{z_1 - z_2}, \qquad \rho_2 = u_1 - \rho_1 z_1.$$

The same two proofs recover the other two wire-mask pairs. The adversary then queries the target statement with the same reset parameters and removes the recovered masks, obtaining

$$\sum_{t\in[d]} w_t L_t(z_3) = A_3(z_3) - (\rho_1 z_3 + \rho_2) Z_H(z_3),$$

and similarly for B_3 and C_3. Each session thus yields one evaluation of each of $\widehat{A}, \widehat{B}, \widehat{C}$ for the target. After d distinct target points, interpolate these degree-$< d$ polynomials and evaluate them at the nodes of H. This recovers the complete wire assignment, including the private witness.

To justify the point distribution, first condition on $s \notin H$. With a freshly sampled wire-mask pair, $A(s)$ is uniform in $\mathbb{F}$, because the coefficient of ρ_2 is $Z_H(s) \neq 0$. Consequently, for a fixed statement, its first wire commitment is independent and uniform across sessions. Among d sessions for each of the three statements, commitment collisions have probability at most $3\binom{d}{2}/p$. Different statements give different transcript encodings. Except on these collisions, the evaluation-challenge inputs are fresh, and the random oracle returns independent uniform field elements. This does not require distinct polynomials to have distinct KZG commitments in general.

The $2d$ auxiliary points hit H with probability at most $2d^2/p$, the two auxiliary points in any session coincide with probability at most d/p in total, and the

target points collide with probability at most $\binom{d}{2}/p$. Across $3d$ proofs, permutation denominator failures contribute at most $9d^2/p$. For uniformly sampled s, the setup event $s \in H$ contributes d/p. Excluding the at most $d+2$ exceptional evaluation points for each of the $3d$ proofs adds at most $3d(d+2)/p$ (this deliberately also counts the auxiliary domain hits already listed). A union bound gives failure probability at most

$$\frac{d + 3\binom{d}{2} + 2d^2 + d + \binom{d}{2} + 9d^2 + 3d(d+2)}{p} + \epsilon_{\mathrm{samp}} = \frac{16d^2 + 6d}{p} + \epsilon_{\mathrm{samp}}.$$

The attack uses exactly $3d$ proof queries and polynomial-time interpolation. If setup samples s outside H, its setup-failure term is omitted. This proves Proposition 4.3.

A.4 Adaptive Prefix Replay in Salted Fiat–Shamir

We formalize the replay operation used by the strong resetting attack. Split the compiled prover's tape into its fixed interactive-prover tape ρ and mutually independent λ-bit salt blocks $\tau_1, \ldots, \tau_R$. Each block has a fixed, publicly specified position, and round i samples its salt after computing the prover message a_i. Hash inputs contain the full preceding transcript as in Definition 2.4, use injective encodings, and distinguish rounds; a fresh input gives a uniform challenge in the appropriate challenge space. The interactive algorithms use no access to this compilation oracle beyond the challenges supplied to them. The attack starts with no prior queries in this oracle domain and makes only the proof-generation calls and their challenge lookups. Here M is a deterministic upper bound; intentional replay of a saved hash input is not a collision event.

Lemma A.6 (Adaptive prefix replay). *Suppose an extractor accesses an honest public-coin prover by starting fresh executions, replaying saved prefixes, and extending them with fresh independent uniform challenges. Its choice of saved prefix may depend on its entire revealed view. It may discard sampled challenges and try again, charging every draw to its running time, but cannot prescribe a new challenge or request a draw from a conditioned distribution in one step. If a bounded execution of the wrapper uses at most M round-hash evaluations, including hidden suffixes, then strong-reset access to the salted prover emulates the extractor's view within statistical distance $\binom{M}{2}/2^{\lambda}$.*

Proof. For each returned proof, record the index of its newly appended tape in RandGen, together with the transcript. A fresh root uses $I = \varnothing$. To rewind a saved execution v just before its round-i challenge, use its own recorded index j_v and preserve exactly $\rho, \tau_1, \ldots, \tau_{i-1}$. The recomputed execution has the same prefix through a_i, while the current and later salts are freshly sampled. To replay an already sampled challenge at this prefix, preserve its salt as well. Every branch stores its own index: a later rewind must copy that branch's prefix, which can differ from the initial root's prefix.

The wrapper reveals only the requested interactive messages and challenges. Although each proof query computes a full suffix, the wrapper neither reveals that suffix early nor uses it when choosing a rewind point. Eager sampling of hidden independent salts and oracle outputs is therefore equivalent to sampling them when requested. For Q complete proof queries, counting $M \le QR$ includes these hidden suffix hashes.

Condition on the revealed history and the next chosen rewind point. Each newly sampled salt is independent and uniform. A reused hash input at a fresh-challenge step requires matching a previously used salt in the same hash context. A union bound over at most M sampled blocks gives probability at most $\binom{M}{2}/2^{\lambda}$. Outside this event, each new challenge is a fresh uniform oracle output, while replayed prefixes reproduce their previously sampled challenges exactly. Coupling these operations with the extractor's interactive execution proves the claim.

Theorem A.7 (Witness recovery by prefix replay). *Suppose an honest-prover extractor uses only the interface of Lemma A.6, runs in strict polynomial time, and returns a valid witness with failure probability ϵ_E for the specified PPT target sampler. If its complete-proof wrapper uses at most M round hashes, a strong-reset adversary against salted Fiat–Shamir recovers a valid witness with failure probability at most*

$$\epsilon_E + \binom{M}{2} 2^{-\lambda} + \epsilon_{\mathsf{abort}} + \epsilon_{\mathsf{samp}}.$$

Proof. Run the extractor through the wrapper of Lemma A.6, saving each returned proof's tape index. The lemma couples its entire view to the interactive extraction experiment until a salt collision. On the coupled execution it returns the same candidate witness. Verify the candidate and count any prover abort or sampling discrepancy as failure.

Efficient Extraction from the Honest Prover. The interface of Definition 4.4 is the one emulated in Lemma A.6. We now amplify its extraction guarantee.

Theorem A.8 (Amplifying honest-prover extraction). *Suppose Definition 4.4 holds for $\mathcal{J}$. Under the tape and oracle conventions of this subsection, there is a strict-PPT strong-reset adversary against the salted compilation whose witness-recovery failure probability is at most*

$$\nu(\lambda) + e^{-\lambda} + \binom{M_{\mathrm{tot}}}{2} 2^{-\lambda} + \epsilon_{\mathsf{abort}} + \epsilon_{\mathsf{samp}}.$$

Here M_{tot} bounds all round hashes in $L = \lceil \lambda\psi(\lambda) \rceil$ runs capped at $B(\lambda)$ steps. The last two terms bound aborts and sampling errors jointly over all complete proofs. If these errors are negligible, this is a witness-recovering attack in the sense of Definition 3.5.

Proof. First work with the ideal replay interface. Run E independently L times, each with fresh root tapes, challenge draws, and extractor coins, stopping each

run at $B(\lambda)$ steps. Check each candidate against the public relation and return the first valid witness. For every fixed ζ outside the exceptional set in (1), the runs are independent and each succeeds with probability at least $1/\psi(\lambda)$. Their joint failure probability is at most

$$(1 - 1/\psi(\lambda))^L \le e^{-L/\psi(\lambda)} \le e^{-\lambda}.$$

Averaging over ζ adds at most $\nu(\lambda)$.

Implement all runs through the wrapper of Lemma A.6. A new attempt creates fresh roots; no saved state from an earlier attempt is reused. Each interface call requires at most one complete proof query, and each complete proof has polynomial running time. Thus both the adversary's running time and M_{tot} are polynomially bounded. Apply the lemma once to the entire execution, counting hidden suffixes and collisions between attempts. This joint coupling supplies the independent challenges used above without assuming that actual executions sharing one random oracle are independent. Adding the joint abort and sampling errors proves the bound and Theorem 4.5.

Expected Running Time. A uniform conditional expected-time bound is a sufficient special case. Suppose that, outside a set of contexts of negligible probability μ, an extractor using this interface has expected running time at most a common polynomial $t(\lambda)$. Let $q(\zeta)$ be its uncapped failure probability and suppose $\mathbb{E}_\zeta[q(\zeta)] \le \delta(\lambda)$ for negligible δ. Markov's inequality gives $\Pr[q(\zeta) > 1/4] \le 4\delta$. On every remaining context, a cutoff at $4t(\lambda)$ loses success probability at most $1/4$, so the capped run succeeds with probability at least $1/2$. Thus Definition 4.4 holds with $B = 4t$, $\psi = 2$, and $\nu = \mu + 4\delta$.

A bound averaged only over contexts does not give this conclusion. For example, an always-successful procedure can take λ^k steps on a publicly specified class of contexts of probability proportional to λ^{-2k}, for each $1 \le k \le \lambda$. Its average running time is polynomial, but any fixed polynomial cutoff misses an inverse-polynomial mass of contexts. This illustrates the need to bound success within the cutoff on almost every fixed target.

Two-Special-Sound Protocols. For a two-special-sound Σ-protocol, two accepting transcripts with the same first message and different challenges determine a witness.

Corollary A.9. *Let Π be a perfectly complete, two-special-sound Σ-protocol with an efficiently sampled uniform challenge space $\mathcal{C}$ of size $C \ge 2$ and a polynomial-time special-soundness extractor. Then its salted Fiat–Shamir compilation admits a strong-reset witness-recovering attack. More precisely, for every polynomially bounded integer $m \ge 1$, one fresh proof and m replays fixing the prover tape suffice with failure probability at most*

$$C^{-m} + \binom{m+1}{2} 2^{-\lambda} + \epsilon_{\mathsf{samp}}.$$

In particular, $m = \lambda$ suffices even for binary challenges; two proofs suffice when $1/C$ is negligible.

Proof. Save the first message and challenge c. Replay this state with independent uniform challenges until one differs from c, or until m additional draws have been made. Perfect completeness gives two accepting transcripts when a different challenge appears, and special soundness extracts a valid witness. All m additional draws equal c with probability C^{-m}. There is one salted challenge per proof, so Lemma A.6 gives the stated collision term. The special-soundness computation and the bounded number of draws are polynomial-time.

For Schnorr [63] and the field-challenge Okamoto protocol [55], the honest responses satisfy $z = r + cw$ and $z_j = r_j + cw_j$, respectively. Two different field challenges give

$$w = (z - z')/(c - c'), \qquad w_j = (z_j - z'_j)/(c - c').$$

These formulas recover the actual honest witness using only field arithmetic. For Okamoto, this salted attack is distinct from the three-query unsalted attack in Lemma A.4.

Polynomial-Size Challenge Trees. The same argument extends to multi-round special soundness [4]. An accepting $(k_1, \ldots, k_R)$-tree has a common first prover message, k_i distinct challenges at each round-i node, and accepting transcripts on all root-to-leaf paths. Its prover messages agree on common prefixes, and the branching numbers are public, efficiently computable protocol parameters. Write

$$K = \prod_{i=1}^{R} k_i, \qquad S = \sum_{i=0}^{R} \prod_{j=1}^{i} k_j$$

for its number of leaves and total nodes, with the empty product equal to one.

Corollary A.10. *Let Π be a perfectly complete public-coin protocol with efficient witness extraction from every accepting $(k_1, \ldots, k_R)$-tree. If R, K are polynomially bounded and $1 \le k_i \le |\mathsf{Ch}_i|$, then it admits efficient honest-prover replay extraction. Its salted compilation therefore admits witness recovery under strong resetting with the stated tape layout.*

Proof. Start one fresh root and construct the tree recursively. At a round-i node, replay its saved prefix and sample challenges until k_i distinct children have been collected; then extend each child in the same way. Every branch uses the same root tape. Once $j < k_i$ distinct children have been collected, the probability that the next uniform challenge is new is

$$1 - j/|\mathsf{Ch}_i| \ge 1/k_i.$$

Thus the expected number of draws at that node is at most k_i^2. If $k_{\max} = \max_i k_i$, the total expected number of draws is at most

$$\sum_{i=1}^{R} \left(\prod_{j<i} k_j \right) k_i^2 \le k_{\max}(S - 1).$$

Since $S \leq (R+1)K$, this is polynomial. Message lengths, replay computation, and the tree extractor's running time are also polynomial. Perfect completeness makes every leaf accepting, so the extractor succeeds with probability one. These bounds hold uniformly for every fixed context and root tape. The conditional expected-time argument above gives Definition 4.4; amplification and salted replay complete the attack.

The polynomial bound on K is essential to this argument. Constant branching over logarithmically many rounds is covered, while branching twice in each of linearly many rounds gives an exponential tree. More generally, the same proof applies when a publicly testable rule chooses suitable children and each next suitable child is found by uniform sampling with inverse-polynomial probability, provided the total required tree remains polynomial in size. Merely allowing an extractor to choose challenges does not establish this sampling bound.

A.5 Strong Resetting Attacks on Bulletproofs

The honest-prover condition also applies to inner-product arguments. We first reverse the folding steps of the Bulletproofs inner-product argument [18, Section 3] using salted prefix replay. This argument is a proof of knowledge without zero-knowledge, so recovering its vectors illustrates prefix replay on a logarithmic-round protocol; the privacy violations are the range-proof attacks that follow. For range proofs, we instead interpolate a quadratic exposed by the prover. This recovers a value from three proofs and a full commitment opening from five, without salts. For m aggregated values, the full-opening attack uses $3m + 2$ proofs, independent of the bit length.

Honest Inner-Product Extraction. Let the vector length be $N = 2^d$, and let the scalar field be $\mathbb{F}_p$ for an odd prime p. At a folding step, split the honest vectors into equal halves. On challenge $u \in \mathbb{F}_p^*$, the prover replaces them by

$$a' = ua_L + u^{-1}a_R, \qquad b' = u^{-1}b_L + ub_R.$$

After d steps the proof reveals the two remaining scalars. These are the honest folding equations of Protocol 2 in [18].

Proposition A.11. *For polynomially bounded N, the salted Bulletproofs inner-product argument admits recovery of its honest witness vectors from at most N complete proofs, with failure probability at most*

$$\frac{2(N-1)}{p-1} + \binom{NR}{2} 2^{-\lambda} + \epsilon_{\mathsf{samp}},$$

where R is the number of salted challenges per proof. The tape and oracle conventions are those of the prefix-replay analysis, and all scalar challenges are uniform in $\mathbb{F}_p^*$.

Proof. At each saved folding state, obtain two continuations with challenges u, v. If $u^2 = v^2$, report failure. Otherwise, recursively recover both child pairs (a'_u, b'_u) and (a'_v, b'_v). For each coordinate, solve

$$\begin{pmatrix} u & u^{-1} \\ v & v^{-1} \end{pmatrix} \begin{pmatrix} a_L \\ a_R \end{pmatrix} = \begin{pmatrix} a'_u \\ a'_v \end{pmatrix}, \qquad \begin{pmatrix} u^{-1} & u \\ v^{-1} & v \end{pmatrix} \begin{pmatrix} b_L \\ b_R \end{pmatrix} = \begin{pmatrix} b'_u \\ b'_v \end{pmatrix}.$$

Both matrices are invertible when $u^2 \neq v^2$. At depth d, the required child vectors are the scalars exposed by the proof. Induction therefore recovers the exact honest vectors at every node, including the root. Any initial challenge in Protocol 1 is kept fixed; its honest input vectors are unchanged by that challenge.

One full proof supplies an initial path. At each internal node, its saved proof supplies the first child and one further complete proof supplies the second. A binary tree of depth d has $N-1$ internal nodes, so at most N proofs are needed. With independent uniform nonzero challenges, a fresh child has the same squared challenge as its sibling with probability $2/(p-1)$. Union-bound over the internal nodes, then apply Lemma A.6 to all NR round hashes. This proves the claim. Only polynomial-time field arithmetic and public transcript data are used.

This extraction uses the honest folding identities. It does not assert two-special soundness against a malicious prover; the latter requires a separate argument. Thus the honest-prover formulation gives a direct attack without importing the running-time guarantees of witness-extended emulation.

Range Proofs without Salts. We use the single and aggregated logarithmic range proofs of [18, Sections 4.1–4.4]. The statement consists of m Pedersen commitments $V_j = g^{v_j} h^{\gamma_j}$ to values $0 \leq v_j < 2^n < p$. Their concatenated bit vector is $a_L \in \{0,1\}^N$, where $N = mn = 2^d$. In the notation of that protocol, the prover samples masks $\alpha, \rho, s_L, s_R, \tau_1, \tau_2$ and sends

$$A = h^{\alpha} \mathbf{g}^{a_L} \mathbf{h}^{a_R}, \qquad S = h^{\rho} \mathbf{g}^{s_L} \mathbf{h}^{s_R}, \qquad a_R = a_L - \mathbf{1}.$$

Write $y, z, x \in \mathbb{F}_p^*$ for the Fiat–Shamir challenges and define

$$\delta_m(y,z) = (z - z^2) \sum_{k=0}^{N-1} y^k - (2^n - 1) \sum_{j=1}^{m} z^{j+2}.$$

The prover constructs a quadratic whose constant term encodes the committed values:

$$\begin{aligned} t(X) &= t_0 + t_1 X + t_2 X^2, \\ t_0 &= \delta_m(y,z) + \sum_{j=1}^{m} z^{j+1} v_j. \end{aligned} \tag{2}$$

The coefficient commitments are $T_k = g^{t_k} h^{\tau_k}$ for $k \in \{1,2\}$. The prover exposes scalar responses $\widehat{t}$ and τ_x satisfying

$$\begin{aligned} \widehat{t} &= t(x) = t_0 + x t_1 + x^2 t_2, \\ \tau_x &= \tau_1 x + \tau_2 x^2 + \sum_{j=1}^{m} z^{j+1} \gamma_j. \end{aligned} \tag{3}$$

The identity for t_0 follows by expanding the range protocol's inner product and using $a_L \in \{0,1\}^N$; the formula for τ_x follows from the coefficient commitments. Both also hold in the logarithmic version, which replaces the transmitted vectors by the inner-product argument.

Theorem A.12 (Unsalted Bulletproofs range recovery). *Let $N = mn = 2^d$ be polynomially bounded, with $m, n \geq 1$, $2^n < p$, and negligible $1/p$. Use the Bulletproofs range prover above in a group of prime order $p > m+1$ with generators g, h. Suppose ρ and τ_1 are independent uniform field elements sampled from separate, fixed public tape blocks, independently of the other masks. Use unsalted Fiat–Shamir with injective, round-separated encodings of the complete transcript prefix and uniform challenges in $\mathbb{F}_p^*$. For every valid target sampler independent of this oracle, a strict-PPT strong-reset adversary, starting with no prior queries in this oracle domain, recovers all pairs (v_j, γ_j) from $3m+2$ target proofs, with failure probability at most*

$$\frac{\binom{m+2}{2} + 3m}{p} + \frac{\binom{m}{2} + 3m + (m+2)(m+1)}{p-1} + \epsilon_{\mathsf{samp}}. \tag{4}$$

Here ϵ_{samp} bounds the joint discrepancy from ideal independent uniform mask and challenge sampling for this attack.

Proof. Request one fresh base proof, followed by $m+1$ copies that refresh only ρ. For each of the first m base proofs, make two further copies that preserve every block except τ_1, which is independently refreshed. Thus there are $3m+2$ proofs. We first analyze a reference experiment with uniform masks and fresh independent challenges after each refreshed commitment, replaying earlier challenges. In Fiat–Shamir, repeated prefixes reuse their challenges and new prefixes receive fresh oracle outputs. The two experiments agree until a refreshed commitment repeats; we bound these collisions below.

Across the $m+2$ base proofs, τ_1, τ_2 and all γ_j are fixed. By (3), the public responses give $m+2$ linear equations in $(\tau_1, \tau_2, \gamma_1, \ldots, \gamma_m)$, with row vectors

$$(x, x^2, z^2, z^3, \ldots, z^{m+1}).$$

These rows have full rank except with probability at most $(m+2)(m+1)/(p-1)$. To see this, condition on any proper span of preceding rows and choose a nonzero linear functional annihilating it. Its value on a fresh row is a nonzero polynomial in independent uniform $x, z \in \mathbb{F}_p^*$, of total degree at most $m+1$, since the monomials $x, x^2, z^2, \ldots, z^{m+1}$ are distinct. Schwartz–Zippel bounds the probability that the next row fails to increase rank by $(m+1)/(p-1)$. A union bound gives the claim. Solving these equations recovers all commitment blindings γ_j.

Now fix one of the first m base proofs and its two τ_1-refreshed copies. They have the same A, S, hence the same y, z, and the same polynomial $t(X)$, while their T_1 commitments change. On distinct T_1 values, the two copies receive fresh uniform challenges, independently of the base challenge. If the three challenges

x_1, x_2, x_3 are distinct, the pairs $(x_r, \widehat{t}_r)$ determine the quadratic and in particular its constant term t_0 by interpolation. Conditional on the base proof and prior history, each refreshed τ_1 makes the new $T_1 = g^{t_1} h^{\tau_1}$ independently uniform. A repetition within the triple therefore has probability at most $3/p$. The two fresh challenges collide with the base challenge or each other with probability at most $3/(p-1)$. Over the m triples these contribute $3m/p + 3m/(p-1)$.

Apply this interpolation to the first m base proofs, obtaining $t_{0,i}$ and their respective challenges z_i. Equation (2) gives

$$\frac{t_{0,i} - \delta_m(y_i, z_i)}{z_i^2} = \sum_{j=1}^{m} z_i^{j-1} v_j, \qquad i \in [m].$$

The Vandermonde system recovers all values when the z_i are distinct. Since the base proofs use distinct S values, their z_i challenges are independent uniform elements of $\mathbb{F}_p^*$, and a collision has probability at most $\binom{m}{2}/(p-1)$. The initial S is uniform because h^ρ is uniform. Among the $m+2$ base queries, an S collision has probability at most $\binom{m+2}{2}/p$; outside this event, the complete-prefix encoding gives fresh challenges for each base proof and for each new T_1 in a triple. Adding all rank, collision, and sampling errors proves (4). The adversary checks the recovered openings and ranges before returning them, and all computations and queries are polynomial-time.

For $m = 1$, five proofs recover the full opening (v_1, γ_1). Recovering the value alone needs fewer resettable blocks and only three proofs:

Corollary A.13 (Three-proof value recovery). *For the single-value prover above, it suffices that τ_1 occupy an independent, fixed public tape block. Under the same sampling and oracle conventions, for every valid oracle-independent target sampler, a strict-PPT adversary recovers the sampled value from three target proofs with failure probability at most*

$$\epsilon_3 = \frac{3}{p} + \frac{3}{p-1} + \epsilon_{\mathsf{samp}}.$$

For fixed external group parameters, there is also a distinguisher with advantage at least $1 - \epsilon_3 - 2^{-n}$ under ideal uniform witness sampling. Thus the prover is not srNIZK when ϵ_3 is negligible.

Proof. Copy the first target tape twice, refreshing only τ_1. The three executions have the same A, S, y, z and the same quadratic $t(X)$, while their T_1 commitments give fresh evaluation challenges except with probability $3/p$. With distinct challenges, interpolate t_0 from the three values $\widehat{t} = t(x)$. Since $t_0 = \delta_1(y, z) + z^2 v$ and $z \neq 0$, this recovers v; the interpolation fails with probability at most $3/(p-1)$. This proves the recovery bound.

For the zero-knowledge distinguisher, sample $v \leftarrow \{0, \ldots, 2^n - 1\}$ and $\gamma \leftarrow \mathbb{F}_p$ uniformly after setup. Form $V = g^v h^\gamma$, retain (v, γ), submit this witness for V, request the three proofs above, and output one exactly when the recovered value

equals the retained v. In the real experiment this happens with probability at least $1-\epsilon_3$. In the ideal experiment, V is uniform in the group and independent of v, because γ is uniform and h generates the group. The simulator receives the public statement and reset metadata but not the witness; its simulated tapes and cache pattern are also independent of v. Thus the recovered candidate is independent of the uniform v and matches it with probability at most 2^{-n}. The distinguishing gap is at least $1-\epsilon_3-2^{-n}$. Using a bounded sampler for γ reduces this bound by at most twice its statistical error, which can be made negligible.

The transcript-prefix convention agrees with Definition 2.4 and the Fiat–Shamir instantiation in [18, Section 4.4]. The attack uses the independent-mask prover and reset interface stated above. In dalek, the range prover samples these masks from the caller's RNG [28]; the separate R1CS prover instead uses a witness-rekeyed transcript RNG [27]. A prover that derives its masks through a transcript RNG requires separate analysis of that derivation and of where resets act; these results do not claim an attack on every implementation using the Bulletproofs name.

A.6 Strong Resetting Attacks on PIOPs

A later prover polynomial may depend on earlier challenges. To recover its coefficients, we preserve the prefix that defines it and vary only the subsequent challenges. The following theorem analyzes the attack of Figure 7 when these evaluations determine the polynomial coefficients. Section A.7 proves the general attack without this full-rank requirement.

Recovering Polynomials from a Fixed Prefix. Consider the salted compilation with tape layout from the prefix-replay analysis. The fixed tape ρ includes the PIOP and commitment coins; the salt blocks are independent and occupy fixed public positions. Each proof exposes the verifier's evaluation points and values. Fix polynomial groups $g \in [s]$, with group g sent before challenge t_g and having D_g coefficients in total. Evaluation of this group gives a publicly known linear map on its coefficient vector. All dimensions and query counts below are polynomially bounded.

Theorem A.14 (Fixed-prefix recovery). *Suppose that, for each group g, $N_g \geq 1$ independent uniform continuations of its defining prefix give a stacked evaluation map of rank D_g, except with probability ϵ_g. Suppose a valid witness can be efficiently decoded from these groups in one honest execution and its public challenge transcript. Then a strong-reset adversary uses*

$$Q = 1 + \sum_{g=1}^{s} (N_g - 1)$$

proof queries and recovers a valid witness with failure probability at most

$$\sum_{g=1}^{s} \epsilon_g + \binom{QR}{2} 2^{-\lambda} + \epsilon_{\mathsf{abort}} + \epsilon_{\mathsf{samp}},$$

where R is the number of salted challenges, $\epsilon_{\mathsf{abort}}$ bounds aborts over all Q proofs, and ϵ_{samp} is their joint sampling error. The rank bounds are measured over the target distribution and the random defining prefix, conditioned on which the continuations are independent.

Proof. Request a fresh base proof and save its tape index v, challenge transcript, and evaluations. For group g, make $N_g - 1$ queries copying $\rho, \tau_1, \ldots, \tau_{t_g-1}$ from tape v and refreshing the remaining salts. This preserves the base execution through the group's polynomials and commitments. The new proofs therefore expose further linear measurements of the same coefficient vector.

By Lemma A.6, the joint distribution differs from independent uniform continuations by at most $\binom{QR}{2}2^{-\lambda}$. The count includes every hash in every complete proof, even in suffixes that the reconstruction does not use. Together with the base proof, the continuations supply N_g measurements for group g. On the full-rank event, solve the resulting linear system to recover its coefficients. All groups are recovered from their respective prefixes of the same base execution, so they form one compatible honest transcript. Apply the decoder using that transcript's challenges. A union bound over rank failures, aborts, and sampling error proves the result.

A.7 From Coefficient Extraction to Strong Resetting

We now prove the general attack from the coefficient-oracle knowledge guarantee in Definition 2.9. This attack covers protocols in which the exposed evaluations do not determine the prover's polynomials, for example when a mask coefficient affects no queried value: the extractor only needs coefficients that are consistent with those evaluations. To run the extractor, we must answer even challenge queries that the resetting adversary cannot reproduce. We do so using a polynomial-size table of sampled prefixes, returning zero polynomials after a query leaves the table. We then use the knowledge guarantee to prove that this restricted interaction suffices for extraction.

A Table of Challenge Prefixes. Fix a public context and an R-round public-coin protocol with $R \geq 1$. Draw N independent complete challenge sequences, and let T contain their prefixes of lengths $0, \ldots, R-1$. Thus T is prefix closed and contains at most NR sending prefixes. For $0 < \xi < 1$, at any depth there are at most $1/\xi$ prefixes having probability at least ξ. Each such prefix is absent from T with probability at most $e^{-N\xi}$. Consequently,

$$\Pr[T \text{ misses a prefix of probability at least } \xi] \leq \eta := \frac{R}{\xi} e^{-N\xi}. \tag{5}$$

Outside this event, each possible first prefix outside T has probability less than ξ in a fresh reference interaction. We construct the table by efficient challenge sampling, without enumerating the challenge space.

Restricting the Coefficient Oracle. Let P be any admissible prover whose initialization may include T. Its reference challenges are independent of the samples used to construct T. Define P_T to agree with P at sending prefixes in T, and to send zero polynomials of the prescribed dimensions after leaving T. Let E be the extractor in Definition 2.9, with a polynomial bound $q \geq 1$ on its oracle interaction steps. The table entries and reference interaction use the same sampled prover initialization. Each table entry is computed by replaying its prefix from that initialization, with no resampling of the prover coins. The algorithm E^{P_T} can be run from the coefficient table on T alone.

Choose $\gamma = 2^{-t}$. Independently sample a pairwise-independent hash h from injective encodings of challenge prefixes to $\{0,1\}^t$. Such a hash is obtained by a random affine binary map on fixed-length padded encodings. Define a prover P_h that agrees with P inside T. At its first sending prefix $v \notin T$, it continues as P if $h(v) = 0^t$, and otherwise sends zero polynomials for the rest of that branch. Rewinding restores the branch state. In particular, P_h is an admissible PPT prover; its first messages do not reveal the independent hash seed.

Lemma A.15 (Extraction from a sampled prefix table). *Fix a public context and any auxiliary state external to the sampled prover initialization. Let δ be the joint probability that the reference interaction with P accepts and E^{P_T} returns an invalid witness. Let e be the corresponding knowledge-soundness error for P_h. Under the prefix-cover bound (5),*

$$\delta \leq q\gamma + q\xi + \eta + e/\gamma.$$

The probabilities include the common prover initialization, the sampled table, and the independent reference and extractor coins.

Proof. Couple E^{P_h} and E^{P_T} using the same extractor coins and the same initialization of P. Consider the hypothetical interaction with P_T, which is independent of h, and list the at most q first-exit prefixes queried in it. Until one of these prefixes hashes to zero, both extractor interactions have identical replies.

Let C be the reference interaction's first sending prefix outside T. If the reference never leaves T, give C a distinct sentinel encoding; its hash value is used only in this analysis. On the event $h(C) = 0^t$, the reference interaction with P_h agrees with that with P. This remains true in the sentinel case, since the reference then stays inside T.

Condition on a table satisfying the prefix-cover event, the remaining prover initialization, and the extractor coins. A listed prefix v equals C with probability at most ξ. For $v \neq C$, pairwise independence gives

$$\Pr\left[h(v) = 0^t \ \wedge \ h(C) = 0^t\right] = \gamma^2.$$

Thus the probability that $h(C) = 0^t$ and any listed prefix hashes to zero is at most $q\gamma^2 + q\xi\gamma$. The joint acceptance-and-failure event defining δ is independent of h. Its intersection with $h(C) = 0^t$ has probability $\gamma\delta$. Discarding failed coverage costs at most $\gamma\eta$. On every remaining event without a listed hash hit, the

reference accepts and E^{P_h} returns the same invalid witness as E^{P_T}. Therefore

$$e \geq \gamma\delta - \gamma\eta - q\gamma^2 - q\xi\gamma,$$

which proves the claim using only acceptance on the selected hash event.

Learning the Coefficient Table. The following lemma bounds the chance that a fresh evaluation reveals a linear constraint outside the span of the sampled constraints.

Lemma A.16 (Learning the span of evaluation maps). *Let $L_1, \ldots, L_{n+1}$ be independent, identically distributed matrices with D columns, and let U_n be the span of all rows of $L_1, \ldots, L_n$. Then*

$$\Pr[\mathrm{rowspan}(L_{n+1}) \not\subseteq U_n] \leq \frac{D}{n+1}.$$

Proof. Write p_j for the probability that sample $j+1$ enlarges the span of the first j samples. Since the samples are independent and their spans increase with j, p_j is nonincreasing. At most D samples can enlarge the span, so $\sum_{j=0}^{n} p_j \leq D$. Hence $(n+1)p_n \leq D$.

We now construct the table from exposed evaluations and use Lemma A.16 to bound its error. Group the prover polynomials by round. Let D_r be the total coefficient dimension of the concatenated vector of all polynomials sent in round r, and set

$$n_r = \lceil 4RD_r \rceil.$$

Start with one fresh honest prover tape and sample N complete executions using that tape and independent challenges. These executions define T. Each prefix in T occurs in one of these executions, which the attacker saves for later replay. This gives access even to prefixes with low probability.

For every sending prefix $v \in T$ of length $r-1$, obtain n_r additional independent continuations from its saved execution. Each continuation contributes the full block of query functionals for the round-r polynomial vector. Solve their evaluation equations for a canonical coefficient vector $\widehat{a}_v$, with a fixed variable order and free variables set to zero. The honest polynomial vector at v satisfies these equations. Run E against the table oracle that returns $\widehat{a}_v$ at $v \in T$ and zero polynomials after leaving T, then check the returned witness.

To analyze this procedure, define an admissible prover $\widehat{P}$ with the honest witness as hidden auxiliary input. This witness is part of the prover initialization and is not given to E. It samples an honest tape, T, and the independent training continuations for every node of T. Given the witness, it computes the honest coefficients and constructs the same projected table. At a prefix in T it sends that table's entry. Outside T it sends the honest polynomial vector, computed with its original honest tape and the actual challenge prefix. Its internal honest state is maintained independently of the projected messages.

All these computations take polynomial time. The table is fixed before the reference interaction, so $\widehat{P}$ chooses each message before receiving the next actual challenge. Conditional on a visited prefix, the honest tape, and T, its training continuations and the actual future continuation are independent. By Lemma A.16, the projected and honest evaluations for round r disagree with probability at most

$$D_r/(n_r+1) \le 1/(4R).$$

Outside T the polynomials are honest. A union bound and perfect completeness therefore give reference acceptance probability at least $3/4$, for every fixed public context, target, and witness. The coefficient table learned by the attacker has exactly the distribution of $\widehat{P}$'s table.

Theorem A.17 (Strong resetting from coefficient-oracle extraction). *Suppose a perfectly complete PIOP satisfies Definition 2.9, with polynomially many rounds, polynomials, and coefficients. Assume its public query map, publicly available preprocessing coefficients, and the salted tape layout and exposed evaluations of Section A.6. Then, for every PPT valid target sampler, its salted Fiat–Shamir compilation admits a strict-PPT strong-reset witness-recovering attack.*

More precisely, choose a dyadic $\gamma = \xi \le 1/(64(q+1))$ with $1/\gamma$ polynomially bounded, and

$$N = \xi^{-1}\big(\lambda + 5 + \lceil \log_2(R/\xi) \rceil\big), \qquad Q = \lambda N \left(1 + \sum_{r=1}^{R} n_r\right).$$

The attack uses at most Q proof queries and has failure probability at most

$$\frac{16\nu}{\gamma} + (3/8)^{\lambda} + \binom{QR}{2} 2^{-\lambda} + \epsilon_{\mathsf{abort}} + \epsilon_{\mathsf{samp}},$$

where ν is the negligible knowledge-soundness error of the admissible prover constructed in the proof. Abort and sampling errors are joint bounds over all Q proofs.

Proof. We first analyze one attempt with independent uniform challenges. Apply Lemma A.15 with $P = \widehat{P}$. Write ζ for the fixed public context, target, witness, and sampler side information. Let $e(\zeta)$ be the conditional knowledge-soundness error of the hash-gated prover $\widehat{P}_h$. This is one fixed admissible PPT prover, so the knowledge guarantee gives

$$\mathbb{E}_\zeta[e(\zeta)] \le \nu.$$

By Markov's inequality,

$$\Pr_\zeta[e(\zeta) > \gamma/16] \le 16\nu/\gamma.$$

The choice of N gives $\eta \le 2^{-\lambda}/32 \le 1/32$, and $q\gamma + q\xi \le 1/32$. Hence on every remaining ζ, the table lemma bounds the joint acceptance-and-extraction error

by $1/8$. Since $\widehat{P}$ accepts with probability at least $3/4$, the table extractor returns a valid witness with probability at least $5/8$.

Repeat the attempt λ times with fresh honest tapes, sampled tables, training continuations, and extractor coins. Conditional on ζ, these attempts are independent, so all fail with probability at most $(3/8)^\lambda$ outside the exceptional set above. Each attempt uses N exploration proofs and at most Nn_r training proofs for each round r, which gives the claimed bound Q. The attacker uses only fresh roots, saved-prefix replay, independent uniform continuations, and the evaluations exposed by the verifier's query routine.

Finally, Lemma A.6 implements these operations through strong resets. It bounds the joint statistical error by $\binom{QR}{2}2^{-\lambda}$, counting every round hash in every complete proof, including unused suffixes. Add the joint abort and sampling errors to conclude.

The universal coefficient-oracle knowledge guarantee supplies the extraction used in this theorem, without assuming honest coefficient decoding. We use the same strict-PPT black-box extractor for every admissible prover, including the hash-gated prover, and give it no initial accepting transcript. These quantifiers are essential to the argument; the theorem does not extend this conclusion to acceptance-dependent extraction bounds or expected-time extractors.

A.8 Concrete PLONK Bounds without Salts

We use the transcript, the blinders $\rho_1, \ldots, \rho_9$, and the excluded evaluation points of Section A.3; this attack needs neither a salt nor auxiliary inputs.

Request the first target proof with a fresh tape. For every subsequent query, use $\mathsf{idx} = 1$ and let I contain every tape position except the block for ρ_9. Only ρ_9 is freshly sampled by $\mathsf{RandGen}$, so A, B, C and their commitments remain fixed. The challenges β, γ, and hence $\widehat{P}$, also remain fixed; only the constant blinder of the permutation polynomial changes.

Condition on $s \notin H$ and on nonzero permutation denominators. For each execution, the fresh ρ_9 makes

$$P(s) = \widehat{P}(s) + (\rho_7 s^2 + \rho_8 s + \rho_9) Z_H(s)$$

uniform in $\mathbb{F}$, even conditioned on the fixed wire polynomials and earlier executions. Its KZG commitment is therefore uniform and independent across executions. Since this commitment is included in the evaluation-challenge hash input, those inputs are distinct except with probability $\binom{\eta}{2}/p$ over η queries. Changes to the intervening quotient challenge and quotient polynomials do not change the fixed wire polynomials.

For fresh evaluation inputs, the η points are independent uniform field elements and collide with probability at most $\binom{\eta}{2}/p$. Set $\eta = d + 2$. Each masked wire polynomial has degree at most $d + 1$, so these points suffice to interpolate A, B, C. Evaluating on H removes the masks and returns all wire values. Interpolation itself requires no division by Z_H; nevertheless, we exclude the bad

evaluation points of Section A.3 to ensure the ordinary proof-generation and verification paths apply. Its total failure probability is at most

$$\frac{d+3d+2\binom{d+2}{2}+(d+2)^2}{p}+\epsilon_{\mathrm{samp}}=\frac{2d^2+11d+6}{p}+\epsilon_{\mathrm{samp}}.$$

This proves witness recovery from $d+2$ target proofs in unsalted PLONK under the stated randomness layout. KZG commitments to fixed polynomials are deterministic; the attack varies the permutation polynomial itself while preserving the wire polynomials to be recovered.

B Partial Resets and XOR-Related Keys

We prove the equivalence stated in Section 5.3.

Proof (Proof of Lemma 5.6). First, simulate the reset oracle using one hidden root key K from the related-key game. Maintain a list of offsets $\delta_1, \ldots, \delta_t$, representing the generated keys as $K_i = K \oplus \delta_i$. For the first well-formed query, set $\delta_1 = 0^\ell$. For a subsequent valid reset request (idx, I), copy $\delta_{\mathsf{idx}}[j]$ at coordinates $j \in I$ and sample an independent uniform bit at every coordinate $j \notin I$. Append the resulting offset δ_{t+1}, and obtain the answer for input x by querying the related-key oracle at (δ_{t+1}, x). Reject malformed requests and out-of-range indices exactly as $\mathsf{RandGen}$ does, without appending an offset.

The initial key is uniform. On later queries, copied coordinates agree with K_{idx}, while each other coordinate is uniform and independent of the history, even conditional on K. Thus the entire adaptive key process has exactly the distribution of $\mathsf{RandGen}$. The real related-key oracle returns the required PRF value; the ideal oracle returns the required value of R. Since translation by K is a bijection, equal offsets represent equal keys, so all repetitions and accidental key collisions are preserved. The two simulations are exact.

For the reverse direction, let $\mathcal{A}$ make at most q XOR-related-key queries and set $T = \lambda$. An $\mathsf{sRK\text{-}PRF}$ adversary $\mathcal{B}$ fixes an input x_0 and makes a first, well-formed reset query on x_0. This creates the root key K at list index 1 and reveals $y_0 = \mathsf{F}_K(x_0)$ in the real experiment, or $y_0 = \mathsf{R}(K, x_0)$ in the ideal one. To answer a query (δ, x) from $\mathcal{A}$, start with handle $h = 1$ and its known probe value y_0. For each coordinate i with $\delta_i = 1$, query x_0 with reset request $(h, [\ell] \setminus \{i\})$, refreshing only bit i. If the answer differs from the probe value at h, replace h and its probe value by the new index and answer. Otherwise retry from the same reference handle h, for at most T fresh trials. Abort if no trial succeeds. After all requested bits have been flipped, query x with request $(h, [\ell])$ and return that answer to $\mathcal{A}$. Every trial uses its actual appended list index; failed trials are never used as reference handles. On abort, $\mathcal{B}$ outputs 0.

A changed probe value certifies that the refreshed bit changed, since the new key can differ from the reference key only at that bit. Hence, unless it aborts, the final handle represents exactly $K \oplus \delta$, and the full-copy query evaluates the desired function at x. This holds even if the PRF has output collisions: those

collisions can cause an abort but cannot certify a false flip. The full-copy query also ensures that repeated (δ, x) queries receive the same value from the single random function in the ideal experiment.

Let $L = 1 + q\ell T$. The root and bit-refresh steps make at most L probes at x_0; a full-copy query at x_0 repeats an already-probed key. In the ideal $\mathsf{sRK\text{-}PRF}$ experiment, a collision between probe outputs at distinct keys has probability at most $\binom{L}{2}2^{-\lambda}$, by lazy sampling, including for adaptively chosen keys. If no distinct-key probe outputs collide, a T-trial timeout requires all T independently resampled bits to match the reference bit. A union bound over at most $q\ell$ requested flips gives

$$p_* := \Pr[\mathcal{B} \text{ aborts in the ideal experiment}] \leq q\ell 2^{-T} + \binom{L}{2}2^{-\lambda}.$$

Let $\mathcal{D}$ perform the same simulation and output its abort indicator. Then its distinguishing advantage bounds the difference between the real and ideal abort probabilities. If $\mathsf{Adv}^{\mathsf{XOR\text{-}RKA\text{-}PRF}}_{\mathcal{A},\mathcal{F}}$ denotes $\mathcal{A}$'s advantage in the XOR-related-key experiment, the two simulations give

$$\mathsf{Adv}^{\mathsf{XOR\text{-}RKA\text{-}PRF}}_{\mathcal{A},\mathcal{F}}(\lambda) \leq \mathsf{Adv}^{\mathsf{sRK\text{-}PRF}}_{\mathcal{B},\mathcal{F}}(\lambda) + \mathsf{Adv}^{\mathsf{sRK\text{-}PRF}}_{\mathcal{D},\mathcal{F}}(\lambda) + 2p_*.$$

Both $\mathcal{B}$ and $\mathcal{D}$ make at most $1 + q\ell T + q$ reset-oracle queries and run in strict polynomial time. Since q and ℓ are polynomial, p_* is negligible. This establishes equivalence of the two PRF security notions.