TRAPGC-DV: Trapdoor Garbled Circuit Designated Verification of Groth16 on Bitcoin

Ariel Futoransky[1], Fadi Barbàra[1,2], Ramses Fernandez[1], Gabriel Larotonda[3,4] and Sergio Demian Lerner[1,5]

[1] Fairgate Labs

[2] Sapienza Università di Roma, Italy

[3] Universidad de Buenos Aires, Argentina

[4] CONICET, Argentina

[5] Rootstock Labs

Abstract. Verifying Groth16 proofs on Bitcoin enables a broad class of trust-minimised applications, bridges, light clients, validity rollups, but Bitcoin script lacks the pairing operations required to evaluate the verification equation natively. A recent line of work on Bitcoin-side garbled-circuit SNARK verification (Delbrag, BitVM3, BitVMX-GC, Glock, Argo MAC) has produced practical constructions but does not provide formal universal-composability analysis.

We present $\Pi_{\mathsf{DV\text{-}Groth16}}$, a protocol for *designated evaluation* of Groth16 proofs on Bitcoin. The prover holds a standard Groth16 proof π_{in} under a public, one-time ceremony SRS. The verifier generates, from a secret seed, a compact per-session trapdoor SRS for a fixed outer circuit, the inner Groth16 *verifier*, computed natively over a one-layer 2-chain (BLS12-377 inside BW6-761), and proves it well-formed. The prover produces an outer Groth16 proof that her inner proof verifies, and garbles a small circuit that checks the *outer* proof in scalar form: after the verifier reveals his seed, the prover signs the discrete logarithms of her pre-committed outer proof elements, and the circuit checks them against the commitments together with the scalarized verification identity, releasing a slashing key on failure. Cut-and-choose enforces garbling correctness; deposits and commitments are locked in Bitcoin UTXOs with a three-tier settlement (`refund`, `slash-fast`, `slash-commit`). The per-session SRS has fixed size, *independent of the assertion circuit*, and all elliptic-curve computation inside the garbled circuit reduces to three fixed-base scalar multiplications.

We prove that $\Pi_{\mathsf{DV\text{-}Groth16}}$ GUC-emulates an ideal functionality $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$ capturing the three-tier behaviour against static corruption in the GUC-hybrid model, with a concrete bound assembled from named primitive advantages. We make precise how designated *evaluation and enforcement* differs from designated-verifier *deniability*, and exhibit a variant achieving the latter at the cost of a per-proof SRS for the assertion circuit itself.

Keywords: Universal composability · Designated-verifier zero-knowledge · Garbled circuits

E-mail: futo@fairgate.io (Ariel Futoransky), ramses.fernandez@fairgate.io (Ramses Fernandez), glaroton@dm.uba.ar (Gabriel Larotonda), sergio@fairgate.io (Sergio Demian Lerner)

1 Introduction

1.1 Verifying SNARK proofs on Bitcoin

Succinct non-interactive arguments of knowledge (SNARKs) have become the standard cryptographic tool for compressing arbitrary computation into a small, efficiently-verifiable certificate. Among them, Groth16 [Gro16] remains the dominant pairing-based SNARK in production use: it produces proofs of three group elements, has a per-circuit trusted setup, and is supported by mature tooling. A long line of work has built systems on top of Groth16, ranging from privacy-preserving cryptocurrency (Zcash) to optimistic and zero-knowledge rollups (Aztec, Polygon Zero, Citrea, BOB) to cross-chain bridges.

A natural question is whether Bitcoin, the largest and most security-conscious blockchain, can verify Groth16 proofs natively. The answer determines whether Bitcoin can support trust-minimised bridges to SNARK-based systems, light-client constructions, validity rollups, and a broad class of designated-verifier applications in which one party commits funds on the condition that another can present a proof of off-chain computation.

The question has, for the past two years, attracted intense engineering and cryptographic effort. The present paper contributes to this effort by giving the first UC-secure protocol for verifying standard Groth16 proofs on Bitcoin, providing concrete composability bounds in the GUC framework of Canetti, Dodis, Pass, and Walfish [CDPW07].

1.2 The pairing obstacle

The obstacle to native Bitcoin-side Groth16 verification is structural. Groth16's verification equation has the form

$$e(A_{\mathsf{in}}, B_{\mathsf{in}}) \stackrel{?}{=} e(\alpha\, G_1, \beta\, G_2) \cdot e\big(K_0 + x \cdot K_1,\, \gamma\, G_2\big) \cdot e(C_{\mathsf{in}}, \delta\, G_2), \tag{1}$$

where $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear pairing. Evaluating this equation requires computing pairings, which involve millions of $\mathbb{F}_p$ operations on a pairing-friendly curve. Bitcoin script provides no pairing precompile, no general-purpose $\mathbb{F}_p$ arithmetic, and no opcode for elliptic-curve scalar multiplication on a pairing-friendly curve. Verifying (1) natively in Bitcoin script is, by any reasonable accounting, infeasible.

There are two consensus-level responses to this obstacle and one cryptographic-protocol response. At the consensus level, one can propose a soft fork adding a pairing precompile (and the field- and curve-arithmetic opcodes that native verification would otherwise require). Bitcoin script has no such opcode, and proposals to extend its instruction set in this direction have not been adopted, for reasons that mix conservatism about Bitcoin's consensus rules with the engineering cost of supporting pairings in consensus-critical code. Alternatively, one can build a sidechain that supports pairings, such as Rootstock; this works but requires trusting the sidechain's security model, which differs from Bitcoin's.

The cryptographic-protocol response asks: can we verify a Groth16 proof off-chain, in a way that Bitcoin script can *enforce*? This question, which avoids both consensus changes and trust assumptions external to Bitcoin, is the question this paper, and the recent line of work we discuss next, addresses.

1.3 BitVM, optimistic execution, and garbled locks

The cryptographic-protocol response to the pairing obstacle began with BitVM [AAL+24], introduced by Robin Linus in 2023. BitVM observed that Bitcoin script suffices to evaluate *small fragments* of an arbitrary computation, and that a prover can commit to the entire computation off-chain while only being required to reveal a small fragment if the verifier

disputes the claim. The protocol is optimistic: an honest prover never needs to demonstrate the full computation on-chain, while a malicious prover faces a fraud-proof challenge that forces them to reveal an inconsistency or forfeit a stake.

BitVM2 [LAA+25] refined the dispute resolution protocol and demonstrated that SNARK verification could be expressed as a BitVM instance. The Citrea project later integrated BitVM2 with a complete Bitcoin rollup [cit25]. The BitVMX family [LAM+24] pushed further by introducing a virtual RISC-V CPU and compiling SNARK verifiers to RISC-V execution traces, with the dispute resolution game replaced by a more structured challenge-response protocol over the execution trace. As of mid-2024, BitVMX had demonstrated interactive SNARK verification on Bitcoin mainnet.

In parallel, a second approach emerged based on garbled circuits. Delbrag [Rub25] pioneered the use of Yao's garbled circuits to verify a Groth16 proof off-chain in a way enforceable on-chain: the prover garbles the verification circuit, the verifier evaluates it, and if the evaluation reaches a reject state the verifier learns a secret that authorises a slashing transaction on Bitcoin. The technique is sometimes called a "garbled lock", the funds locked on Bitcoin are conditionally released by the outcome of the garbled-circuit evaluation. Delbrag used Rubin's *Grug* technique for malicious security.

Subsequent work in this space has refined the construction along several dimensions. *BitVM3* [Lin25] proposed an RSA-based homomorphic encryption scheme for the garbled circuit, eliminating the need for a separate ZK proof of garbling correctness; an early version (BitVM3-RSA) was retracted after a security break, and the successor BitVM3s [Lin25] returned to Yao garbled circuits. *Glock* [Eag25] introduced a novel compact designated-verifier SNARK (a variant of Pari [DMS24]) instantiated over a binary elliptic curve, paired with cut-and-choose, verifiable secret sharing, and adaptor signatures; the result is a "garbled lock" whose fraud proof is a single Schnorr signature, a 550× reduction in on-chain footprint compared to BitVM2. Most recently, *Argo MAC* [EL26] introduced a new garbling primitive based on homomorphic MACs over elliptic curves, claiming over 1000× efficiency improvement for garbled SNARK verifiers as a building block for future constructions.

The common pattern across these protocols is striking: each combines off-chain SNARK proving and garbled-circuit evaluation with on-chain commitments and slashing scripts to obtain malicious security without modifying Bitcoin consensus. Each has been described in detail in workshop papers, blog posts, and preprints; several are deployed or near-deployment on mainnet. What is uniformly missing from this line of work is a full universal composability analysis. The constructions are presented as standalone protocols with malicious-security claims, typically argued game-theoretically or through standalone cryptographic reductions. None, to our knowledge, provides a UC functionality with concrete bounds, a composability theorem in the GUC framework, and an explicit simulator construction across all corruption patterns. This is the gap our paper closes.

1.4 Our contribution

We present $\Pi_{\mathsf{DV\text{-}Groth16}}$, to our knowledge, the first Bitcoin Groth16-verification protocol with a full UC analysis. It provides *designated evaluation and enforcement*: only the designated verifier can evaluate the garbled circuits before settlement, and only he can consume the slashing key. Concretely:

1. **A formal ideal functionality** $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$ that captures designated-verifier Groth16 verification on Bitcoin. The functionality models a three-tier outcome space (`refund`, `slash-fast`, `slash-commit`) reflecting the distinct on-chain pathways through which the verifier may obtain compensation under adversarial prover behaviour. We argue

that this three-tier modelling is essential: collapsing it to a binary `accept/slash` hides observable distinctions that the environment can detect via ledger state.

2. **A protocol $\Pi_{\mathsf{DV\text{-}Groth16}}$ realising $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$** in the GUC-hybrid model

$$\big(\mathcal{G}_{\mathsf{Ledger}}, \mathcal{G}_{\mathsf{oGG}}\,;\, \mathcal{F}_{\mathsf{GC}}, \mathcal{F}_{\mathsf{Beacon}}, \mathcal{F}_{\mathsf{HL}}^{+}, \mathcal{F}_{\mathsf{derive}}\big),$$

against static corruption. The bound is concrete and assembled from named primitive advantages:

$$\mathsf{Adv}_{\mathsf{in}}^{\mathsf{Groth16}/\mathcal{G}_{\mathsf{oGG}}}(\lambda, q, |C_{\mathsf{assertions}}|) + \mathsf{Adv}_{\mathsf{out}}^{\mathsf{sound},\mathcal{G}_{\mathsf{oGG}}}(\lambda, q, |\mathcal{C}_{\mathsf{verif}}|) + \mathsf{Adv}_{\pi_{\mathsf{SRS}}}^{\mathsf{ZK}}(\lambda) + \mathsf{Adv}_{\mathsf{BHR}}^{\mathsf{garb}}(\lambda)$$
$$+\mathsf{Adv}_{H}^{\mathsf{CR}} + \binom{s}{s/2}^{-1} + \varepsilon_{\mathcal{R}} + \mathsf{Adv}^{\mathsf{HL}+} + \mathsf{Adv}^{\mathsf{derive}},$$

where s is the cut-and-choose parameter, $\varepsilon_{\mathcal{R}}$ bounds the bias of the public-coin beacon, and $\mathsf{Adv}^{\mathsf{HL}+}$, $\mathsf{Adv}^{\mathsf{derive}}$ are the realisation advantages of the hash-lock and dispute functionalities.

3. **Three new cryptographic techniques**, developed to make the UC analysis go through:

   - *Native one-layer recursion with garbled scalarization.* Verification splits into a public heavy inner layer and a compact per-session outer layer: the outer relation is the inner verification equation itself, computed natively over a 2-chain, so the per-session SRS is fixed-size in the assertion circuit. The garbled circuit then verifies the *outer* proof in scalar form (Theorem 2), the only layer at which proof discrete logarithms ever become available to any party, reducing all in-circuit elliptic-curve work to three fixed-base multiplications bound to pre-reveal commitments.
   - *Complementary input authority.* Each garbled-circuit input wire is bound by exactly one of three mechanisms, by wire class: the prover's one-time-signed scalars, checked in-circuit against her pre-reveal commitments; seed-derived canonical labels, adjudicable from the public seed; and a dispute-contestable derived value. The union covers the input space under both corruption patterns without per-case constructions.
   - *Compatibility with standard Groth16.* Unlike Glock (which uses a modified Pari) or BitVMX-CPU (which compiles verifiers to RISC-V), our construction verifies Groth16 proofs produced by the standard prover with the standard verification equation. Existing Groth16 tooling, libraries, and proving infrastructure work without modification. The outer/inner Groth16 split (Section 1.5) is what enables this: the inner proof is the proof the user actually wants verified, natively for inner curves at the bottom of a 2-chain (e.g. BLS12-377), and via non-native emulation of the verifier circuit otherwise (Section 9), and the outer proof is a workhorse instantiated over a curve we choose for native recursion and Bitcoin-side garbling.

4. **A complete proof.** We give the simulator construction for all three corruption patterns ($\mathcal{C} = \emptyset$, $\{\mathtt{P}\}$, $\{\mathtt{V}\}$, $\{\mathtt{P},\mathtt{V}\}$), the full hybrid sequence in each case, and explicit bounds at each transition. The proof is self-contained: every lemma is followed by its proof in the body, and no appendix relegation is used.

1.5 Technical overview

We now sketch the construction informally, leaving formal definitions to Section 4 onward. The reader familiar with garbled-lock constructions will recognise the high-level shape and

can focus on the points where our design differs.

The outer/inner Groth16 split. The starting observation is that a Groth16 verification equation can be rewritten as an algebraic identity over discrete logarithms. If $A_{\mathsf{out}} = a\,G_1$, $B_{\mathsf{out}} = b\,G_2$, $C_{\mathsf{out}} = c\,G_1$ for scalars (a, b, c), and the SRS trapdoor scalars are $(\alpha, \beta, \gamma, \delta)$, then the verification equation for $(A_{\mathsf{out}}, B_{\mathsf{out}}, C_{\mathsf{out}})$ reduces to

$$a\,b \stackrel{?}{=} \alpha\beta + \gamma\,k_{\mathsf{pub}} + c\,\delta, \tag{2}$$

where k_{pub} collects the public-input verification-key terms, a scalar identity requiring no pairings, *provided one knows the scalars on the right-hand side.*

The catch is that for the inner proof no party ever knows them: the inner trapdoor is destroyed by the ceremony. They are knowable at exactly one layer, the verifier's per-session *outer* SRS, whose trapdoors he derives from a seed and reveals only after the prover has committed. We therefore split verification in two. The *outer* Groth16 instance proves the statement "the inner proof verifies": its circuit $\mathcal{C}_{\mathsf{verif}}$ is the inner Groth16 *verifier* itself, computed natively by placing the inner curve at the bottom of a one-layer 2-chain (BLS12-377 inside BW6-761), so $\mathcal{C}_{\mathsf{verif}}$, and with it the per-session SRS, is small and *independent of the assertion circuit.* The garbled circuit then checks the *outer* proof in the scalar form (2), together with three fixed-base consistency checks $aG_1 \stackrel{?}{=} A_{\mathsf{out}}$, $bG_2 \stackrel{?}{=} B_{\mathsf{out}}$, $cG_1 \stackrel{?}{=} C_{\mathsf{out}}$ that bind the revealed scalars to the points committed before the seed moved. The binding is temporal: the proof points are fixed while the prover is still in the soundness regime, and the scalars she signs afterwards are forced to be the discrete logarithms of exactly those points. All elliptic-curve work inside the garbled circuit is thereby confined to three fixed-base multiplications, amenable to off-chain handling.

The trapdoor SRS. If the outer Groth16 SRS were publicly generated, the prover would know all its scalars, and the soundness argument would collapse. We avoid this by having the verifier generate the outer SRS from a secret seed, kept hidden until Phase II of the protocol. While the seed is hidden, the prover sees only the group-form SRS and cannot break Groth16 soundness in the standard sense; this is the *epistemic invariant* of the construction, formalised within the observable generic group model of Bobolz, Farshim, Kohlweiss, and Takahashi [BFKT24]. After seed reveal, the prover computes the discrete logarithms of her own committed outer proof by direct arithmetic and signs them for the garbled evaluation; soundness is unaffected, having been anchored pre-reveal.

Cut-and-choose over garbled circuits. Given π_{out}, the prover constructs s garbled copies of the scalar-form checker $\mathcal{C}_{\mathsf{GC}}$, commits to all s via a Merkle root, and the verifier challenges a random subset of size $s/2$ for opening. The surviving $s/2$ circuits are then evaluated by the verifier on the prover's signed scalar package. By the malicious-secure cut-and-choose framework of Lindell–Pinkas [LP15], with the optimal balanced-cut bound $\binom{s}{s/2}^{-1}$ from [Lin16], the probability that a corrupt prover produces s garbled circuits all of whose openings verify but at least one surviving copy is malformed is at most $\binom{s}{s/2}^{-1}$.

Because the statement points $(A_{\mathsf{in}}, B_{\mathsf{in}}, C_{\mathsf{in}})$ are themselves committed inside the master commitment c_1 in Phase I, the inner proof is pinned directly by c_1; no separate hash-pin public input is needed in R_{outer}.

Cross-circuit input binding. The garbled circuit checks (2) together with the point-consistency checks, and cut-and-choose requires cross-circuit consistency: the prover must feed the *same* signed values $(a_{\mathsf{out}}, b_{\mathsf{out}}, c_{\mathsf{out}}, T_{\mathsf{ic}})$ to all $s/2$ surviving circuits, otherwise she could garble half the survivors on one assignment and half on another. (The seed-derived class-(B) values are identical across survivors by construction, so only the prover-signed inputs need binding.) We resolve this by committing the proof points, inner and outer, in

c_1 before the seed reveal, and signing each scalar once with a one-time signature shared across copies: collision resistance of H and signature unforgeability force every surviving circuit to evaluate on the same inputs, with no extra discrete-log assumption.

Complementary input authority and three-tier slashing. The garbled circuit's wires are bound by complementary mechanisms, one per wire class: the prover's signed scalars, checked *in-circuit* against her pre-reveal commitments; seed-derived canonical labels for the raw trapdoor wires; and a dispute-adjudicated derived value. Script-checkable evidence, commitment inconsistency, signature equivocation, a wrong-bit label reveal, a lost dispute, triggers `slash-commit`, the Tier-2 slashing path, in which the verifier posts explicit evidence on-chain. A direct circuit-reject (the prover's scalars do not satisfy (2)) triggers `slash-fast`, the Tier-1 path, in which the verifier learns the slashing key directly via the reject label and signs the slashing transaction without posting explicit evidence. An honest evaluation produces `refund`. The ideal functionality $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$ models all three pathways and distinguishes the simulator's obligations in each.

Composability. The protocol is shown to realise $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$ in the GUC-hybrid model, with the bound assembled from the advantages of the underlying primitives. The use of the observable generic group $\mathcal{G}_{\mathsf{oGG}}$ [BFKT24] sidesteps the algebraic-group-model versus random-oracle-model trade-off for Groth16 soundness: $\mathcal{G}_{\mathsf{oGG}}$ provides simulation-extractability for Groth16 in a UC-friendly form without requiring programmable random oracles for the SNARK itself. The result composes with concurrent sessions via the GUC composition theorem [CDPW07] under standard assumptions; multi-session bounds scale linearly in the session count.

1.6 Limitations and scope

The result is for static corruption: the set of corrupt parties is fixed at session start. Extending to adaptive corruption requires additional machinery (non-committing encryption in the garbled circuits, a programmable random oracle for hash commitments, equivocable Pedersen via the simulator's trapdoor); the construction is compatible with such extensions but we leave the full adaptive proof to future work.

The protocol is specified for single public-input statements ($n = 1$); extension to multi-input statements is straightforward and leaves the commitment structure unchanged (only the public-input portion of the inner verification key grows with n). The cut-and-choose parameter s controls the soundness amplification $\binom{s}{s/2}^{-1}$; choosing s to achieve $2^{-\lambda}$ soundness requires $s \approx \lambda + O(\log \lambda)$ total circuits ($s/2$ surviving), which dominates the off-chain bandwidth. We discuss concrete instantiations in Section 9.

We do not address the engineering question of how to instantiate $\mathcal{F}_{\mathsf{HL}}^{+}$ (hash-locks with equivocation slashing and canonical-bit adjudication), $\mathcal{F}_{\mathsf{derive}}$ (the dispute path for the derived input T_{ic}), and $\mathcal{F}_{\mathsf{Beacon}}$ on Bitcoin script. Standard candidates exist (Lamport/Winternitz hash-locks, BitVMX/ESSPI-style dispute protocols, commit-reveal beacons); the adjudication and dispute *interfaces* are stated as part of the functionalities, and the realisation-specific advantages $\mathsf{Adv}^{\mathsf{HL}+}$, $\mathsf{Adv}^{\mathsf{derive}}$, $\varepsilon_{\mathcal{R}}$ are carried explicitly through the bound. The cryptographic content of the paper is parametric in these realisations.

1.7 Paper organisation

Section 2 positions the work against prior lines: Bitcoin-side SNARK verification, designated-verifier SNARKs, cut-and-choose over garbled circuits, and UC on Bitcoin. Section 3 fixes notation and recalls the cryptographic primitives. Section 4 defines the ideal functionality $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$, the hybrid model, and states the main UC theorem. Section 5 develops the building blocks: outer Groth16 SRS structure, garbled circuits, structural binding

lemmas, and the scalarization of the outer verification equation. Section 6 presents the protocol $\Pi_{\mathsf{DV\text{-}Groth16}}$ across its three phases, together with honest-party guarantees. Section 7 constructs the simulator for each corruption pattern, and Section 8 carries out the hybrid argument and proves the main theorem. Section 9 discusses limitations, alternative instantiations, and future work.

2 Related work

The construction sits in four related lines of research, which we discuss in turn: Bitcoin-side SNARK verification (Section 2.1), designated-verifier SNARKs (Section 2.2), cut-and-choose over garbled circuits (Section 2.3), and UC on Bitcoin (Section 2.4).

2.1 Bitcoin-side SNARK verification

The problem of enforcing the outcome of an off-chain SNARK verification on Bitcoin script, without consensus-level changes, has produced a vigorous line of construction work over the past two years.

Optimistic execution. BitVM [AAL+24] introduced the optimistic-execution paradigm: a prover commits off-chain to the entire computation, and an on-chain dispute game forces them to reveal a specific fragment if the verifier disputes. BitVM2 [LAA+25] refined the dispute resolution protocol and demonstrated that SNARK verification could be expressed as a BitVM instance. BitVMX [LAM+24] pushed further by introducing a virtual RISC-V CPU and compiling SNARK verifiers to RISC-V execution traces, with the dispute resolution game replaced by a more structured challenge-response protocol over the execution trace. The Citrea project [cit25] integrated BitVM2 into a complete Bitcoin rollup; subsequent infrastructure work [LJF25; AMLC25] addressed scalability and bridge composition. The optimistic line offers practical deployability but pays in on-chain dispute complexity and lacks formal UC analysis.

Garbled-circuit SNARK verification. A second approach uses Yao's garbled circuits to perform the SNARK verification off-chain in a way enforceable on-chain. Delbrag [Rub25] pioneered this for Groth16: the prover garbles the verification circuit, the verifier evaluates it, and if the evaluation reaches a reject state the verifier learns a slashing secret. BitVM3 [Lin25] proposed an RSA-based homomorphic-encryption garbling scheme; an early version was retracted after a security break, and BitVM3s [Lin25] returned to Script-based proofs. BitVMX-GC [LAM+24] (the garbled-circuit plug-in for the BitVMX platform) combined garbled circuits with a designated-verifier SNARK and a separate ZK proof of garbling correctness. Glock [Eag25] introduced a compact designated-verifier SNARK (a variant of Pari [DMS24]) instantiated over a binary elliptic curve, paired with cut-and-choose, verifiable secret sharing, and adaptor signatures. Argo MAC [EL26] most recently introduced a new garbling primitive based on homomorphic MACs over elliptic curves, designed as a building block for future garbled SNARK verifiers.

Our positioning. Our construction sits in this garbled-circuit family. We share with Delbrag, BitVMX-GC, BitVM3s, Glock, and Argo MAC the basic strategy of garbling the outer SNARK verification circuit and using cut-and-choose for soundness amplification. We diverge from prior work along two axes.

First, we provide *full UC analysis with concrete composability bounds in a stated GUC-hybrid model*. To our knowledge, none of the prior garbled-lock constructions has been accompanied by a UC functionality, a simulator construction across corruption patterns, and a hybrid-sequence proof with primitive-by-primitive accounting. The UC framework matters in this design space because the constructions are intended to compose with other

Bitcoin-layer protocols (bridges, rollups, MuSig2 multisignatures, hash-locks), and UC composability gives a clean compositional story rather than ad-hoc reductions.

Second, our protocol verifies *standard* Groth16 proofs without modification. Glock uses a modified Pari SNARK; BitVMX-CPU compiles SNARK verifiers to RISC-V; Argo MAC uses custom garbling primitives that are not yet integrated into a complete protocol. By contrast, $\Pi_{\mathsf{DV\text{-}Groth16}}$ accepts the proofs produced by the standard Groth16 prover. Existing Groth16 tooling, libraries, and trusted-setup infrastructure (jellyfish, snarkjs, etc.) work without modification. This compatibility comes at a cost: the outer Groth16 proof is itself a pairing-based proof of a scalar circuit, doubling the proving work for the prover; we view this as an acceptable trade given the deployability benefit.

Bridges and applications. The garbled-lock construction is the cryptographic core of several Bitcoin-bridge designs. FLEX [LF25] provides capital-efficient optimistic bridges with on-demand security bonds; TOOP [FBF+25] extends to ownership transfer with bridge composition; Union [AMLC25] provides a 1-of-n-honest multi-party bridge. The BitLayer BitVM Bridge [Bit25] is an early deployment of the BitVM2 line. Our construction is application-independent: $\Pi_{\mathsf{DV\text{-}Groth16}}$'s designated-verifier semantics map naturally to the verifier role in any of these bridge protocols, with the bridge providing the deposit-and-slashing flow that we model abstractly via $\mathcal{G}_{\mathsf{Ledger}}$.

2.2 Designated-verifier SNARKs

Designated-verifier (DV) zero-knowledge proofs allow a prover to convince a specific verifier of a statement while preventing all other parties from being convinced, even given the proof transcript [JSI96]. The motivation is that the verifier's secret key (or analogous shared secret) acts as a trapdoor: anyone with the secret can simulate the proof, so a third party cannot distinguish a real proof from a simulation. The DV semantics map naturally to slashing-based protocols on a public ledger, where the verifier's privileged information is the seed underlying the trapdoor SRS.

The literature on *DV-SNARKs*, succinct, non-interactive arguments of knowledge in the designated-verifier setting, includes constructions based on KZG commitments, dual-mode commitments, and pairing-based simulation. These constructions optimise for proof size and verifier efficiency in standalone settings. Our work differs in three ways. First, we do not introduce a new DV-SNARK primitive: we apply existing Groth16 in a designated-verifier setting via the cut-and-choose plus trapdoor SRS pattern. Second, we target Bitcoin-script enforceability specifically, whereas prior DV-SNARK constructions target generic standalone deployment. Third, we provide UC analysis, which prior DV-SNARK constructions typically do not.

UC-secure SNARKs more broadly. Concurrent with the Bitcoin-side construction work, there has been substantial progress on UC-secure SNARK frameworks. Chiesa–Fenzi [CF24] establish unconditional UC-security for zkSNARKs in the ROM. The kkk-gogg framework of Bobolz–Farshim–Kohlweiss–Takahashi [BFKT24] provides simulation-extractability for Groth16 in the observable generic group model without random-oracle programming. Earlier work includes the UC-Groth16 constructions of [BKSV21] and [ARS20], which require a separate simulation-extractable wrapper. We adopt the kkk-gogg framework: it gives the cleanest composition story for our setting because the inner and outer Groth16 proofs share the same generic-group model, and the framework's observability interface supports both extraction (for corrupt-prover analysis) and simulation (for both-honest and corrupt-verifier analysis) without rewinding.

2.3 Cut-and-choose over garbled circuits

Garbled circuits, due to Yao [Yao82; Yao86], provide a means for one party to evaluate a function on encrypted inputs without learning intermediate computation values. The malicious-secure variants relevant to our setting require additional machinery to prevent a corrupt garbler from producing an incorrect garbling: cut-and-choose, in which the garbler produces multiple garbled copies of the same circuit and opens a random subset for verification, is the standard remedy [LP15; Lin16].

The Bellare–Hoang–Rogaway framework [BHR12] formalises the security properties of garbling schemes (privacy, obliviousness, authenticity) in a unified way; we adopt the BHR notation throughout and rely specifically on authenticity for the slashing-key binding in our capsule construction (Lemma 7).

Our use of cut-and-choose differs from the generic 2PC setting in two respects. First, the circuit being garbled is the small, fixed scalar-form checker $\mathcal{C}_{\mathsf{GC}}$ (Definition 4), the outer-verification circuit $\mathcal{C}_{\mathsf{verif}}$ itself is proven, never garbled, not a generic user-supplied circuit. This means circuit-size optimisations specific to pairing equation verification are available (we exploit them in the QAP structure of $\mathcal{C}_{\mathsf{verif}}$). Second, our verifier is designated, not adversarial: cut-and-choose defends specifically against a malicious prover, while the verifier's resistance is separately bounded by Lemma 8.

2.4 UC on Bitcoin

The universal composability framework, due to Canetti [Can01], and its generalised variant GUC [CDPW07] are the standard cryptographic-security frameworks for protocols composing arbitrarily with their environment. Composability on Bitcoin requires modelling the ledger itself as a UC functionality; the foundational treatment is [BMTZ24], which models Bitcoin script verification, UTXO posting, and time-locks as ideal functionalities. Subsequent work has refined the model: practical UC with global random oracles [[CJS14]], UC issues and resolutions [CS25], UC multisignatures [BDPT25] (used in MuSig2-style aggregation).

Our hybrid model is a direct application of this body of work: we use $\mathcal{G}_{\mathsf{Ledger}}$ [BMTZ24] as the global Bitcoin ledger functionality and $\mathcal{G}_{\mathsf{oGG}}$ [BFKT24] as the observable generic-group functionality, both shared across protocol sessions. Beyond this standard composition setup, our paper does not introduce new framework machinery; rather, we apply the existing framework to a specific protocol that has not previously been analysed in it.

3 Preliminaries

3.1 Notation

Let λ denote the security parameter. All adversaries are non-uniform probabilistic polynomial-time machines unless stated otherwise. A function $f(\lambda)$ is *negligible*, written $f = \mathsf{negl}(\lambda)$, if it eventually decreases faster than any inverse polynomial. We write $x \leftarrow D$ to denote sampling x from a distribution D, and $x \leftarrow X$ for sampling uniformly from a set X. For a randomised algorithm $\mathcal{A}$, we write $y \leftarrow \mathcal{A}(x)$ for the random variable obtained by running $\mathcal{A}$ on input x with fresh randomness.

We use $[n]$ for $\{1, \ldots, n\}$ and $\binom{n}{k}$ for the binomial coefficient. Concatenation of bit-strings is denoted $\|$. We use $\oplus$ for bitwise XOR. The notation $\mathsf{lsb}_j(\cdot)$ denotes the j-th least significant bit of a bit-string interpreted as an integer.

Throughout the paper, we use the following two-party-protocol naming convention: P

(Prover, often referred to as Alice in narrative passages) and $\mathtt{V}$ (Verifier, Bob). Corruption sets are $\mathcal{C} \subseteq \{\mathtt{P}, \mathtt{V}\}$.

3.2 Bilinear groups and Groth16

We work over a pairing-friendly elliptic curve with asymmetric Type-3 pairing $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, where $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are cyclic groups of prime order p, with generators $G_1 \in \mathbb{G}_1$, $G_2 \in \mathbb{G}_2$, and $g_T = e(G_1, G_2) \in \mathbb{G}_T$. All scalar arithmetic is over the field $\mathbb{F}_p$. The bilinear pairing satisfies $e(aG_1, bG_2) = g_T^{ab}$ for $a, b \in \mathbb{F}_p$. We assume the standard discrete-log assumption holds in each group: for $\mathbb{G} \in \{\mathbb{G}_1, \mathbb{G}_2\}$, the advantage of any PPT adversary in computing a from aG is bounded by $\mathsf{Adv}^{\mathsf{DL}}_{\mathbb{G}}(\lambda) = \mathsf{negl}(\lambda)$.

Throughout the paper we use Groth's pairing-based SNARK [Gro16] for an NP relation $\mathcal{R}$, henceforth Groth16. The proof system $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ operates on a quadratic arithmetic program (QAP) representation of the relation circuit; a per-circuit structured reference string (SRS) is produced by Setup. The verifier checks the single pairing-product equation $e(A, B) \stackrel{?}{=} e(\alpha G_1, \beta G_2) \cdot e\big(K_0 + x_1 K_1 + \cdots + x_n K_n,\, \gamma G_2\big) \cdot e(C, \delta G_2)$, where the proof is $\pi = (A, B, C) \in \mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{G}_1$, the public inputs are $(x_1, \ldots, x_n) \in \mathbb{F}_p^n$, the SRS scalars are $(\alpha, \beta, \gamma, \delta) \in \mathbb{F}_p^4$, and $K_0, K_1, \ldots, K_n \in \mathbb{G}_1$ are derived from the verification key. We take $n = 1$ for simplicity, writing x for the single public input. Knowledge-soundness of Groth16 is established in the generic group model [Gro16] under the q-PKE and q-SDH assumptions; we recall a stronger UC-flavoured form in Section 3.3.

3.3 The observable generic group functionality $\mathcal{G}_{\mathsf{oGG}}$

To get a UC-style simulation-extractability statement for Groth16, we work in the *observable generic group* model of Bobolz, Farshim, Kohlweiss, and Takahashi [BFKT24]. The framework is captured by a global functionality $\mathcal{G}_{\mathsf{oGG}}$ which exposes a generic-group oracle for the pairing-friendly curve together with extraction and simulation interfaces accessible only to the UC environment's simulator. We summarise here only what we need; the full treatment is in the original paper.

Remark 1. **Why $\mathcal{G}_{\mathsf{oGG}}$.** The Groth16 soundness proof of [Gro16] is in the standalone generic group model and does not, by itself, provide the simulation-extractability or zero-knowledge simulator required for UC-style composability. The natural alternatives, algebraic group model with random oracles, or programmable-ROM constructions, introduce their own complications when composed across multiple sessions or with other UC building blocks. $\mathcal{G}_{\mathsf{oGG}}$ provides both extractability and simulation in a single global-functionality interface that composes cleanly under GUC composition.

We extract three concrete properties of $\mathcal{G}_{\mathsf{oGG}}$ used throughout the paper:

1. **Simulation-extractability.** For Groth16 with respect to any NP relation $\mathcal{R}$, given a verifying proof π, the simulator may invoke the $\mathcal{G}_{\mathsf{oGG}}$ extraction interface to recover w such that $(x, w) \in \mathcal{R}$, except with probability $\mathsf{Adv}^{\mathsf{Groth16}/\mathcal{G}_{\mathsf{oGG}}}(\lambda, q, |C|)$, where q is the number of $\mathcal{G}_{\mathsf{oGG}}$ queries made by the environment and $|C|$ is the size of the circuit underlying the QAP.

2. **Zero-knowledge simulation.** (2a) *Zero-knowledge (true statements).* For any true $(x, w) \in R$, the simulator produces a proof whose distribution is indistinguishable from the honest prover's, with advantage at most $\mathsf{Adv}^{\mathsf{Groth16}/\mathcal{G}_{\mathsf{oGG}}}$. (2b) *Designated-verifier trapdoor simulation (any statement).* Given the SRS trapdoor (held by $\mathtt{V}$, and recovered by $\mathcal{S}$ via $\mathcal{G}_{\mathsf{oGG}}$ observability of $\mathtt{V}$'s queries), the simulator produces a proof verifying under vk on *any* statement, true or false. Only trapdoor-holders have (2b); a corrupt prover does not, which is what preserves soundness.

3. **Explainability (trapdoor-held proofs only).** For a proof simulated under a trapdoor that $\mathcal{S}$ holds,in this protocol the *outer* proof, whose SRS trapdoor $\mathcal{S}$ recovers via $\mathcal{G}_{\mathsf{oGG}}$ observability of $\mathtt{V}$'s queries (property (2b)),on later receiving a witness w, $\mathcal{S}$ can output randomness that explains the proof as a Groth16 proof on (x, w). This does *not* extend to the *inner* proof, whose SRS trapdoor is destroyed at setup (Definition 1) and which $\mathcal{S}$ obtains by oblivious sampling, leaving no trapdoor and no discrete-logarithm representation to invert.

We use $\mathcal{G}_{\mathsf{oGG}}$ as a global setup functionality throughout the paper. All security claims are stated in the $\mathcal{G}_{\mathsf{oGG}}$-hybrid model, and the bound contains the advantage term $\mathsf{Adv}^{\mathsf{Groth16}/\mathcal{G}_{\mathsf{oGG}}}$ wherever a Groth16 extraction or simulation step is invoked.

3.4 Universal Composability

We work in the Universal Composability framework of Canetti [Can01] and its Generalized variant (GUC) of Canetti, Dodis, Pass, and Walfish [CDPW07], which permits global setup functionalities accessible across protocol sessions. We recall only the elements directly relevant to our analysis; the standard references provide the full framework.

A protocol Π *GUC-emulates* an ideal functionality $\mathcal{F}$ if for every PPT real-world adversary $\mathcal{A}$ there exists a PPT ideal-world simulator $\mathcal{S}$ such that for every PPT environment $\mathcal{Z}$, the real-world execution $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$ and the ideal-world execution $\mathsf{EXEC}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$ are computationally indistinguishable. When the security claim holds only against a restricted class of adversaries (e.g., those that corrupt parties at session start), the corresponding restriction is part of the claim.

In this paper, all security claims are with respect to *static* corruption: the set $\mathcal{C} \subseteq \{\mathtt{P}, \mathtt{V}\}$ is chosen by the adversary at session start and remains fixed throughout. The corresponding ideal-world security parameter is the distinguishing advantage of $\mathcal{Z}$ between $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$ and $\mathsf{EXEC}_{\mathcal{F}_{\mathsf{DV\text{-}Groth16}},\mathcal{S},\mathcal{Z}}$, bounded by a function of the underlying advantages (Section 4).

A *hybrid model* declares a set of ideal functionalities that the protocol is allowed to invoke. We write $(\mathcal{F}_1, \ldots, \mathcal{F}_k)$-hybrid model to denote that Π has access to $\mathcal{F}_1, \ldots, \mathcal{F}_k$ as black boxes; the protocol's own messages, and its interaction with $\mathcal{Z}$, are written explicitly. Global functionalities (which are shared across sessions) are written with an over-bar: $\mathcal{G}_{\mathsf{Ledger}}$ for the ledger, $\mathcal{G}_{\mathsf{oGG}}$ for the observable generic group. Setup functionalities are written without the bar.

The *GUC composition theorem* [CDPW07] states that if Π GUC-emulates $\mathcal{F}$ in some hybrid model with global functionalities $\mathcal{G}_1, \ldots, \mathcal{G}_l$, and each invoked $\mathcal{F}_j$ is itself GUC-realised by some sub-protocol Π_j in a compatible global-functionality model, then the composed protocol GUC-emulates the composed functionality with the union of the global functionalities. We invoke composition in two places: when arguing that the protocol composes with concrete realisations of $\mathcal{F}_{\mathsf{HL}}^{+}$, $\mathcal{F}_{\mathsf{derive}}$, $\mathcal{F}_{\mathsf{Beacon}}$, etc., and when arguing that multi-session execution inherits per-session security with a linear bound degradation.

3.5 Cryptographic primitives

Hash function and random oracle modelling. We use a single hash function $H\colon \{0,1\}^* \to \{0,1\}^{2\lambda}$, modelled as a random oracle in the security analysis. The function plays three distinct roles in the protocol:

- *Collision-resistant binding*, in the master commitment c_1, the Merkle root $\mathsf{root}_{\mathsf{GC}}$, and the slashing-key pin $h = H(k)$. The associated advantage is $\mathsf{Adv}_H^{\mathsf{CR}}(\lambda) = \mathsf{negl}(\lambda)$.

- *Hiding commitment*, in c_1, where the nonce ρ_1 combined with the random-oracle assumption on H provides statistical hiding.

- *One-wayness*, in the hash-lock commitments of the canonical-label construction (Section 5.4) and in the slashing-key pin. The one-wayness advantage is $\mathsf{Adv}^{\mathsf{OW}}_{H}(\lambda)$ and is absorbed into the hash-lock advantage $\mathsf{Adv}^{\mathsf{HL}+}$ in the global bound.

Domain separation across the three roles is handled by explicit label prefixes (`cl`, `mask`, `wire`, `alpha`, `beta`, etc.). We treat collisions across distinct label prefixes as cryptographically negligible under the random-oracle assumption.

Seed derivation. The per-session outer trapdoors are derived as $H(\mathsf{seed} \,\|\, \texttt{label})$, with H modelled as a random oracle *for this derivation only*; the single programming step appears inside the soundness reduction of Lemma 3. (Earlier versions used a PRF here; the random-oracle form is what the pre-reveal soundness reduction requires.)

Garbled circuits. A garbling scheme [BHR12] is a tuple $(\mathsf{Garble}, \mathsf{Eval}, \mathsf{Ver})$ that, given a Boolean circuit C, produces a garbled circuit $\widetilde{C}$, input-encoding labels $\{\mathsf{Lab}_{j,b}\}_{j,b}$ for each wire j and bit $b \in \{0,1\}$, and an evaluation algorithm that on a single label per wire outputs the corresponding output label. We require BHR-style malicious-secure garbling: privacy (the garbled circuit reveals nothing about its underlying gates), authenticity (an evaluator cannot produce an output label not consistent with the circuit's computation on the given inputs), and obliviousness. The associated advantage is $\mathsf{Adv}^{\mathsf{garb}}_{\mathsf{BHR}}(\lambda) = \mathsf{negl}(\lambda)$.

Hash commitments. The protocol's on-chain commitments (c_1, $\mathsf{com}_r, \mathsf{com}_s$, the garbled-circuit root $\mathsf{root}_{\mathsf{GC}}$) are hash commitments evaluated on $\mathcal{G}_{\mathsf{Ledger}}$: a commitment to a value v with nonce ρ is $H(v \,\|\, \rho)$. Binding reduces to collision-resistance of H ($\mathsf{Adv}^{\mathsf{CR}}_{H}$) and hiding to its one-wayness; the protocol is parametric in the Bitcoin-script realisation (Lamport/Winternitz/BitVMX hash-locks).

Hash-locks. A hash-lock is a one-time signature scheme in which a one-bit signature on a wire (i,j) with value $b \in \{0,1\}$ consists of revealing a preimage of $\mathsf{hl}_{i,j,b}$, the corresponding hash-lock commitment. The associated UC functionality $\mathcal{F}^{+}_{\mathsf{HL}}$, which additionally carries equivocation-slash and canonical-bit adjudication interfaces, is defined in Section 5.4. Concrete realisations include Lamport [Lam79] and Winternitz [Mer87] one-time signatures, or BitVMX-style hash-locks; the protocol is parametric in the realisation. The combined hash-lock advantage is denoted $\mathsf{Adv}^{\mathsf{HL}+}$ and absorbs $\mathsf{Adv}^{\mathsf{OW}}_{H}$ for the underlying preimage one-wayness together with the realisation cost of the two additional interfaces.

Public-coin beacon. The protocol requires a source of public randomness for the cut-and-choose challenge. This is captured by a beacon functionality $\mathcal{F}_{\mathsf{Beacon}}$ (Section 5.7) which delivers a uniformly-random string after a fixed delay. Concrete realisations include commit-reveal coin tossing, block-hash-based beacons with confirmation depth, and external randomness beacons (e.g., Drand). The bias of any concrete realisation is bounded by $\varepsilon_{\mathcal{R}}$, which appears in the global security bound.

Bitcoin ledger. On-chain state is modelled by the global ledger functionality $\mathcal{G}_{\mathsf{Ledger}}$ of Badertscher, Maurer, Tschudi, and Zikas [BMTZ24] in its UC formulation. The functionality models UTXO posting, script verification (under the subset of Bitcoin script the protocol uses), and the basic temporal structure (blocks, confirmation depth). We use $\mathcal{G}_{\mathsf{Ledger}}$ as a global functionality, shared across protocol sessions; this is the natural fit since Bitcoin is genuinely shared infrastructure rather than per-session setup.

4 The DV-Groth16 functionality and main result

4.1 The ideal functionality $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$

The ideal functionality is the abstract specification of what the protocol achieves. It exposes a designated-evaluator interface: a single verifier `V` learns the acceptance bit of the inner Groth16 proof, while the on-chain outcome is public.

The functionality interacts with a prover `P`, a designated verifier `V`, an adversary $\mathcal{A}$, and an environment $\mathcal{Z}$. It is parameterised by an assertion circuit $C_{\mathsf{assertions}}$, a deposit value D, and timing parameters $T_2 < T_1$ governing the seed-reveal and refund windows. Informally, `P` submits a witness; $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$ checks whether the witness satisfies $C_{\mathsf{assertions}}$; `V` triggers evaluation; $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$ announces an on-chain outcome in {`refund`, `slash-fast`, `slash-commit`} and privately notifies `V` of the acceptance bit b. The formal specification is given in Algorithm 1; the informal guarantees are stated in Remark 2.

Algorithm 1 Ideal functionality $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$

State: $(\mathsf{sid}, \mathtt{phase}, x, w, b, \mathsf{out}, \tau)$ with `phase` being $\perp$, `init`, `witnessed`, `evaluated`, or `closed`; each message first time τ from $\mathcal{G}_{\mathsf{Ledger}}$. Parameters $(C_{\mathsf{assertions}}, D, T_2 < T_1)$.

I. Init. On $(\mathtt{init}, \mathsf{sid}, x, D)$ from `P` with `phase` $= \perp$:

1: `phase` $\leftarrow$ `init`; record $(x, D, \tau_0 := \tau)$; send $(\mathtt{init\text{-}leak}, \mathsf{sid}, x, D)$ to $\mathcal{A}$ and $(\mathtt{init\text{-}notify}, \mathsf{sid}, x, D)$ to `V`.

II. Prove. On $(\mathtt{prove}, \mathsf{sid}, w)$ from `P` with `phase` $=$ `init`:

2: $b \leftarrow C_{\mathsf{assertions}}(x; w)$; `phase` $\leftarrow$ `witnessed`. Leak to $\mathcal{A}$: $(\mathtt{prove\text{-}leak}, \mathsf{sid}, b, w)$ if `P` is corrupt, or if `P` is honest and $b = 0$ (a failing assignment carries no witness to hide; Remark 30); otherwise $(\mathtt{prove\text{-}leak}, \mathsf{sid}, b)$ only ($b = 1$: witness-hiding is by zero-knowledge).

III. Evaluate. On $(\mathtt{evaluate}, \mathsf{sid})$ from `V` with `phase` $\in \{\mathtt{witnessed}, \mathtt{init}\}$. If `phase` $=$ `witnessed`, set out by:

3: `P` honest: out $\leftarrow$ `refund` if $b = 1$, else `slash-fast`.

4: `P` corrupt, $b = 1$: $\mathcal{A}$ picks out $\in \{\mathtt{refund}, \mathtt{slash\text{-}fast}, \mathtt{slash\text{-}commit}\}$.

5: `P` corrupt, $b = 0$: $\mathcal{A}$ picks out $\in \{\mathtt{slash\text{-}fast}, \mathtt{slash\text{-}commit}\}$.

6: Else if `phase` $=$ `init` and `P` corrupt: $b \leftarrow \perp$; $\mathcal{A}$ picks out $\in \{\mathtt{slash\text{-}commit}, \perp\}$ (witness-less path).

7: If out $\neq \perp$: send $(\mathtt{acceptance\text{-}bit}, \mathsf{sid}, b)$ to `V`; `phase` $\leftarrow$ `evaluated`; announce $(\mathtt{outcome}, \mathsf{sid}, \mathsf{out})$ to $\mathcal{G}_{\mathsf{Ledger}}$; settle D to `V` if out $\neq$ `refund`, else to `P`.

IV. Refund timeout. If `phase` $\in \{\mathtt{init}, \mathtt{witnessed}\}$ and $\tau \geq \tau_0 + T_1$:

8: out $\leftarrow$ `refund`; `phase` $\leftarrow$ `evaluated`; announce $(\mathtt{outcome}, \mathsf{sid}, \mathtt{refund})$ to $\mathcal{G}_{\mathsf{Ledger}}$; settle D to `P`.

V. Transcript leakage ($\mathcal{S}$-only). $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$ authorises $\mathcal{S}$ to post simulated transcript fragments on $\mathcal{G}_{\mathsf{Ledger}}$ consistent with out: on `leak-phase1`, x and the proof shapes; on `leak-phase2`, the simulated c_1, the UTXO2 seed commitment, and the UTXO3 scalar openings; on `leak-phase3`, the outcome script execution. $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$ produces no transcript content itself.

VI. Corruption handling. Static: $\mathcal{A}$ declares $\mathcal{C} \subseteq \{\mathtt{P}, \mathtt{V}\}$ at session start.

9: `P` $\in \mathcal{C}$: $\mathcal{A}$ may supply any w via `prove` and learns (w, b), or abstain.

10: `V` $\in \mathcal{C}$: $\mathcal{A}$ receives $(\mathtt{acceptance\text{-}bit}, \mathsf{sid}, b)$; may decline `evaluate`, forcing timeout IV.

11: $\mathcal{C} = \emptyset$: $\mathcal{S}$ receives only the clause-II `prove-leak` (b, and w when $b = 0$); nothing further.

12: $\mathcal{C} = \{\mathtt{P}, \mathtt{V}\}$: $\mathcal{A}$ drives both parties; the protocol trivially simulates.

VII. Delivery. All $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$ outputs to honest parties are deliverable through $\mathcal{A}$, which may delay but not forge them.

Remark 2. $\mathcal{F}_{\text{DV-Groth16}}$ guarantees:

- An invalid witness yields $\texttt{out} \in \{\texttt{slash-fast}, \texttt{slash-commit}\}$ on-chain with probability 1 in the ideal world, as long as honest `V` calls `evaluate`.
- A valid witness with honest `P` and honest `V` yields $\texttt{out} = \texttt{refund}$; a corrupt `P` holding a valid witness may nonetheless be slashed (Remark 4).
- The acceptance bit b is revealed only to `V` (or to $\mathcal{A}$ if $\texttt{V} \in \mathcal{C}$). Third-party observers of the ledger see only `out` in $\{\texttt{refund}, \texttt{slash-fast}, \texttt{slash-commit}\}$, which reveals b only one-sidedly ($\texttt{refund} \Rightarrow b = 1$; a slashing outcome does not imply $b = 0$ when `P` is corrupt) and never the witness.
- The witness w is never visible to $\mathcal{Z}$ unless $\texttt{P} \in \mathcal{C}$. When `P` is honest and `V` is corrupt, w is passed to the ideal-world simulator $\mathcal{S}$ *only in the* $b = 0$ *branch* of clause II of Algorithm 1 (a false statement, whose invalid assignment carries no witness to hide; likewise in the both-honest case, where $\mathcal{S}$ must reproduce honest `P`'s failing artifacts); for a true statement ($b = 1$) only the bit is disclosed and $\mathcal{S}$ simulates the proofs by zero-knowledge, which is what establishes witness-hiding against a corrupt `V` (Remark 30). In all cases $\mathcal{S}$ does not leak w to $\mathcal{Z}$.
- Only $\mathcal{F}_{\text{DV-Groth16}}$ can instruct $\mathcal{G}_{\text{Ledger}}$ to move D from `P`'s escrow; $\mathcal{A}$ cannot directly steal the deposit.

It does *not* guarantee:

- Unconditional ideal-world slashing soundness in the real protocol: soundness is by-definition in the ideal world, but the real protocol matches this only up to the distinguishing advantage in Theorem 1.
- Confidentiality of the statement x: x is a public input and is visible to $\mathcal{Z}$ through `init-leak`.
- Privacy of the inner proof π_{in}: it is visible on-chain in the real world (to match the ideal-world public outcome, $\mathcal{S}$ must produce a simulated π_{in} via the $\mathcal{G}_{\text{oGG}}$ interface).

Remark 3. The real protocol spends UTXO1 through different script paths depending on the failure mode: `slash-fast` uses the hash-lock-k path, `slash-commit` uses a script condition proving commitment inconsistency against c_1, and `refund` uses the time-lock-T_1 path. These are three distinct transaction shapes on-chain, so $\mathcal{Z}$ can distinguish them by reading $\mathcal{G}_{\text{Ledger}}$. The ideal functionality therefore must announce which of the three occurred, not just a binary slash-or-refund. The simulator (Section 7) produces the corresponding script execution based on $\mathcal{F}_{\text{DV-Groth16}}$'s announced `out`.

Remark 4. In Algorithm 1 clause III, when `P` is corrupt we allow $\mathcal{A}$ to choose the outcome, subject only to the restriction that `refund` requires $b = 1$. This reflects the real adversary's freedom. With an invalid witness, a corrupt prover may fail through the circuit-reject path (`slash-fast`) or through script-checkable evidence (`slash-commit`); the choice is hers, via the inputs she malforms. With a *valid* witness, a corrupt prover can still self-slash: signing scalars inconsistent with her committed proof points fails checks 1–3 of $\mathcal{C}_{\text{GC}}$ and releases k (Tier 1), and equivocating a one-time signature is Tier-2 evidence, so the functionality must not force `refund` at $b = 1$ for a corrupt prover. An honest prover is never slashed (Lemma 8, Proposition 1); the witness-less `slash-commit` of the `init` branch is likewise available only against a corrupt prover.

Remark 5. $\mathcal{F}_{\text{DV-Groth16}}$'s state contains no group elements, scalars, seeds, garbled circuits, or proofs. All cryptographic objects are produced by $\mathcal{S}$ in the ideal-world execution, using only $\mathcal{F}_{\text{DV-Groth16}}$'s plaintext outputs (x, b, out) as input. This is the standard UC design

pattern and makes the functionality modular: changes to the cryptographic instantiation (e.g., switching from Lamport to Winternitz hash-locks, or changing the Groth16 curve) do not change $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$ at all. They change only the simulator's construction of the public transcript.

4.2 The hybrid model

The protocol $\Pi_{\mathsf{DV\text{-}Groth16}}$ operates in a GUC-style hybrid model with two global functionalities (representing infrastructure shared across all protocol sessions and unrelated protocols) and four session-local functionalities (representing per-session setup and primitives).

Global functionalities. Two functionalities are global, written with an over-bar:

- $\bar{\mathcal{G}}_{\mathsf{Ledger}}$: the global Bitcoin ledger functionality, following the composable treatment of [BMTZ24]. It models UTXOs, script-verified spends, confirmation depth, and time-locks, and exposes interfaces for `Post-UTXO`, `Spend-UTXO-by-script`, and `Read-Confirmed-State` operations.
- $\bar{\mathcal{G}}_{\mathsf{oGG}}$: the observable global generic (bilinear) group of [BFKT24]. It models a shared pairing-friendly group where group operations are performed opaquely, while offering an observability interface (by domain separation) that simulators can use to extract information from queries. It supports oblivious sampling of group elements with unknown discrete logarithm, matching real-world practice via hash-to-curve [BLS01].

Session-local functionalities. Four functionalities are session-local:

- $\mathcal{F}_{\mathsf{GC}}$: a malicious-secure garbled-circuit functionality with cut-and-choose. Models the s-fold garbled-circuit-with-CDS abstraction; security instantiated by the Lindell–Pinkas cut-and-choose framework [LP15] with the balanced bound of [Lin16].
- $\mathcal{F}_{\mathsf{Beacon}}$: an unbiasable randomness functionality, instantiated by any standard two-party commit-reveal coin-toss with default-loses abort penalty (Section 5.7).
- $\mathcal{F}_{\mathsf{HL}}^{+}$: a hash-lock / one-time-signature functionality with two additional interfaces, equivocation slashing (two valid preimages of one wire are script-checkable evidence) and canonical-bit adjudication for the class-(B) wires against the public seed (Section 5.4); instantiated by any malicious-secure one-time hash-based signature scheme, with realisation advantage $\mathsf{Adv}^{\mathsf{HL}+}$.
- $\mathcal{F}_{\mathsf{derive}}$: a derivation functionality for the single class-(C) input T_{ic}, a deterministic public computation from $(\mathsf{seed}, \mathsf{stmt})$, signed by the prover and contestable through a BitVMX/ESSPI-style dispute path invoked only on disagreement; realisation advantage $\mathsf{Adv}^{\mathsf{derive}}$.

Groth16 in this model. Crucially, we do *not* use a separate simulation-extractable NIZK functionality. In the $\bar{\mathcal{G}}_{\mathsf{oGG}}$-hybrid model, Groth16 as originally published [Gro16] UC-realises a weak NIZK functionality $\mathcal{F}_{\mathsf{wNIZK}}$, by [BFKT24]. This means:

- Groth16 is used *unchanged*, no random-oracle wrapper, no signature-of-knowledge transform, no proof-size overhead.
- The simulator's ability to extract witnesses from verifying Groth16 proofs, and to simulate accepting proofs without witnesses, is inherited directly from the global-group functionality via its observability interface.
- The distinguishing advantage has a concrete bound in the query complexity of the environment and the size of the circuit; see [[BFKT24], Theorem 1].

The protocol $\Pi_{\mathsf{DV\text{-}Groth16}}$ (specified in Section 6) is thus a protocol in the

$$\left(\mathcal{G}_{\mathsf{Ledger}}, \mathcal{G}_{\mathsf{oGG}}\,;\, \mathcal{F}_{\mathsf{GC}}, \mathcal{F}_{\mathsf{Beacon}}, \mathcal{F}_{\mathsf{HL}}^{+}, \mathcal{F}_{\mathsf{derive}}\right)\text{-GUC-hybrid model.}$$

Remark 6. By adopting $\mathcal{G}_{\mathsf{oGG}}$ rather than a local NIZK functionality, $\Pi_{\mathsf{DV\text{-}Groth16}}$ inherits the global-group composition guarantees of [BFKT24]: the protocol may run concurrently with any other protocol sharing the same group (e.g., Bitcoin ECDSA signatures on secp256k1, other SNARK-based protocols on BN254 or BLS12-381) without compromising security. [BFKT24]'s Section 6 treats the restricted regime of protocols that cannot tolerate group-operation observation; $\Pi_{\mathsf{DV\text{-}Groth16}}$ falls outside this restricted regime because no secret off-chain group operation depends on unobservability for its security. We are therefore in the standard, observability-tolerant composition regime.

4.3 Main theorem

Theorem 1. *Let $\Pi_{\mathsf{DV\text{-}Groth16}}$ be the DV-Groth16 protocol as specified in Algorithms 6, 7, and 8, run in the $(\mathcal{G}_{\mathsf{Ledger}}, \mathcal{G}_{\mathsf{oGG}}\,;\, \mathcal{F}_{\mathsf{GC}}, \mathcal{F}_{\mathsf{Beacon}}, \mathcal{F}_{\mathsf{HL}}^{+}, \mathcal{F}_{\mathsf{derive}})$-GUC-hybrid model. Then the protocol $\Pi_{\mathsf{DV\text{-}Groth16}}$ GUC-emulates $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$ with respect to static corruption. That is, for every PPT adversary $\mathcal{A}$ in the real world, there exists a PPT simulator $\mathcal{S}$ in the ideal world such that for every PPT environment $\mathcal{Z}$, $\mathsf{EXEC}_{\Pi_{\mathsf{DV\text{-}Groth16}},\mathcal{A},\mathcal{Z}} \approx_c \mathsf{EXEC}_{\mathcal{F}_{\mathsf{DV\text{-}Groth16}},\mathcal{S},\mathcal{Z}}$. The distinguishing advantage of $\mathcal{Z}$ is bounded by*

$$\begin{aligned}\mathsf{Adv}_{\mathsf{in}}^{\mathsf{Groth16}/\mathcal{G}_{\mathsf{oGG}}}(\lambda, q, |C_{\mathsf{assertions}}|) + \mathsf{Adv}_{\mathsf{out}}^{\mathsf{sound},\mathcal{G}_{\mathsf{oGG}}}(\lambda, q, |\mathcal{C}_{\mathsf{verif}}|) + \mathsf{Adv}_{\pi_{\mathsf{SRS}}}^{\mathsf{ZK}}(\lambda) + \mathsf{Adv}_{\mathsf{BHR}}^{\mathsf{garb}}(\lambda) \\ + \mathsf{Adv}_{H}^{\mathsf{CR}} + \binom{s}{s/2}^{-1} + \varepsilon_{\mathcal{R}} + \mathsf{Adv}^{\mathsf{HL}+} + \mathsf{Adv}^{\mathsf{derive}},\end{aligned} \tag{3}$$

where q is $\mathcal{Z}$'s query complexity against $\mathcal{G}_{\mathsf{oGG}}$, s is the cut-and-choose parameter, and $\varepsilon_{\mathcal{R}}$ bounds the bias of the public-coin beacon.

The proof occupies Sections 7, and 8. Section 7 constructs the simulator $\mathcal{S}$ for each of the three corruption patterns ($\mathcal{C} \in \{\emptyset, \{\mathtt{P}\}, \{\mathtt{V}\}\}$, with the $\mathcal{C} = \{\mathtt{P}, \mathtt{V}\}$ case being trivial). Section 8 establishes indistinguishability via a hybrid argument, with one hybrid sequence per corruption pattern; the bound (3) is assembled by telescoping the per-transition advantages.

Remark 7. A notable feature of Theorem 1 is that Groth16 [Gro16] is used unchanged. Prior UC analyses of Groth16 [BKSV21; ARS20] required either a random-oracle-based simulation-extractable transform or the AGM/UC-AGM framework [ABK+21], which has composability limitations flagged in [BFKT24]. By targeting the $\mathcal{G}_{\mathsf{oGG}}$-hybrid model of [BFKT24], $\Pi_{\mathsf{DV\text{-}Groth16}}$ obtains UC security for Groth16 with no wrapper and no per-proof overhead. This matters both for efficiency (proof size is unchanged from standard Groth16) and for clarity (the simulator's extraction of witnesses is a direct invocation of the framework's extraction lemma, rather than a random-oracle trapdoor argument).

Remark 8. The bound (3) is the telescoped sum of the per-transition advantages of the corrupt-prover hybrid sequence (Theorem 3), which dominates the other two corruption patterns. Each transition replaces one real component by its ideal counterpart, and each summand is the distinguishing advantage of that single replacement. The two non-unit coefficients and the combinatorial factor are the only parts that warrant comment.

Soundness versus extraction. The protocol contains two Groth16 instances doing different jobs. The inner instance carries the assertion witness: the bound charges $\mathsf{Adv}_{\mathsf{in}}^{\mathsf{Groth16}/\mathcal{G}_{\mathsf{oGG}}}$ for its extraction via $\mathcal{F}_{\mathsf{wNIZK}}^{\mathcal{C}_{\mathsf{assertions}}}$. The outer instance certifies a deterministic public predicate (Definition 5): its witness is the verification trace, no outer extraction exists, and the bound charges only the plain-soundness term $\mathsf{Adv}_{\mathsf{out}}^{\mathsf{sound},\mathcal{G}_{\mathsf{oGG}}}$ through Lemma 3's pre-reveal

reduction. These are different SRSs, different circuits ($\langle c_{\mathsf{verif}} \rangle$ vs $\sim 10^6$ constraints), and different security properties; neither subsumes the other (Remark 32).

The factor $\binom{s}{s/2}^{-1}$. This is the cut-and-choose soundness error, and it is this small, rather than the constant-per-circuit error of generic cut-and-choose, precisely because of the cross-circuit input binding. The committed proof points $(A_{\mathsf{in}}, B_{\mathsf{in}}, C_{\mathsf{in}})$ (bound via c_1) and the seed-derived canonical labels force every circuit to evaluate on the *same* inputs, so a single correctly-garbled surviving circuit already fixes the correct accept/reject verdict. To suppress an honest rejection a corrupt prover must therefore malform *every* surviving circuit while keeping *every* opened circuit well-formed, i.e. partition the s circuits into a good half and a bad half coinciding exactly with the verifier's challenge set J. Since J is drawn (up to bias $\varepsilon_{\mathcal{R}}$) uniformly from the $\binom{s}{s/2}$ subsets only *after* the circuits are committed, the prover succeeds only by guessing J in advance, with probability $\binom{s}{s/2}^{-1}$. As $\binom{s}{s/2} = \Theta(2^s/\sqrt{s})$, taking $s = \lambda + O(\log \lambda)$ yields $\binom{s}{s/2}^{-1} \leq 2^{-\lambda}$.

The unit-coefficient terms. $\mathsf{Adv}^{\mathsf{garb}}_{\mathsf{BHR}}$, $\mathsf{Adv}^{\mathsf{CR}}_{H}$, $\varepsilon_{\mathcal{R}}$, $\mathsf{Adv}^{\mathsf{HL+}}$ and $\mathsf{Adv}^{\mathsf{derive}}$ are each a single computational assumption invoked at one transition. Although H is evaluated at polynomially many sites (the master commitment c_1, the Merkle root $\mathsf{root}_{\mathsf{GC}}$, the slashing-key pin $h = H(k)$, and every hash-lock), a prover breaks the protocol only by exhibiting *some* collision; the union bound over these sites multiplies $\mathsf{Adv}^{\mathsf{CR}}_{H}$ by a polynomial factor that is absorbed into the (negligible) term. The beacon bias $\varepsilon_{\mathcal{R}}$ is listed separately from $\binom{s}{s/2}^{-1}$ because the latter is the soundness under a perfectly uniform challenge while $\varepsilon_{\mathcal{R}}$ measures the deviation of the realised beacon from uniform; for an ideal beacon $\varepsilon_{\mathcal{R}} = 0$.

Writing the bound in additive form, rather than collapsing terms into $\mathsf{Adv}^{\mathsf{Groth16}/\mathcal{G}_{\mathsf{oGG}}}$, keeps the dependence on each primitive visible and makes any future tightening of an individual transition immediately legible.

Remark 9. Theorem 1 is per-session. By the GUC composition theorem [CDPW07], an environment orchestrating N concurrent sessions of $\Pi_{\mathsf{DV\text{-}Groth16}}$ achieves distinguishing advantage at most N times the single-session bound (3), with the global functionalities $\mathcal{G}_{\mathsf{Ledger}}$ and $\mathcal{G}_{\mathsf{oGG}}$ shared across sessions. The scaling is therefore linear in the session count, with the per-session terms unchanged.

Remark 10. Under Assumption 2, the reduction in the proof of Lemma 3 embeds an externally-sampled outer Groth16 SRS into the corrupt prover's Phase-I view; consistency of the simulated π_{SRS} with this embedded SRS costs $\mathsf{Adv}^{\mathsf{ZK}}_{\pi_{\mathsf{SRS}}}$, the zero-knowledge advantage of the SRS-correctness proof system. Under any Fiat–Shamir instantiation of π_{SRS} in the random-oracle model, this term is $\mathsf{negl}(\lambda)$.

5 Building blocks

This section develops the cryptographic infrastructure on which the protocol $\Pi_{\mathsf{DV\text{-}Groth16}}$ is built. We treat each building block twice: a classical view giving the standard cryptographic construction and primitive, and a UC view casting the same primitive as an ideal functionality used in the hybrid model of Section 4.2. The subsections are ordered by dependency: inner Groth16 (public, reusable SRS), outer Groth16 with the per-proof trapdoor SRS, garbled circuits, hash-locked canonical labels, two structural lemmas binding the commitment layers, the inner/outer bridge (the scalarization of the outer verification equation), the beacon and ledger functionalities, and a summary table of instantiation choices.

Definition 1. The assertion-circuit Groth16 SRS $\mathsf{SRS}^{\mathsf{in}}$ (with verification key $\mathsf{vk}_{\mathsf{in}}$) is a

public, one-time structured reference string for $C_{\mathsf{assertions}}$, produced once by any trusted ceremony and reused across all sessions. Its trapdoor $(\tau_{\mathsf{in}}, \alpha_{\mathsf{in}}, \beta_{\mathsf{in}}, \gamma_{\mathsf{in}}, \delta_{\mathsf{in}})$ is destroyed by the ceremony and is *never revealed* to either party; in particular it is not derived from, and is independent of, the per-session seed. The verification-key term for public input x is $\mathsf{vk}_x = (p_0 + x\,p)\,G_1$, where p_0, p are the (public) closed-form QAP evaluations of Remark 21. `P` holds $\mathsf{SRS}^{\mathsf{in}}$ in group form, exactly as a standard Groth16 prover. The inner curve sits at the bottom of a one-layer 2-chain: the outer curve's scalar field equals the inner base field $\mathbb{F}_q$ (reference instantiation: inner BLS12-377, outer BW6-761). This is what makes the outer circuit's arithmetic and the outer statement's encoding native (Definition 3); inner curves outside a chain are reachable only via non-native emulation, with the costs discussed in Section 9.2.

Remark 11. Only the small per-proof *outer* SRS (Definition 2) is seed-derived and revealed; the heavy inner SRS is public and reusable, with its trapdoor permanently destroyed. This asymmetry, a public heavy inner setup against a compact per-proof outer SRS, is the central optimisation: the per-session phase burns only the $\approx 2\langle c_{\mathsf{verif}}\rangle$-point outer SRS (working estimate $\sim 10^5$ points, 5–10 MB) per proof, independent of $|C_{\mathsf{assertions}}|$, while the heavy inner SRS is amortised across all sessions.

5.1 Inner Groth16

The inner Groth16 scheme $(\mathsf{Setup}_{\mathsf{in}}, \mathsf{Prove}_{\mathsf{in}}, \mathsf{Verify}_{\mathsf{in}})$ is instantiated for the assertion circuit $C_{\mathsf{assertions}}$ under the *public, reusable* SRS $\mathsf{SRS}^{\mathsf{in}}$ of Definition 1, produced once by a trusted ceremony with its trapdoor destroyed. `P` produces a standard Groth16 proof π_{in}; witness-hiding is the ordinary zero-knowledge of Groth16. The designation of the protocol does not come from the inner SRS (which is public): it comes from the per-proof outer trapdoor (Definition 2), which only `V` holds and which is required to evaluate the garbled verification circuit (Remark 19). Soundness against a cheating prover reduces to ordinary inner-Groth16 knowledge-soundness combined with the pre-reveal outer soundness (Definition 5, Lemma 3).

Classical view. Groth16 as in [Gro16], with proofs $\pi_{\mathsf{in}} = (A_{\mathsf{in}}, B_{\mathsf{in}}, C_{\mathsf{in}}) \in \mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{G}_1$ and pairing-based verification.

UC view. In the $\mathcal{G}_{\mathsf{oGG}}$-hybrid model, inner Groth16 UC-realises the weak NIZK functionality $\mathcal{F}_{\mathsf{wNIZK}}^{C_{\mathsf{assertions}}}$, by [[BFKT24], Thm. 1]. Concretely, $\mathcal{F}_{\mathsf{wNIZK}}^{C_{\mathsf{assertions}}}$ offers four interfaces: a `prove` interface that on input $(\texttt{prove}, \mathsf{sid}, x, w)$ with $C_{\mathsf{assertions}}(x; w) = 1$ returns a proof handle π_{in}, leaking (x, π_{in}) but not w to $\mathcal{S}$; a `verify` interface returning the correctness bit on $(\mathsf{sid}, x, \pi_{\mathsf{in}})$; a `simulate` interface (available only to $\mathcal{S}$) producing a verifying proof on a true statement without a witness; and an `extract` interface (also $\mathcal{S}$-only) that on a verifying π_{in} not previously produced by `simulate` returns a witness w with $C_{\mathsf{assertions}}(x; w) = 1$. The distinguishing advantage of any environment against this emulation is

$$\mathsf{Adv}^{\mathsf{Groth16}/\mathcal{G}_{\mathsf{oGG}}}(\lambda, q, |C_{\mathsf{assertions}}|) = O\left(\frac{q^2 \cdot |C_{\mathsf{assertions}}|}{p}\right),$$

where q is the environment's query complexity against $\mathcal{G}_{\mathsf{oGG}}$ and p is the group order [[BFKT24], Thm. 1].

Assumption 1. *In the $\mathcal{G}_{\mathsf{oGG}}$-hybrid model of [BFKT24], Groth16 with SRS $\mathsf{SRS}^{\mathsf{in}}$ and verification key $\mathsf{vk}_{\mathsf{in}}$ UC-realises $\mathcal{F}_{\mathsf{wNIZK}}^{C_{\mathsf{assertions}}}$ with the advantage bound stated above.*

Remark 12. Witness-hiding of π_{in} is the standard zero-knowledge of Groth16: the `simulate` interface of $\mathcal{F}_{\mathsf{wNIZK}}^{C_{\mathsf{assertions}}}$ produces a verifying proof on a true statement without a witness, using the ceremony trapdoor in the security analysis only (no protocol party holds it).

This is invoked in the both-honest and corrupt-verifier simulators of Section 7; it is a property of Groth16, surfaced as the `simulate` interface of $\mathcal{F}_{\mathsf{wNIZK}}$.

Remark 13. Assumption 1 is invoked in the simulator constructions of Section 7: the `simulate` interface supports the cases of both-honest and corrupt verifier (producing simulated π_{in}^{S} without a witness), and the `extract` interface supports the corrupt-prover case (recovering w^{extr} from $\mathcal{A}$'s $\pi_{\mathsf{in}}^{\mathcal{A}}$).

5.2 Outer Groth16 and trapdoor SRS

The outer Groth16 scheme is instantiated *per proof* by the verifier `V`. It is the standard Groth16 construction applied to the fixed, small outer circuit $\mathcal{C}_{\mathsf{verif}}$ defined in Section 6.1, whose statement is $(x, A_{\mathsf{in}}, B_{\mathsf{in}}, C_{\mathsf{in}})$ and whose acceptance is, by Definition 5, the statement that the inner proof π_{in} verifies. Its SRS, denoted $\mathsf{SRS}^{\mathsf{out}}$, is the Groth16 SRS for $\mathcal{C}_{\mathsf{verif}}$, parameterised by a random seed from which `V` derives all trapdoor scalars.

Classical view. $\mathsf{SRS}^{\mathsf{out}}_{\mathsf{group}}$ consists of the standard Groth16 SRS group elements for $\mathcal{C}_{\mathsf{verif}}$, listed explicitly in Definition 2 below. Proofs $\pi_{\mathsf{out}} = (A_{\mathsf{out}}, B_{\mathsf{out}}, C_{\mathsf{out}}) \in \mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{G}_1$ are produced by the standard Groth16 prover; π_{out} *certifies* π_{in} (it attests that $\mathcal{C}_{\mathsf{verif}}$ accepts on $(x, A_{\mathsf{in}}, B_{\mathsf{in}}, C_{\mathsf{in}})$, that is, that π_{in} verifies; Definition 5).

UC view. In the same $\mathcal{G}_{\mathsf{oGG}}$-hybrid model, outer Groth16 UC-realises the weak NIZK functionality $\mathcal{F}_{\mathsf{wNIZK}}^{\mathcal{C}_{\mathsf{verif}}}$ for the outer circuit $\mathcal{C}_{\mathsf{verif}}$ (Definition 3). The interface is identical to the inner case; the only difference is the circuit on which the functionality is parameterised, which is *fixed* ($\langle c_{\mathsf{verif}} \rangle$ constraints) and independent of the assertion-circuit size; only the `verify` interface is used at this layer, R_{outer} is a deterministic predicate (Definition 5) and no outer extraction exists or is invoked. The advantage bound is $\mathsf{Adv}^{\mathsf{Groth16}/\mathcal{G}_{\mathsf{oGG}}}(\lambda, q, |\mathcal{C}_{\mathsf{verif}}|)$, where $|\mathcal{C}_{\mathsf{verif}}|$ is the (constant) size of the outer circuit.

Remark 14. `V`'s seed-derived trapdoor scalars $(\tau_{\mathsf{out}}, \alpha_{\mathsf{out}}, \beta_{\mathsf{out}}, \gamma_{\mathsf{out}}, \delta_{\mathsf{out}})$ are, from $\mathcal{G}_{\mathsf{oGG}}$'s perspective, the discrete logarithms of the SRS group elements `V` publishes. $\mathcal{G}_{\mathsf{oGG}}$'s observability interface exposes these to $\mathcal{S}$ (via the symbolic representations of `V`'s queries), which is what enables the $\mathcal{F}_{\mathsf{wNIZK}}$ extractor to work on the outer circuit. In the hybrid design the simulators need the outer trapdoors only *after* the seed is public (to compute the package scalars as an honest prover would, Algorithm 11), so observability-based trapdoor recovery is no longer load-bearing for the corrupt-verifier case; it remains available as a convenience and is what the $\mathcal{F}_{\mathsf{wNIZK}}$ framework supplies in general.

Definition 2. Given $\mathsf{seed} \in \{0,1\}^{\lambda}$, derive $(\tau_{\mathsf{out}}, \alpha_{\mathsf{out}}, \beta_{\mathsf{out}}, \gamma_{\mathsf{out}}, \delta_{\mathsf{out}}) \leftarrow H(\mathsf{seed} \,\|\, \cdot)$ with labels $\{\texttt{tau}, \texttt{alpha}, \texttt{beta}, \texttt{gamma}, \texttt{delta}\}$. The outer SRS in group form consists of the standard Groth16 SRS elements for the outer circuit $\mathcal{C}_{\mathsf{verif}}$:

$$\begin{aligned}\mathsf{SRS}^{\mathsf{out}}_{\mathsf{group}} := \big\{ &\alpha_{\mathsf{out}} G_1,\ \beta_{\mathsf{out}} G_1,\ \beta_{\mathsf{out}} G_2,\ \gamma_{\mathsf{out}} G_2,\ \delta_{\mathsf{out}} G_1,\ \delta_{\mathsf{out}} G_2,\ \{\tau_{\mathsf{out}}^{j} G_1, \tau_{\mathsf{out}}^{j} G_2\}_{j=1}^{d_{\mathsf{out}}-1},\\ &\{\tau_{\mathsf{out}}^{j} t_{\mathsf{out}}(\tau_{\mathsf{out}}) \delta_{\mathsf{out}}^{-1} G_1\}_{j=0}^{d_{\mathsf{out}}-2},\\ &\{\delta_{\mathsf{out}}^{-1}(\beta_{\mathsf{out}} u_i^{\mathsf{out}} + \alpha_{\mathsf{out}} v_i^{\mathsf{out}} + w_i^{\mathsf{out}})(\tau_{\mathsf{out}})\, G_1\}_{i \in \mathsf{priv}},\ \{K_i^{\mathsf{out}}\}_i \big\}\end{aligned}$$

where $K_i^{\mathsf{out}} = \gamma_{\mathsf{out}}^{-1}(\alpha_{\mathsf{out}} v_i^{\mathsf{out}}(\tau_{\mathsf{out}}) + \beta_{\mathsf{out}} u_i^{\mathsf{out}}(\tau_{\mathsf{out}}) + w_i^{\mathsf{out}}(\tau_{\mathsf{out}}))G_1$ for the QAP polynomials of $\mathcal{C}_{\mathsf{verif}}$, with d_{out} the corresponding QAP degree bound and $t_{\mathsf{out}}(X)$ the corresponding target polynomial. Because $\mathcal{C}_{\mathsf{verif}}$ is fixed and independent of $|C_{\mathsf{assertions}}|$, $\mathsf{SRS}^{\mathsf{out}}_{\mathsf{group}}$ contains $\approx 2\langle c_{\mathsf{verif}} \rangle$ points (working estimate $\sim 10^5$, that is, 5–10 MB; to be pinned with the $\mathcal{C}_{\mathsf{verif}}$ benchmark). The outer verification key $\mathsf{vk}_{\mathsf{outer}}(\mathsf{seed})$ is the standard Groth16 vk derived from $\mathsf{SRS}^{\mathsf{out}}_{\mathsf{group}}$.

Remark 15. The QAP of $\mathcal{C}_{\mathsf{verif}}$ has constant degree d_{out} ($\langle c_{\mathsf{verif}} \rangle$ R1CS constraints, Definition 3), independent of $|C_{\mathsf{assertions}}|$. Consequently $\mathsf{SRS}^{\mathsf{out}}$ is of fixed size regardless of the assertion

circuit: $\approx 2\langle c_{\mathsf{verif}}\rangle \approx 10^5$ points against $\approx 2|C_{\mathsf{assertions}}| \approx 2\text{–}3 \times 10^6$ for a per-session SRS over a 10^6-constraint assertion circuit, with the advantage growing linearly in $|C_{\mathsf{assertions}}|$ and a crossover near $|C_{\mathsf{assertions}}| \approx 5 \times 10^4$. This is the per-proof saving claimed in Remark 11; the missing $\delta_{\mathsf{out}}^{-1}$-scaled prover terms, without which $\mathsf{Prove}_{\mathsf{out}}$ cannot compute C_{out}, are now included above.

Let $\vec{\sigma} = \big(H(\mathsf{seed} \,\|\, \mathtt{tau}), \ldots, H(\mathsf{seed} \,\|\, \mathtt{delta})$, then π_{SRS} is a non-interactive proof of knowledge for the relation $R_{\mathsf{SRS}} := \big\{(hs, \mathsf{com}_{\vec{\sigma}}\,;\, \mathsf{seed}) \,:\, H(\mathsf{seed}) = hs \wedge \vec{\sigma}\big\}$, that is: it binds the seed commitment to the five trapdoor scalars, exposed through commitments $\mathsf{com}_{\vec{\sigma}}$ included in the statement. SRS *well-formedness* is deliberately not part of R_{SRS}, and splits into two properties checked at two times. *Internal consistency*, that the published points realise *some* scalar vector, is publicly checkable by $\mathtt{P}$ on receipt via standard batched pairing checks ($e(\tau_{\mathsf{out}}^{j} G_1, G_2) = e(\tau_{\mathsf{out}}^{j-1} G_1, \tau_{\mathsf{out}} G_2)$ and analogues under a random linear combination), at $O(\langle c_{\mathsf{verif}}\rangle)$ group operations and $O(1)$ pairings; these checks are independent of seed. *Seed expansion*, that this scalar vector equals $\big(H(\mathsf{seed} \,\|\, \mathtt{tau}), \ldots, H(\mathsf{seed} \,\|\, \mathtt{delta})\big)$, is checkable only once seed is public, and is enforced by the seed-expansion check of Algorithm 7 (Lemma 4). The commitments $\mathsf{com}_{\vec{\sigma}}$ are binding commitments (e.g. hash-based, $\mathsf{com}_i = H(\sigma_i \,\|\, \rho_i)$); their role is confined to making R_{SRS} a well-defined pre-reveal statement, and no security property of the protocol rests on relating them to the published points. (The inner SRS is public and ceremony-generated, Definition 1, so it is not part of R_{SRS}.) π_{SRS} is instantiated as a universal-setup SNARK over a one-time universal SRS; the circuit covers the hash derivations and commitment openings only, and π_{SRS} is of constant size. In the UC view, π_{SRS} realises a small $\mathcal{F}_{\mathsf{wNIZK}}^{R_{\mathsf{SRS}}}$ in the same hybrid model.

Assumption 2. *The NIZK proof system used to construct π_{SRS} for the relation R_{SRS} is zero-knowledge in the $(\mathcal{G}_{\mathsf{oGG}}, RO)$-hybrid model: there exists a PPT simulator $\mathsf{Sim}_{\pi_{\mathsf{SRS}}}$ such that, on input a statement $(\mathsf{SRS}_{\mathsf{group}}^{\mathsf{out}}, hs)$ for which there exists a witness seed satisfying R_{SRS}, $\mathsf{Sim}_{\pi_{\mathsf{SRS}}}$ produces a proof ${\pi_{\mathsf{SRS}}}^*$ such that no PPT distinguisher can tell apart $(\mathsf{SRS}_{\mathsf{group}}^{\mathsf{out}}, hs, {\pi_{\mathsf{SRS}}}^*)$ from $(\mathsf{SRS}_{\mathsf{group}}^{\mathsf{out}}, hs, \pi_{\mathsf{SRS}})$, where the latter is produced honestly by $\mathtt{V}$ using seed, with advantage at most $\mathsf{Adv}_{\pi_{\mathsf{SRS}}}^{\mathsf{ZK}}(\lambda)$. The standard Fiat–Shamir compilation of a sigma protocol for R_{SRS} satisfies this assumption with $\mathsf{Adv}_{\pi_{\mathsf{SRS}}}^{\mathsf{ZK}}(\lambda) = \mathsf{negl}(\lambda)$ in the random-oracle model.*

5.3 Garbled circuits with authenticated labels

Classical view. We use a projective garbling scheme $(\mathsf{Gb}, \mathsf{Ev}, \mathsf{De})$ in the sense of Bellare-Hoang-Rogaway [BHR12], satisfying privacy, obliviousness, and authenticity. The scheme takes a Boolean circuit C and outputs $(\widetilde{C}, e, d)$, where e is the input encoding (a mapping from clear inputs to labels) and d is the decoding information (output labels λ_0, λ_1 for reject/accept).

UC view. We model garbled circuits via the ideal functionality $\mathcal{F}_{\mathsf{GC}}^{s,C}$ parameterised by the cut-and-choose parameter s and the circuit C being garbled (Algorithm 2).

Algorithm 2 Ideal functionality $\mathcal{F}_{\mathsf{GC}}^{s,C}$ (cut-and-choose garbled circuits)

Parameters: cut-and-choose parameter $s \in \mathbb{N}$ and Boolean circuit C.
State: $\{(\widetilde{C}_i, e_i, d_i)\}_{i=1}^{s}$, status flags $\{\mathsf{open}_i\}_{i=1}^{s}$, opened-subset record $J \subset [s]$.

I. Garble. On $(\mathtt{garble}, \mathsf{sid}, s, C)$ from the garbler $\mathtt{P}$:
1: **for** $i = 1$ **to** s **do** $(\widetilde{C}_i, e_i, d_i) \leftarrow \mathsf{Gb}(1^\lambda, C)$; store internally
2: **end for**
3: **return** $\{\widetilde{C}_i\}_{i=1}^{s}$ to the caller and to $\mathcal{S}$ ▷ garbled handles are observable
II. Check-open. On $(\mathtt{check\text{-}open}, \mathsf{sid}, J)$ from any party, $J \subset [s]$, $|J| = s/2$:
4: **for** each $i \in J$ **do**
5: reveal (e_i, d_i, r_i), where r_i is the garbling randomness; set $\mathsf{open}_i \leftarrow \mathtt{opened}$
6: **end for**
7: send revealed material to the requester ▷ enables standalone verification against Gb
III. Evaluate. On $(\mathtt{evaluate}, \mathsf{sid}, i, \mathsf{labels})$ from the evaluator $\mathtt{V}$, $i \in [s] \setminus J$:
8: $\mathsf{out_label} \leftarrow \mathsf{Ev}(\widetilde{C}_i, \mathsf{labels})$
9: **return** $\mathsf{out_label}$ ▷ decodes via d_i to a Boolean output bit
IV. Simulate. ($\mathcal{S}$-only.) On $(\mathtt{simulate}, \mathsf{sid}, \{b_i\}_{i \in [s] \setminus J})$ from $\mathcal{S}$:
10: produce simulated handles $\{\widetilde{C}_i^{\mathsf{sim}}\}_{i=1}^{s}$ and encoded labels such that the check (II) on any $J' \subset [s]$, $|J'| = s/2$, verifies w.h.p., and evaluation (III) of each $i \in [s] \setminus J$ outputs b_i regardless of input

Assumption 3. *The garbling scheme of [BHR12] UC-realises $\mathcal{F}_{\mathsf{GC}}^{s,C}$ in two separable senses:*

1. ***Privacy/authenticity .*** *For every PPT environment $\mathcal{Z}$, the distinguishing advantage between the real garbling and the $\mathcal{F}_{\mathsf{GC}}$-emulated handles, when the garbler is honest or when no soundness condition is at stake, is at most $\mathsf{Adv}_{\mathsf{BHR}}^{\mathsf{garb}}(\lambda)$.*

2. ***Cut-and-choose soundness.*** *Against a malicious garbler, the probability that the protocol opens $s/2$ challenge circuits all of which pass inspection while leaving at least one malformed circuit in $[s] \setminus J$ is at most $\binom{s}{s/2}^{-1}$, by the standard cut-and-choose analysis of [Lin16].*

These two losses are charged separately *in the hybrid sequence: the BHR loss at Lemma 12 (and at the analogous step in the both-honest and corrupt-verifier sequences); the cut-and-choose loss at the soundness step where surviving-circuit acceptance is invoked, namely Lemma 14 step (iii). This separation avoids any appearance of double-counting and matches the actual structure of the hybrid argument.*

5.4 Hash-locked canonical labels

This subsection describes the mechanism that makes $\Pi_{\mathsf{DV\text{-}Groth16}}$'s GC input-authority policy concrete. For each class-(B) input wire of $\mathcal{C}_{\mathsf{GC}}$, the raw seed-derived outer trapdoors $\alpha_{\mathsf{out}}, \beta_{\mathsf{out}}, \delta_{\mathsf{out}}$ (Definition 4), the canonical label is determined by the public seed: the wire value is itself an $H(\mathsf{seed} \,\|\, \mathtt{label})$ output, so the canonical bit of each wire-bit is a bit of that output. Class-(A) and class-(C) wires have no canonical bit; they are bound by $\mathtt{P}$'s one-time signatures (Definition 4).

Algorithm 3 Canonical label binding (prover-side construction)

Input: the s garbled circuits $\{(\widetilde{C}_i, e_i, d_i)\}_{i=1}^{s}$ from Algorithm 2.
Output: hash-lock commitments $\{\mathsf{hl}_{i,j,b}\}_{i,j,b}$, included in each c_i^{gc} and $\mathsf{root}_{\mathsf{GC}}$.

I. Hash-lock commitment (Phase I).
for each circuit $i \in [s]$, each Bob-authoritative wire j, each bit $b \in \{0,1\}$ **do**
  let $\lambda_{i,j,b}$ be the input label in e_i for wire j, bit b
  P publishes (as part of $\mathsf{root}_{\mathsf{GC}}$) $\mathsf{hl}_{i,j,b} := H(\mathtt{cl} \,\|\, i \,\|\, j \,\|\, b \,\|\, \lambda_{i,j,b})$
end for

II. Canonical-bit derivation (Phase III, post seed reveal).
for each class-(B) wire-bit j **do** $\mathsf{cb}_j := \mathsf{bit}_{\beta(j)}\big(H(\mathsf{seed}, \mathtt{label}(j))\big)$ ▷ = the $\beta(j)$-th bit of the corresponding trapdoor scalar; Remark 17
end for

III. Label release (Phase III).
for each surviving circuit $i \in [s] \setminus J$, each class-(B) wire-bit j **do**
  P (garbler) reveals the canonical-bit label $\lambda_{i,j,\mathsf{cb}_j}$ in UTXO3
  V (and, on dispute, $\mathcal{F}_{\mathsf{HL}}^{+}$.`adjudicate`) checks $H(\mathtt{cl} \,\|\, i \,\|\, j \,\|\, \mathsf{cb}_j \,\|\, \lambda_{i,j,\mathsf{cb}_j}) = \mathsf{hl}_{i,j,\mathsf{cb}_j}$ and that cb_j is the genuine public bit
end for

Algorithm 4 Ideal functionality $\mathcal{F}_{\mathsf{HL}}^{+}$ (hash-locks with equivocation slashing and canonical-bit adjudication)

State: for each wire (i,j), a label pair $(\lambda_{i,j,0}, \lambda_{i,j,1})$ and a flag in $\{\perp, \mathtt{committed}, \mathtt{opened}\}$.

I. Commit. On $(\mathtt{commit}, \mathsf{sid}, \mathsf{wire}_{i,j}, \lambda_{i,j,0}, \lambda_{i,j,1})$ from P:
if the wire is already committed **then return** ▷ ignore
end if
store the two labels; set status to `committed`
return a hash-lock handle to P and to $\mathcal{S}$

II. Open. On $(\mathtt{open}, \mathsf{sid}, \mathsf{wire}_{i,j}, b, \lambda)$ from any party:
if $\lambda = \lambda_{i,j,b}$ as previously committed **then** set status to `opened`; **return** `true`
else return `false`
end if

III. Observe. ($\mathcal{S}$-only.) On $(\mathtt{observe}, \mathsf{sid})$ from $\mathcal{S}$:
return the complete table of committed labels

IV. Equivocation-slash. On valid openings of *both* slots $(\mathsf{wire}_{i,j}, 0)$ and $(\mathsf{wire}_{i,j}, 1)$ of any wire:
award the bonded penalty against the committing party; record the event on $\mathcal{G}_{\mathsf{Ledger}}$ ▷ script-native: two preimages

V. Adjudicate. (seed public.) On $(\mathtt{adjudicate}, \mathsf{sid}, \mathsf{wire}_{i,j}, b, \lambda)$ for a class-(B) wire:
return `true` iff $\lambda = \lambda_{i,j,b}$ as committed *and* $b = \mathsf{bit}_{\beta(j)}\big(H(\mathsf{seed} \,\|\, \mathtt{label}(j))\big)$ ▷ wrong-bit reveals are positive slash-commit evidence

The instantiation of $\mathcal{F}_{\mathsf{HL}}^{+}$ is by any malicious-secure one-time hash-based signature scheme (Lamport, Winternitz, or BitVMX-style hash-locks) for interfaces I–IV; interface V additionally requires the canonical bit to be checkable from the on-chain seed, whose Bitcoin-script realisation (hash-evaluation in script, taproot-leaf strategies, the absence of `OP_CAT`) is engineering scoped out of this paper exactly as before, but the *interface* is now part of the functionality, and its realisation-specific advantage is $\mathsf{Adv}^{\mathsf{HL}+}(\lambda)$. The protocol additionally uses a derivation functionality $\mathcal{F}_{\mathsf{derive}}$ for the single class-(C) wire: T_{ic} is a deterministic public computation from $(\mathsf{seed}, \mathsf{stmt})$, signed by `P` and contestable by `V` through a BitVMX/ESSPI-style dispute path [LJF25], invoked only on disagreement; its soundness advantage is $\mathsf{Adv}^{\mathsf{derive}}(\lambda)$. Both terms appear in the bound of Theorem 1.

Remark 16. The garbled circuit's input wires are bound by exactly one of three mechanisms, chosen by wire class (Definition 4), we call this *complementary input authority*. Class (A): `P`'s one-time-signed scalars $(a_{\mathsf{out}}, b_{\mathsf{out}}, c_{\mathsf{out}})$, whose *values* are checked in-circuit against the c_1-committed proof points (checks 1–3); equivocation is script-slashable, and wrong values yield the reject label, note that a corrupt `P` holding a *valid* witness can therefore still self-slash by signing garbage, which is why $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$ permits slashing of a corrupt prover at $b = 1$ (Remark 4). Class (B): seed-derived canonical labels released by `P` post-reveal, with wrong-bit reveals adjudicable from the public seed ($\mathcal{F}_{\mathsf{HL}}^{+}$). Class (C): the signed derived value T_{ic}, contestable through $\mathcal{F}_{\mathsf{derive}}$. No wire is bound twice; the union covers the input space, and the case analysis under both corruption patterns needs no per-case construction: the prover-side guarantee is Corollary 1 (invoked inside Lemma 14); the verifier-side guarantee is Lemma 8.

5.5 Structural lemmas

This subsection collects two short lemmas on the binding of the master commitment c_1 and the canonical labels. Together with the cut-and-choose soundness of Assumption 3, they form the binding foundation on which the inner/outer bridge (Section 5.6) and the corrupt-prover hybrid argument (Section 8.1) rest.

Lemma 1. *Let* $(\mathsf{open}_1, \mathsf{open}_2)$ *be two distinct openings of* c_1 *that both parse correctly as Phase-I output tuples. Then an adversary producing both openings can be converted to a collision-finder for* H *with the same advantage.*

Proof. Immediate from the collision-resistance of H: c_1 is the hash of the concatenation of Phase-I outputs with nonce ρ_1, and any two distinct openings mapping to the same c_1 yield a hash collision. □

Lemma 2. *Given the on-chain* seed*, any two distinct values opened as the canonical label of a class-(B) wire* j *in a circuit* $i \in [s] \setminus J$ *against the pre-committed* $\mathsf{hl}_{i,j,\mathsf{cb}_j}$ *yield a collision for* H*; the canonical label is thus unique except with advantage* $\mathsf{Adv}_{H}^{\mathsf{CR}}$*.*

Remark 17. Under the derivation of Definition 4, the trapdoor scalars *are* $H(\mathsf{seed} \,\|\, \mathtt{label})$ outputs, so the canonical bit of wire-bit j, $\mathsf{bit}_{\beta(j)}(H(\mathsf{seed} \,\|\, \mathtt{label}(j)))$, is literally the $\beta(j)$-th bit of the corresponding trapdoor scalar. Concatenating the canonical bits of a class-(B) scalar recovers the scalar itself, so the values entering $\mathcal{C}_{\mathsf{GC}}$'s class-(B) wires in every surviving circuit are the genuine seed-derived $\alpha_{\mathsf{out}}, \beta_{\mathsf{out}}, \delta_{\mathsf{out}}$. `P` (the garbler) holds the label preimages and reveals exactly the canonical-bit labels post-reveal; `V` and, on dispute, the $\mathcal{F}_{\mathsf{HL}}^{+}$ adjudication interface verify the revealed bit against the public seed. Garbling is unaffected: `P` commits hash-locks for both bits in Phase I, and a wrong-bit reveal is positive slash-commit evidence (Lemma 5).

Proof. The label is determined by the committed $\mathsf{hl}_{i,j,\mathsf{cb}_j}$ (included in c_i^{gc}, hence in $\mathsf{root}_{\mathsf{GC}}$); two distinct preimages of one hash-lock are a collision. The bit cb_j itself is a deterministic public function of the on-chain seed. □

5.6 The inner/outer bridge

This subsection carries the construction's central mechanism. The bridge from outer-relation truth to inner validity is definitional (Definition 5); what requires proof is that the *garbled* layer faithfully decides whether the committed outer proof verifies. Theorem 2 states the exact scalarization of the outer verification equation; Corollary 1 converts it into the enforcement guarantee used by the hybrid argument (Lemma 14).

Theorem 2 (Scalarization of the outer verification equation)**.** *Let* $(A_{\mathsf{out}}, B_{\mathsf{out}}, C_{\mathsf{out}})$ *have discrete logarithms* $(a^*_{\mathsf{out}}, b^*_{\mathsf{out}}, c^*_{\mathsf{out}})$ *with respect to the fixed generators of the outer curve, let* $(\tau_{\mathsf{out}}, \alpha_{\mathsf{out}}, \beta_{\mathsf{out}}, \gamma_{\mathsf{out}}, \delta_{\mathsf{out}})$ *be the outer trapdoors, and let* T_{ic} *be as in Definition 4. Then the outer Groth16 verification equation holds for* $(A_{\mathsf{out}}, B_{\mathsf{out}}, C_{\mathsf{out}})$ *on* stmt *iff*

$$a^*_{\mathsf{out}}\, b^*_{\mathsf{out}} \;=\; \alpha_{\mathsf{out}}\beta_{\mathsf{out}} + T_{\mathsf{ic}} + c^*_{\mathsf{out}}\, \delta_{\mathsf{out}} \qquad \text{in } \mathbb{F}_q.$$

Proof. Both sides of the pairing equation are powers of $g_T = e(G_1, G_2)$; in a prime-order group exponentiation is injective, so the equation holds iff the exponents agree: $a^*_{\mathsf{out}} b^*_{\mathsf{out}} = \alpha_{\mathsf{out}}\beta_{\mathsf{out}} + \gamma_{\mathsf{out}} \sum_k \mathsf{stmt}_k\, \mathsf{dlog}(K^{\mathsf{out}}_k) + c^*_{\mathsf{out}}\delta_{\mathsf{out}}$. Since $\gamma_{\mathsf{out}} \cdot \mathsf{dlog}(K^{\mathsf{out}}_k) = (\beta_{\mathsf{out}} u^{\mathsf{out}}_k + \alpha_{\mathsf{out}} v^{\mathsf{out}}_k + w^{\mathsf{out}}_k)(\tau_{\mathsf{out}})$ by Definition 2, the middle term sums to T_{ic}. □

Corollary 1 (Enforcement)**.** *Fix the* c_1*-committed* $(A_{\mathsf{out}}, B_{\mathsf{out}}, C_{\mathsf{out}})$ *and suppose the genuine class-(B) values and* T_{ic} *are used (Definition 4; enforcement of this hypothesis is by* $\mathcal{F}^+_{\mathsf{HL}}$ *and* $\mathcal{F}_{\mathsf{derive}}$*). The unique scalars passing checks 1–3 of* $\mathcal{C}_{\mathsf{GC}}$ *are the genuine discrete logarithms of the committed points, and by Theorem 2 they pass check 4 iff the committed triple verifies under* $\mathsf{vk}_{\mathsf{outer}}(\mathsf{seed})$*. Hence, for every honestly-garbled surviving circuit: if the committed* π_{out} *verifies, the genuine inputs yield the accept label (completeness); if it does not, every input assignment* $\mathtt{P}$ *can authorise fails some check and yields the reject label, releasing* k*. Surviving-circuit acceptance is therefore equivalent, up to garbling soundness,* $\binom{s}{s/2}^{-1}$*,* $\mathsf{Adv}^{\mathsf{HL}+}$ *and* $\mathsf{Adv}^{\mathsf{derive}}$*, to verification of the committed outer proof, and thereby (Definition 5, Lemma 3) to validity of the inner proof.*

Proof. Uniqueness in checks 1–3 is uniqueness of discrete logarithms. Under the corollary's hypothesis the class-(B) values and T_{ic} entering check 4 are the genuine ones, so on a surviving circuit $\mathtt{P}$'s only authority is the class-(A) signing of $(a_{\mathsf{out}}, b_{\mathsf{out}}, c_{\mathsf{out}})$; the two cases below are therefore exhaustive over what $\mathtt{P}$ can authorise. Completeness and the reject direction are Theorem 2 applied to the genuine scalars, plus the observation that any signed $(a_{\mathsf{out}}, b_{\mathsf{out}}, c_{\mathsf{out}})$ other than the genuine ones fails one of checks 1–3. The error terms quantify the hypotheses: a malformed survivor (cut-and-choose), a forged output label (garbling authenticity), a subverted class-(B) adjudication ($\mathcal{F}^+_{\mathsf{HL}}$), or a subverted $\mathcal{F}_{\mathsf{derive}}$ dispute. □

Remark 18. Theorem 2 and Corollary 1 establish that surviving-circuit acceptance certifies that the *committed* π_{out} verifies, hence (Definition 5, Lemma 3) that π_{in} is a *verifying* Groth16 proof for statement x. They do not produce a witness w for $C_{\mathsf{assertions}}(x; \cdot)$: extraction requires inner knowledge-soundness, the `extract` interface of $\mathcal{F}_{\mathsf{wNIZK}}^{C_{\mathsf{assertions}}}$ (Assumption 1), invoked inside Lemma 14. The overall bound of Theorem 1 therefore contains both $\mathsf{Adv}^{\mathsf{sound},\mathcal{G}_{\mathsf{oGG}}}_{\mathsf{out}}$ (outer soundness, plain, no outer extraction exists or is needed, since $\mathcal{C}_{\mathsf{verif}}$'s witness is public data) and $\mathsf{Adv}^{\mathsf{Groth16}/\mathcal{G}_{\mathsf{oGG}}}_{\mathsf{in}}$ (inner extraction): different jobs, no redundant slack.

Remark 19. V does not compute the outer pairing equation on-chain, the whole point of the construction is to avoid that. Rather, V evaluates its scalar form inside the garbled circuit $\mathcal{C}_{\mathsf{GC}}$ (Definition 4, Theorem 2), using the canonical labels released against the public seed and P's signed scalars $(a_{\mathsf{out}}, b_{\mathsf{out}}, c_{\mathsf{out}}, T_{\mathsf{ic}})$, which checks 1–3 bind to the c_1-committed proof points. The output of the GC is the Boolean conclusion of pairing verification, obtained without any on-chain pairing opcode.

5.7 Beacon and ledger

Algorithm 5 Ideal functionality $\mathcal{F}_{\mathsf{Beacon}}$ (public-coin beacon)

Parameters: output length ℓ and delivery time T_{beacon}.

I. Request. On $(\texttt{request}, \mathsf{sid})$ from any party before time T_{beacon}:

1: sample $r \leftarrow \{0,1\}^{\ell}$ uniformly; store internally ▷ in the proof, $\mathcal{S}$ may bias by $\leq \varepsilon_{\mathcal{R}}$

II. Deliver. At time T_{beacon}:

2: output r to all parties through $\mathcal{S}$

Any standard commit-reveal construction with a default-loses abort penalty realises $\mathcal{F}_{\mathsf{Beacon}}$ in the $\mathcal{G}_{\mathsf{Ledger}}$-hybrid model; the distinguishing advantage $\varepsilon_{\mathcal{R}}$ is the realisation's bias bound, which under any realisation with collision-resistant commitments is at most $\mathsf{Adv}^{\mathsf{CR}}_{H}$. Alternative realisations include block-hash-based beacons (with confirmation depth controlling the bias) and external randomness beacons such as Drand bridged to Bitcoin via inclusion proofs.

The global ledger $\mathcal{G}_{\mathsf{Ledger}}$ follows the treatment of [BMTZ24]. We use only its UTXO/script interface and its time-lock enforcement.

5.8 Instantiation summary

$\Pi_{\mathsf{DV\text{-}Groth16}}$ is parametric in the instantiations of $\mathcal{F}^{+}_{\mathsf{HL}}$ and $\mathcal{F}_{\mathsf{Beacon}}$. The security bound treats these abstractly via the advantage terms $\mathsf{Adv}^{\mathsf{HL}+}$ and $\varepsilon_{\mathcal{R}}$ respectively; the on-chain commitments (c_1, com_r, com_s, $\mathsf{root}_{\mathsf{GC}}$) are hash commitments on $\mathcal{G}_{\mathsf{Ledger}}$, charged via collision-resistance $\mathsf{Adv}^{\mathsf{CR}}_{H}$. Concretely:

1. Any malicious-secure one-time signature scheme realises $\mathcal{F}^{+}_{\mathsf{HL}}$ (Lamport [Lam79], Winternitz [Mer87], or BitVMX-style hash-locks).
2. Any unbiasable commit-reveal coin-toss with abort-penalty enforcement realises $\mathcal{F}_{\mathsf{Beacon}}$. The bias bound $\varepsilon_{\mathcal{R}}$ is realisation-dependent.

6 The DV-Groth16 protocol $\Pi_{\mathsf{DV\text{-}Groth16}}$

We now present $\Pi_{\mathsf{DV\text{-}Groth16}}$, the protocol that realises $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$ (Theorem 1). The construction proceeds in three phases: an off-chain Phase I where the parties exchange the outer SRS and assemble Phase-I commitments, an on-chain Phase II for cut-and-choose and seed reveal, and a Phase III for evaluation and on-chain settlement. The outer relation R_{outer} proven by the outer Groth16 proof is defined first (Section 6.1), then each phase is given as an explicit algorithm (Sections 6.2, 6.3, and 6.4), and finally we state the honest-party guarantees that the UC theorem implies (Section 6.5).

6.1 The outer relation R_{outer}

The outer relation R_{outer} is the relation that the outer Groth16 proof π_{out} certifies: that the committed inner proof verifies under the public $\mathsf{vk}_{\mathsf{in}}$ (Definition 5). Two distinct circuits play distinct roles and are never conflated: $\mathcal{C}_{\mathsf{verif}}$ (Definition 3) is the circuit of R_{outer}, proven by π_{out} and never garbled; $\mathcal{C}_{\mathsf{GC}}$ (Definition 4) is the circuit garbled under cut-and-choose, and it verifies π_{out} *in scalar form* (Theorem 2). The inner proof is pinned directly by its inclusion in c_1; no separate hash-pin is used.

6.1.1 The outer circuit.

Definition 3. The outer circuit $\mathcal{C}_{\mathsf{verif}}$ is the arithmetic circuit over $\mathbb{F}_q$, the scalar field of the outer curve, equal by the 2-chain condition of Definition 1 to the base field of the inner curve, that, on public input $\mathsf{stmt} = (x, A_{\mathsf{in}}, B_{\mathsf{in}}, C_{\mathsf{in}})$, whose inner-point coordinates are native $\mathbb{F}_q$ elements, evaluates the inner Groth16 verification equation

$$e(A_{\mathsf{in}}, B_{\mathsf{in}}) \stackrel{?}{=} e(\alpha_{\mathsf{in}} G_1, \beta_{\mathsf{in}} G_2) \cdot e\big(\mathsf{vk}_x,\, \gamma_{\mathsf{in}} G_2\big) \cdot e(C_{\mathsf{in}}, \delta_{\mathsf{in}} G_2), \qquad \mathsf{vk}_x = (p_0 + x\, p)\, G_1,$$

in-circuit (Miller loops, final exponentiation, and the public-input term) and outputs the result. Its witness is the trace of this computation and contains no prover secret. $|\mathcal{C}_{\mathsf{verif}}| = \langle c_{\mathsf{verif}} \rangle$ R1CS constraints (working estimate $2\text{–}5 \times 10^4$ for BLS12-377 inside BW6-761, per the one-layer-composition verifier gadgets; to be pinned against gnark/arkworks benchmarks before submission). $\mathcal{C}_{\mathsf{verif}}$ is *proven* by π_{out} and is never garbled.

Definition 4. The garbled circuit $\mathcal{C}_{\mathsf{GC}}$ is the Boolean circuit garbled in s copies (Algorithm 6); it verifies π_{out} in scalar form. **Hardwired constants** (fixed at garbling time; verified against c_1 at cut-and-choose opening): the committed outer proof points $A_{\mathsf{out}}, B_{\mathsf{out}}, C_{\mathsf{out}}$, the generators, and the capsule data. **Class (A) wires** (prover-authoritative; one-time-signed on-chain in UTXO3 post-reveal, one signed value shared across all copies): the scalars $a_{\mathsf{out}}, b_{\mathsf{out}}, c_{\mathsf{out}} \in \mathbb{F}_q$. **Class (B) wires** (raw seed-derived; canonical bit of wire-bit j is $\mathsf{bit}_{\beta(j)}(H(\mathsf{seed} \,\|\, \mathtt{label}(j)))$, adjudicable through $\mathcal{F}^{+}_{\mathsf{HL}}$ once seed is public): the outer trapdoors $\alpha_{\mathsf{out}}, \beta_{\mathsf{out}}, \delta_{\mathsf{out}}$. **Class (C) wire** (derived; prover-signed, contestable through $\mathcal{F}_{\mathsf{derive}}$): the single element $T_{\mathsf{ic}} := \beta_{\mathsf{out}}\, U(\tau_{\mathsf{out}}) + \alpha_{\mathsf{out}}\, V(\tau_{\mathsf{out}}) + W(\tau_{\mathsf{out}}) \in \mathbb{F}_q$, where $U(X) = \sum_k \mathsf{stmt}_k\, u_k^{\mathsf{out}}(X)$ over the public outer-QAP statement wires, and analogously V, W, a value publicly recomputable from $(\mathsf{seed}, \mathsf{stmt})$ post-reveal.

Checks:

1. $a_{\mathsf{out}} \cdot G_1 \stackrel{?}{=} A_{\mathsf{out}}$ (fixed-base scalar multiplication, in-circuit)
2. $b_{\mathsf{out}} \cdot G_2 \stackrel{?}{=} B_{\mathsf{out}}$ (fixed-base; on BW6-761, $\mathbb{G}_2$ is also over the base field)
3. $c_{\mathsf{out}} \cdot G_1 \stackrel{?}{=} C_{\mathsf{out}}$ (fixed-base)
4. $a_{\mathsf{out}} \cdot b_{\mathsf{out}} \stackrel{?}{=} \alpha_{\mathsf{out}}\beta_{\mathsf{out}} + T_{\mathsf{ic}} + c_{\mathsf{out}}\, \delta_{\mathsf{out}}$ (three field multiplications)

If all checks pass, $\mathcal{C}_{\mathsf{GC}}$ outputs the accept label; otherwise the reject label $\lambda_0^{(i)}$, which through the capsule $u_i = k \oplus H(\mathtt{mask} \,\|\, i \,\|\, \lambda_0^{(i)})$ (pinned in c_1 via $h = H(k)$; k enters no circuit) releases the slashing key to $\mathtt{V}$. By Theorem 2, check 4 on the unique scalars passing checks 1–3 holds iff the committed $(A_{\mathsf{out}}, B_{\mathsf{out}}, C_{\mathsf{out}})$ verifies under $\mathsf{vk}_{\mathsf{outer}}(\mathsf{seed})$; by Definition 5 and the pre-reveal binding (Lemma 3), this is what enforces inner validity. The shortcut of dropping checks 1–3 and making all wires seed-derived is unsound: the garbler holds every label preimage and could, for a false statement, reveal labels for cooked values satisfying check 4 alone, the temporal anchor (points committed before trapdoor knowledge) plus the in-circuit multiplications bridging post-reveal scalars to pre-reveal points is irreducible.

Remark 20 (Pin construction: windowed fixed-base combs). Each pin check i (checks 1–3 of Definition 4), $\mathsf{s} \cdot P \stackrel{?}{=} Q$ for a public base $P \in \{G_1, G_2\}$ and a Class (A) scalar $\mathsf{s} \in \mathbb{F}_q$ ($\sim \lceil \log_2 q \rceil = 377$ bits), is realized in-circuit as a *windowed fixed-base comb*, not naive double-and-add. Fix a window width w and set $t = \lceil 377/w \rceil$ windows.

1. **Precomputation (garbling time, public, free).** For each window $j \in \{0, \ldots, t-1\}$ and each digit $u \in \{0, \ldots, 2^w - 1\}$, the garbler tabulates the public constant $P_{j,u} := (u \cdot 2^{jw})\, P$. These are hardwired, not garbled, and shared across all s copies.
2. **In-circuit accumulation.** Initialize the accumulator to $\mathcal{O}$. For each window j, a 2^w-to-1 multiplexer, driven by the w wire-bits of s in positions $[jw, jw+w)$, selects $P_{j,\cdot}$; the selected point is added to the accumulator by one *mixed* (Jacobian + affine) addition. After t windows the accumulator equals $\mathsf{s} \cdot P$, compared to Q by a coordinate equality gadget.

No point doublings occur in-circuit (they are absorbed into the precomputed table). Signed-digit (NAF-w) recoding may be used to halve the nonzero digits at the cost of storing negations; it is optional and does not affect soundness. On BW6-761 the multiplexer cost per window is $\Theta(2^w \cdot |\text{coord}|)$ AND, so the total pin cost $t \cdot \left(2^w |\text{coord}| + c_{\mathsf{add}}\right)$ is minimized near $w = 5$–6; the exact optimum is implementation-dependent and is pinned together with the other deployment benchmarks. Each in-circuit field multiplication uses a Karatsuba/Toom gadget with fast reduction specialized to the BW6-761 modulus and lazy (deferred) reduction across the point-addition formula; identical sub-blocks across the three pins are emitted once and shared (Remark 2, fan-out). All of this is engine-neutral: the resulting Boolean circuit is garbled by Algorithm 6 exactly as before.

Remark 21. The term $p_0 := \gamma_{\mathsf{in}}^{-1}(\alpha_{\mathsf{in}} v_0(\tau_{\mathsf{in}}) + \beta_{\mathsf{in}} u_0(\tau_{\mathsf{in}}) + w_0(\tau_{\mathsf{in}}))$ is the assertion verification-key term K_0 for the constant-one wire of $C_{\mathsf{assertions}}$; $p := \gamma_{\mathsf{in}}^{-1}(\alpha_{\mathsf{in}} v_1(\tau_{\mathsf{in}}) + \beta_{\mathsf{in}} u_1(\tau_{\mathsf{in}}) + w_1(\tau_{\mathsf{in}}))$ is the analogue for the public-input wire x. Both are *public* closed-form evaluations fixed by the public inner SRS (Definition 1); they are not seed-derived. They appear in the inner verification key $\mathsf{vk}_x = (p_0 + x\, p) G_1$ used by the inner verification equation (Definition 3).

6.1.2 The outer relation.

Definition 5. The relation R_{outer} certified by the outer Groth16 proof π_{out} has statement $\mathsf{stmt} := (x, A_{\mathsf{in}}, B_{\mathsf{in}}, C_{\mathsf{in}})$, all components fixed before π_{out} is produced (x public; $(A_{\mathsf{in}}, B_{\mathsf{in}}, C_{\mathsf{in}})$ the committed inner proof, bound pre-reveal via c_1), and is defined by

$$R_{\mathsf{outer}}(\mathsf{stmt}) = 1 \iff (A_{\mathsf{in}}, B_{\mathsf{in}}, C_{\mathsf{in}}) \text{ verifies under the public } \mathsf{vk}_{\mathsf{in}} \text{ on input } x,$$

i.e. the inner Groth16 verification equation of Definition 3 holds. R_{outer} is a *deterministic predicate of public data*: the witness of $\mathcal{C}_{\mathsf{verif}}$ is the trace of the verification-equation computation and contains no secret of P's. Consequently (i) the outer proof requires no zero-knowledge, and the prover coins may be fixed to $r_{\mathsf{out}} = s_{\mathsf{out}} = 0$; (ii) outer *soundness*, not knowledge-soundness, is the property the analysis consumes at this layer (Lemma 3); and (iii) "R_{outer} true $\Rightarrow$ π_{in} verifies" holds by definition, the inner/outer bridge needs no separate theorem, and Theorem 2 instead carries the scalarization identity on which the garbled circuit operates.

Definition 6. The protocol uses two Groth16 proofs at two layers. (i) The *inner proof* $\pi_{\mathsf{in}} = (A_{\mathsf{in}}, B_{\mathsf{in}}, C_{\mathsf{in}})$ is the standard Groth16 proof for $C_{\mathsf{assertions}}$ under the public reusable SRS $\mathsf{SRS}^{\mathsf{in}}$ (Definition 1) on public input x and witness w; witness-hiding is ordinary Groth16 zero-knowledge. (ii) The *outer proof* $\pi_{\mathsf{out}} = (A_{\mathsf{out}}, B_{\mathsf{out}}, C_{\mathsf{out}})$ is the Groth16 proof of R_{outer} (Definition 5) under the per-proof seed-derived $\mathsf{SRS}^{\mathsf{out}}$ (Definition 2); it attests R_{outer} (Definition 5), i.e. that π_{in} verifies, with the garbled layer checking the outer proof in scalar form (Theorem 2). The on-chain phase reveals only the compact $\mathsf{SRS}^{\mathsf{out}}$; $\mathsf{SRS}^{\mathsf{in}}$

stays public and fixed. This is the recursive structure of the protocol's Figure 1: a public heavy inner setup with a compact per-proof outer SRS.

Remark 22 (Why the prover-equation relation cannot be made sound). Earlier versions of this work defined R_{outer} by the inner *prover linear-combination equations*, with the linear-combination terms supplied either as scalars or as precomputed group elements. Both variants are unusable, for dual reasons. With point-valued inputs, the relation is *trivial*: for any $(A_{\mathsf{in}}, B_{\mathsf{in}}, C_{\mathsf{in}})$, the witness $r = s = 0$, $P := A_{\mathsf{in}} - \alpha_{\mathsf{in}} G_1$, $Q := B_{\mathsf{in}} - \beta_{\mathsf{in}} G_2$, $F := C_{\mathsf{in}} - G_1$, $\widetilde{H} := G_1$ satisfies all three shape equations, and no extraction model repairs this, an algebraic adversary who output A_{in} can output $A_{\mathsf{in}} - \alpha_{\mathsf{in}} G_1 - r\delta_{\mathsf{in}} G_1$ with a representation, so the equations are definitions of the witness points, not constraints on the statement. Rigidity requires the linear-combination terms to enter as *scalars*, but $p(\tau_{\mathsf{in}}), q(\tau_{\mathsf{in}}), h(\tau_{\mathsf{in}})$ require the destroyed inner trapdoor and are computable by no party. The relation must therefore verify the inner proof the honest way, which is Definition 5; the in-circuit fixed-base MSM alternative (witness $\{a_i\}$) costs $\Theta(|C_{\mathsf{assertions}}|)$ scalar multiplications and is dominated. The 2-chain instantiation (Definition 1) is what keeps Definition 5's circuit small.

6.2 Phase I: off-chain setup and proof generation

In Phase I, V generates the outer SRS from a fresh seed, attests to its correctness via π_{SRS}, and transmits both to P. P then produces the inner Groth16 proof on her witness, computes the outer proof π_{out} in group form (without knowing the seed-derived scalar trapdoors), and assembles the master commitment c_1 that binds all Phase-I material.

Algorithm 6 Phase I: off-chain setup and proof generation

State: Public $\mathsf{SRS}^{\mathsf{in}}, \mathsf{vk}_{\mathsf{in}}, x$; $\mathtt{P}$'s witness w; cut-and-choose parameter s

$\mathtt{V}$: sample $\mathsf{seed} \leftarrow \{0,1\}^{\lambda}$; derive the per-proof outer trapdoors $(\tau_{\mathsf{out}}, \alpha_{\mathsf{out}}, \beta_{\mathsf{out}}, \gamma_{\mathsf{out}}, \delta_{\mathsf{out}})$, each as $H(\mathsf{seed} \,\|\, \mathtt{label})$ ▷ the inner SRS $\mathsf{SRS}^{\mathsf{in}}$ is public and fixed, Definition 1

$\mathtt{V}$: compute $\mathsf{SRS}^{\mathsf{out}}_{\mathsf{group}}$ (Definition 2) and $hs := H(\mathsf{seed})$

$\mathtt{V}$: compute π_{SRS} for the seed–scalar binding

$\mathtt{V} \to \mathtt{P}$: $(\mathsf{SRS}^{\mathsf{out}}_{\mathsf{group}}, hs, \pi_{\mathsf{SRS}})$

$\mathtt{P}$: verify π_{SRS} and run the batched pairing well-formedness checks on $\mathsf{SRS}^{\mathsf{out}}_{\mathsf{group}}$; abort if either fails

$\mathtt{P}$: compute $\pi_{\mathsf{in}} \leftarrow \mathsf{Prove}_{\mathsf{in}}(\mathsf{SRS}^{\mathsf{in}}, x, w) = (A_{\mathsf{in}}, B_{\mathsf{in}}, C_{\mathsf{in}})$

$\mathtt{P}$: compute the outer proof $\pi_{\mathsf{out}} \leftarrow \mathsf{Prove}_{\mathsf{out}}\big(\mathsf{SRS}^{\mathsf{out}}_{\mathsf{group}},\ \mathsf{stmt} = (x, A_{\mathsf{in}}, B_{\mathsf{in}}, C_{\mathsf{in}});\ \mathsf{wit} = \mathsf{trace}(\mathsf{stmt});\ r_{\mathsf{out}} = s_{\mathsf{out}} = 0\big) = (A_{\mathsf{out}}, B_{\mathsf{out}}, C_{\mathsf{out}})$ ▷ deterministic; the witness is the public verification trace, Definition 5, Remark 23

$\mathtt{P}$: sample slashing secret $k \leftarrow \{0,1\}^{2\lambda}$; set $h := H(k)$

$\mathtt{P}$: generate one-time signature keys (via $\mathcal{F}^{+}_{\mathsf{HL}}$) for the Class (A)/(C) package $(a_{\mathsf{out}}, b_{\mathsf{out}}, c_{\mathsf{out}}, T_{\mathsf{ic}})$

for $i = 1$ to s **do**

  $\mathtt{P}$: garble one copy: $(\widetilde{C}_i, e_i, d_i) \leftarrow \mathsf{Gb}(1^{\lambda}, \mathcal{C}_{\mathsf{GC}})$, with the committed $(A_{\mathsf{out}}, B_{\mathsf{out}}, C_{\mathsf{out}})$ hardwired (Definition 4)

  $\mathtt{P}$: compute capsule $u_i := k \oplus H(\mathtt{mask} \,\|\, i \,\|\, \lambda_0^{(i)})$, where $\lambda_0^{(i)} \in d_i$ is the reject label

  for each Class (B) wire j **do**

    for $b \in \{0,1\}$ **do** $\mathtt{P}$: compute hash-lock $\mathsf{hl}_{i,j,b} := H(\mathtt{cl} \,\|\, i \,\|\, j \,\|\, b \,\|\, \lambda_{i,j,b})$

    end for

  end for

  $\mathtt{P}$: compute $c_i^{gc} := H(\widetilde{C}_i \,\|\, \{\mathsf{hl}_{i,j,b}\}_{j,b} \,\|\, u_i)$

end for

$\mathtt{P}$: compute $\mathsf{root}_{\mathsf{GC}} := \mathsf{MerkleRoot}(c_1^{gc}, \ldots, c_s^{gc})$

$\mathtt{P}$: sample $\rho_1 \leftarrow \{0,1\}^{\lambda}$; compute

$$c_1 := H(x \,\|\, A_{\mathsf{in}} \,\|\, B_{\mathsf{in}} \,\|\, C_{\mathsf{in}} \,\|\, A_{\mathsf{out}} \,\|\, B_{\mathsf{out}} \,\|\, C_{\mathsf{out}} \,\|\, h \,\|\, \mathsf{root}_{\mathsf{GC}} \,\|\, \rho_1)$$

Remark 23. At the end of Phase I, $\mathtt{P}$ possesses $\mathsf{SRS}^{\mathsf{out}}_{\mathsf{group}}$ in group form only: she knows no scalar in $\{\tau_{\mathsf{out}}, \alpha_{\mathsf{out}}, \beta_{\mathsf{out}}, \gamma_{\mathsf{out}}, \delta_{\mathsf{out}}\}$. The witness she used to produce π_{out} is the *public* computation trace of the verification equation (Definition 5); she computes no discrete logarithm, and the proof is deterministic ($r_{\mathsf{out}} = s_{\mathsf{out}} = 0$, the outer layer needs no zero-knowledge). Not knowing the outer trapdoor places her in the standard Groth16 soundness regime for the outer circuit throughout Phase I, which is what Lemma 3 exploits: the proof *points* $(A_{\mathsf{out}}, B_{\mathsf{out}}, C_{\mathsf{out}})$ are fixed in c_1 inside this regime, and the scalars she later signs are forced by $\mathcal{C}_{\mathsf{GC}}$'s checks 1–3 (Definition 4) to be their discrete logarithms.

6.3 Phase II: on-chain commitment, cut-and-choose, seed reveal

In Phase II, the parties post their on-chain commitments, $\mathtt{V}$ selects the cut-and-choose challenge, $\mathtt{P}$ opens the challenged garbled circuits, and $\mathtt{V}$ reveals the seed. After the seed reveal $\mathtt{P}$ signs and posts her scalar package in UTXO3 (the outer proof π_{out} itself was already produced in Phase I).

Algorithm 7 Phase II: on-chain commitment and seed reveal

State: Phase I objects

P: create $\mathsf{UTXO1}$ with deposit D and commitment c_1
V: create $\mathsf{UTXO2}$ with deposit $D_{\mathtt{V}}$ (the verifier's bond) and seed commitment hs ▷ $\mathsf{UTXO2}$'s T_2 path is forfeiture: absent a seed-reveal spend by $T_2 < T_1$, P claims $D_{\mathtt{V}}$ (Lemma 6)
Both parties wait for confirmation of $\mathsf{UTXO1}$ and $\mathsf{UTXO2}$
V: sample challenge $J \subset [s]$ with $|J| = s/2$ from $\mathcal{F}_{\mathsf{Beacon}}$
$\mathtt{V} \to \mathtt{P}$: challenge set J
for each $i \in J$ **do**
    P: open circuit i by revealing $(\widetilde{C}_i, e_i, d_i, \rho_i^{gc}, \{\lambda_{i,j,b}\}_{j,b})$
    V: verify that $\widetilde{C}_i$ is a correct garbling of $\mathcal{C}_{\mathsf{GC}}$ with the opened randomness, that its hardwired constants equal the c_1-committed $(A_{\mathsf{out}}, B_{\mathsf{out}}, C_{\mathsf{out}})$, and that c_i^{gc} matches the $\mathsf{root}_{\mathsf{GC}}$ Merkle path
    V: verify that each $\mathsf{hl}_{i,j,b}$ matches the opened labels
end for
if any opened circuit or any hash-lock fails verification **then**
    V: abort and publish opening as commitment-inconsistency evidence
end if
$\mathtt{V} \to \mathtt{P}$: seed over authenticated off-chain channel
P: verify $H(\mathsf{seed}) = hs$; abort if not
V (or P cooperatively): spend $\mathsf{UTXO2}$ via seed-reveal path, making seed public on-chain
Both parties: derive the outer trapdoors $(\tau_{\mathsf{out}}, \alpha_{\mathsf{out}}, \beta_{\mathsf{out}}, \gamma_{\mathsf{out}}, \delta_{\mathsf{out}})$ from the public seed, each as $H(\mathsf{seed} \,\|\, \mathtt{label})$ ▷ the inner SRS is public; the derivation is modelled with H a random oracle
P: **seed-expansion check**: recompute the atomic points $\alpha_{\mathsf{out}}G_1, \beta_{\mathsf{out}}G_1, \beta_{\mathsf{out}}G_2, \gamma_{\mathsf{out}}G_2, \delta_{\mathsf{out}}G_1, \delta_{\mathsf{out}}G_2, \tau_{\mathsf{out}}G_1, \tau_{\mathsf{out}}G_2$ from the derived trapdoors and compare with the published $\mathsf{SRS}^{\mathsf{out}}_{\mathsf{group}}$ ▷ $O(1)$ fixed-base multiplications; with the Phase-I internal-consistency checks, atomic equality implies $\mathsf{SRS}^{\mathsf{out}}_{\mathsf{group}} = \mathsf{ExpandSRS}(\mathsf{seed})$, Lemma 4
if the seed-expansion check fails **then**
    P: **publish no** $\mathsf{UTXO3}$; wait for the timelock-T_1 refund (Lemma 4)
end if
P: compute $(a_{\mathsf{out}}, b_{\mathsf{out}}, c_{\mathsf{out}})$ from her outer assignment and the now-public trapdoors, and T_{ic} from $(\mathsf{seed}, \mathsf{stmt})$; sign and publish in $\mathsf{UTXO3}$: the c_1 openings (including π_{out}'s points and the nonce ρ_1), her one-time signatures on $(a_{\mathsf{out}}, b_{\mathsf{out}}, c_{\mathsf{out}}, T_{\mathsf{ic}})$, and the canonical-bit label reveals $\{\lambda_{i,j,\mathsf{cb}_j}\}$ for every Class (B) wire of every surviving circuit

The package P publishes requires no discrete-logarithm computation beyond her own data: post seed-reveal she knows the outer trapdoors (public) and her own outer assignment, hence the genuine $(a_{\mathsf{out}}, b_{\mathsf{out}}, c_{\mathsf{out}})$; and T_{ic} is a public function of $(\mathsf{seed}, \mathsf{stmt})$.

Remark 24. P's outer proof π_{out} is produced in Phase I (Algorithm 6) from the group-form outer SRS, before the seed is revealed. Since she does not hold the outer trapdoor in Phase I, she is in the standard Groth16 soundness regime for the outer circuit throughout Phase I; this is the epistemic discipline exploited by Lemma 3, and it is the temporal anchor of the construction: the proof *points* are fixed while forging is infeasible, and the post-reveal scalars are forced by $\mathcal{C}_{\mathsf{GC}}$'s checks 1–3 to be the discrete logarithms of exactly those points.

Lemma 3 (Pre-reveal outer soundness)**.** *Under Assumption 2 and the Groth16-over-$\mathcal{G}_{\mathsf{oGG}}$ guarantees of Assumption 1 applied to the outer circuit $\mathcal{C}_{\mathsf{verif}}$, for every PPT adversary $\mathcal{A}$ playing corrupt* P *and making at most q queries to $\mathcal{G}_{\mathsf{oGG}}$, the probability that the c_1-*

committed outer proof $\pi^{\mathcal{A}}_{\mathsf{out}} = (A_{\mathsf{out}}, B_{\mathsf{out}}, C_{\mathsf{out}})$ *verifies under* $\mathsf{vk}_{\mathsf{outer}}(\mathsf{seed})$ *on a statement* stmt *with* $R_{\mathsf{outer}}(\mathsf{stmt}) = 0$ *is at most* $\mathsf{Adv}^{\mathsf{sound},\mathcal{G}_{\mathsf{oGG}}}_{\mathsf{out}}(\lambda, q, |\mathcal{C}_{\mathsf{verif}}|) \;+\; \mathsf{Adv}^{\mathsf{ZK}}_{\pi_{\mathsf{SRS}}}(\lambda)$.

Proof. **The event, precisely.** Run the real Phase I with honest $\mathtt{V}$ and corrupt $\mathtt{P} = \mathcal{A}$ in the $\mathcal{G}_{\mathsf{oGG}}$-hybrid model, and stop the execution at the instant $\mathcal{A}$ posts the master commitment c_1. Let $(\mathsf{stmt}, \pi^{\mathcal{A}}_{\mathsf{out}})$ be the statement and outer-proof points bound by that commitment; both $\mathsf{stmt} = (x, A_{\mathsf{in}}, B_{\mathsf{in}}, C_{\mathsf{in}})$ and $\pi^{\mathcal{A}}_{\mathsf{out}} = (A_{\mathsf{out}}, B_{\mathsf{out}}, C_{\mathsf{out}})$ are fixed by that commitment, but they are *not* recovered from $\mathcal{G}_{\mathsf{oGG}}$ observability: $c_1 = H(x \,\|\, A_{\mathsf{in}} \,\|\, B_{\mathsf{in}} \,\|\, C_{\mathsf{in}} \,\|\, A_{\mathsf{out}} \,\|\, B_{\mathsf{out}} \,\|\, C_{\mathsf{out}} \,\|\, h \,\|\, \mathsf{root}_{\mathsf{GC}} \,\|\, \rho_1)$ is a hash commitment with hidden nonce ρ_1, so the group-operation log exposes the *set* of handles $\mathcal{A}$ has computed but not which are sealed in c_1. Since H is the random oracle answered by the reduction (item 5) and $\mathcal{A}$ can open c_1 only by exhibiting a preimage, $\mathcal{A}$ has, except for the unqueried case bounded at the end of this proof, queried H on the full preimage, from which the reduction reads $(\mathsf{stmt}, \pi^{\mathcal{A}}_{\mathsf{out}})$; the observability of $\mathcal{G}_{\mathsf{oGG}}$ ([BFKT24], §4) supplies their symbolic representations only afterwards, where the win transfer needs them. Define the event

$$\mathsf{Bad} \;:=\; \big[\; \pi^{\mathcal{A}}_{\mathsf{out}} \text{ verifies under } \mathsf{vk}_{\mathsf{outer}}(\mathsf{seed}) \;\wedge\; R_{\mathsf{outer}}(\mathsf{stmt}) = 0 \;\big].$$

We bound $\Pr[\mathsf{Bad}]$.

Reduction $\mathcal{B}$ to outer Groth16 soundness. Let $\mathcal{C}$ be the challenger of the Groth16-over-$\mathcal{G}_{\mathsf{oGG}}$ soundness game for the fixed circuit $\mathcal{C}_{\mathsf{verif}}$: $\mathcal{C}$ samples a fresh outer trapdoor $(\tau_{\mathsf{out}}, \alpha_{\mathsf{out}}, \beta_{\mathsf{out}}, \gamma_{\mathsf{out}}, \delta_{\mathsf{out}})$, produces the corresponding SRS $\mathsf{SRS}^{\mathsf{out}}_{\mathsf{group}}$ in group form (proving and verification keys), hands it to $\mathcal{B}$, and exposes the $\mathcal{G}_{\mathsf{oGG}}$ oracle; $\mathcal{B}$ wins if it outputs $(\mathsf{stmt}^\star, \pi^\star_{\mathsf{out}})$ with $\pi^\star_{\mathsf{out}}$ verifying under the derived $\mathsf{vk}_{\mathsf{outer}}$ and $\mathcal{C}_{\mathsf{verif}}(\mathsf{stmt}^\star) = 0$ (equivalently $R_{\mathsf{outer}}(\mathsf{stmt}^\star) = 0$, since $\mathcal{C}_{\mathsf{verif}}$ evaluates R_{outer}). $\mathcal{B}$ simulates honest $\mathtt{V}$'s Phase I to $\mathcal{A}$ as follows.

1. *Inner SRS.* Use the genuine public ceremony output $\mathsf{SRS}^{\mathsf{in}}$ (Definition `def:srs1`); it is identical in both worlds.

2. *Outer SRS.* Forward $\mathcal{C}$'s $\mathsf{SRS}^{\mathsf{out}}_{\mathsf{group}}$ as $\mathtt{V}$'s outer SRS.

3. *Seed commitment.* Sample $\mathsf{seed}^* \leftarrow \{0,1\}^\lambda$ uniformly and set $hs^* = H(\mathsf{seed}^*)$, a genuine hash commitment.

4. *Scalar commitments and SRS proof.* Form the statement commitments $\mathsf{com}_{\vec{\sigma}}$ as commitments to dummy values and obtain $\pi^*_{\mathsf{SRS}} \leftarrow \mathsf{Sim}_{\pi_{\mathsf{SRS}}}(\mathsf{SRS}^{\mathsf{out}}_{\mathsf{group}}, hs^*, \mathsf{com}_{\vec{\sigma}})$ from the simulator guaranteed by Assumption `ass:piSRS-zk` (the $\mathcal{F}^{R_{\mathsf{SRS}}}_{\mathsf{wNIZK}}$ simulation interface, whose statement carries $\mathsf{com}_{\vec{\sigma}}$).

5. Send $(\mathsf{SRS}^{\mathsf{out}}_{\mathsf{group}}, hs^*, \mathsf{com}_{\vec{\sigma}}, \pi^*_{\mathsf{SRS}})$ to $\mathcal{A}$ as the Phase I message, relay every $\mathcal{G}_{\mathsf{oGG}}$ query of $\mathcal{A}$ to $\mathcal{C}$'s oracle, and answer each random-oracle query $H(\cdot)$ lazily with a fresh uniform value (with $H(\mathsf{seed}^*)$ pinned to hs^*).

6. When $\mathcal{A}$ posts c_1, locate the random-oracle query whose output is c_1 in the table maintained in item 5 (unique up to the collision term below), parse $(\mathsf{stmt}, \pi^{\mathcal{A}}_{\mathsf{out}}) = (x, A_{\mathsf{in}}, B_{\mathsf{in}}, C_{\mathsf{in}}, A_{\mathsf{out}}, B_{\mathsf{out}}, C_{\mathsf{out}})$ from its preimage, output $(\mathsf{stmt}, \pi^{\mathcal{A}}_{\mathsf{out}})$ to $\mathcal{C}$, and *halt.*

Why the halt makes the simulation clean. The seed is transmitted only at Phase II.5 and the seed-dependent reveals (the canonical-bit openings $H(\mathsf{seed} \,\|\, \mathtt{label})$, the public batched-pairing well-formedness check relating $\mathsf{SRS}^{\mathsf{out}}$ to seed) occur only at Phase II.5 and later. This is distinct from the batched check $\mathtt{P}$ runs on receipt (the Phase-I line "verify π_{SRS} and run the batched pairing well-formedness checks on $\mathsf{SRS}^{\mathsf{out}}_{\mathsf{group}}$"), which tests only the internal consistency of $\mathsf{SRS}^{\mathsf{out}}_{\mathsf{group}}$ and is *independent* of seed; the externally-sampled SRS $\mathcal{B}$ forwards from $\mathcal{C}$ is a genuine Groth16 SRS and passes it unchanged, so the halt at c_1 leaves

no seed-dependent obligation outstanding. Because $\mathcal{B}$ halts at c_1, strictly before Phase II.5, it is *never* required to reveal seed^* or to answer any query whose consistency would expose that $\mathcal{C}$'s SRS is not the seed-expansion of seed^*. This is the precise point at which the earlier programmable-PRF reduction failed: there the reduction had to keep simulating past the reveal, where the publicly checkable relation $\mathsf{SRS}^{\mathsf{out}} = \mathsf{ExpandSRS}(\mathsf{PRF}(\mathsf{seed}, \cdot))$ contradicted an embedded, seed-independent SRS. The cutoff removes that obligation entirely.

View distance up to c_1. Compare $\mathcal{A}$'s view in the reduction with its view in the real pre-reveal game.

- $\mathsf{SRS}^{\mathsf{in}}$: identical (item 1), zero distance.
- $\mathsf{SRS}^{\mathsf{out}}_{\mathsf{group}}$: in the real world the outer trapdoor is seed-derived. Modelling the derivation as a random oracle, the trapdoor is uniform and $\mathcal{C}$'s trapdoor is uniform by sampling; a Groth16 SRS is a deterministic function of its trapdoor, so the two SRS distributions are identical, zero distance.
- $hs^* = H(\mathsf{seed}^*)$ for uniform seed^*: distributed exactly as the real $hs = H(\mathsf{seed})$, zero distance.
- Random-oracle answers: the only way an answer could be inconsistent is if $\mathcal{A}$ queries $H(\mathsf{seed}^* \,\|\, \cdot)$ or $H(\mathsf{seed}^*)$ before c_1. Since seed^* is information-theoretically hidden behind hs^* at this stage, each such query occurs with probability at most $2^{-\lambda}$, so over q queries the answers are consistent except with probability $q\,2^{-\lambda}$, which is absorbed into $\mathsf{Adv}^{\mathsf{sound},\mathcal{G}_{\mathsf{oGG}}}_{\mathsf{out}}$.
- $(\mathsf{com}_{\vec{\sigma}}, \pi_{\mathsf{SRS}})$: real commitments-and-proof versus dummy commitments with a simulated proof. By Assumption `ass:piSRS-zk` (the $\mathcal{F}^{R_{\mathsf{SRS}}}_{\mathsf{wNIZK}}$ simulator produces the statement commitments together with the proof, indistinguishably from honest production), the two are distinguishable with advantage at most $\mathsf{Adv}^{\mathsf{ZK}}_{\pi_{\mathsf{SRS}}}$.

Hence $\big|\Pr[\mathsf{Bad} \text{ in reduction}] - \Pr[\mathsf{Bad} \text{ in real game}]\big| \le \mathsf{Adv}^{\mathsf{ZK}}_{\pi_{\mathsf{SRS}}} + q\,2^{-\lambda}$.

Win transfer. On Bad, $\pi^{\mathcal{A}}_{\mathsf{out}}$ verifies under $\mathsf{vk}_{\mathsf{outer}}$ and $R_{\mathsf{outer}}(\mathsf{stmt}) = 0$, so $(\mathsf{stmt}, \pi^{\mathcal{A}}_{\mathsf{out}})$ is exactly a winning pair for $\mathcal{C}$; thus $\Pr[\mathcal{B} \text{ wins}] \ge \Pr[\mathsf{Bad} \text{ in reduction}]$. By Assumption `ass:groth16-inner` applied to $\mathcal{C}_{\mathsf{verif}}$ (the `verify` interface: no extraction is invoked, R_{outer} being a deterministic predicate, Definition `def:Router`), $\Pr[\mathcal{B} \text{ wins}] \le \mathsf{Adv}^{\mathsf{sound},\mathcal{G}_{\mathsf{oGG}}}_{\mathsf{out}}(\lambda, q, |\mathcal{C}_{\mathsf{verif}}|)$. Combining,

$$\Pr[\mathsf{Bad}] \;\le\; \mathsf{Adv}^{\mathsf{sound},\mathcal{G}_{\mathsf{oGG}}}_{\mathsf{out}}(\lambda, q, |\mathcal{C}_{\mathsf{verif}}|) + \mathsf{Adv}^{\mathsf{ZK}}_{\pi_{\mathsf{SRS}}}(\lambda),$$

the $q\,2^{-\lambda}$ term having been absorbed into $\mathsf{Adv}^{\mathsf{sound},\mathcal{G}_{\mathsf{oGG}}}_{\mathsf{out}}$.

Unqueried commitment. If $\mathcal{A}$ posts c_1 without a prior H-query on a preimage of the form above, it can produce a valid Phase-III opening only by finding a preimage of the fixed value c_1 under H; on the complement either $\mathcal{A}$ never opens c_1, so no verifying π_{out} is settled and Bad does not occur, or its opening exhibits the preimage already read in item 6. This preimage-finding event, together with the uniqueness of the matching query (no H-collision), is of the same order as $\mathsf{Adv}^{\mathsf{CR}}_{H}$ over q queries and is absorbed into $\mathsf{Adv}^{\mathsf{sound},\mathcal{G}_{\mathsf{oGG}}}_{\mathsf{out}}$ alongside the existing $q\,2^{-\lambda}$ term. □

Remark 25. The reduction never extracts a witness for π_{out} and never rewinds: it reads the committed points off the random-oracle preimage of c_1 and stops. Soundness, not knowledge-soundness, is therefore the only outer property consumed, consistent with Definition `def:Router`.

6.4 Phase III: evaluation and settlement

In Phase III, V verifies the on-chain commitments and openings, derives canonical labels from the revealed seed, evaluates each surviving garbled circuit, and triggers the appropriate settlement transaction.

Algorithm 8 Phase III: evaluation and settlement

State: Confirmed $\mathsf{UTXO1}, \mathsf{UTXO2}, \mathsf{UTXO3}$
Interfaces: Outcome in $\{\texttt{slash-fast}, \texttt{slash-commit}, \texttt{refund}\}$

V: parse the $\mathsf{UTXO3}$ package: c_1 openings, P's one-time signatures on $(a_{\mathsf{out}}, b_{\mathsf{out}}, c_{\mathsf{out}}, T_{\mathsf{ic}})$, and the Class (B) canonical-label reveals
V: verify $c_1 = H$(reconstructed concat. from $\mathsf{UTXO3}$ openings and $\mathsf{UTXO1}$ contents)
if commitment consistency fails **then**
  return `slash-commit` ▷ Tier 2, evidence (i) of Lemma 5
end if
V: verify P's signatures on $(a_{\mathsf{out}}, b_{\mathsf{out}}, c_{\mathsf{out}}, T_{\mathsf{ic}})$; on failure or equivocation, **return** `slash-commit` ▷ evidence (ii)
V: recompute the canonical bits $\mathsf{cb}_j := \mathsf{bit}_{\beta(j)}\big(H(\mathsf{seed} \,\|\, \texttt{label}(j))\big)$ for each Class (B) wire j and check the revealed labels against the committed $\mathsf{hl}_{i,j,\mathsf{cb}_j}$; on a wrong-bit reveal, **return** `slash-commit` via $\mathcal{F}_{\mathsf{HL}}^{+}$ adjudication ▷ evidence (iii)
V: recompute T_{ic} from $(\mathsf{seed}, \mathsf{stmt})$ and compare with the signed value; on mismatch, open an $\mathcal{F}_{\mathsf{derive}}$ dispute; on winning, **return** `slash-commit` ▷ evidence (iv)
V (off-chain, optional early check): verify the committed π_{out} against $\mathsf{vk}_{\mathsf{outer}}(\mathsf{seed})$ on statement stmt ▷ π_{out} was bound pre-reveal via c_1; soundness derives from the pre-reveal commitment (Lemma 3), not from post-reveal verification, a public trapdoor destroys soundness for anything committed after it. If π_{out} is invalid, the evaluation below releases k (Tier 1); no separate script path is needed
for each surviving circuit $i \in [s] \setminus J$ **do**
  V: assemble input labels: Class (A)/(C) labels selected by P's signed values under e_i; Class (B) labels from P's verified canonical reveals
  V: evaluate $\widetilde{C}_i$ on the assembled labels to obtain output label Λ_i
  if Λ_i is the reject label $\lambda_0^{(i)}$ **then** V: compute $k := u_i \oplus H(\texttt{mask} \,\|\, i \,\|\, \Lambda_i)$
    if $H(k) = h$ **then return** `slash-fast` ▷ Tier 1: GC rejects, k released
    end if
  end if
end for
return `refund` after timelock T_1

6.5 Honest-party guarantees

The UC emulation of Theorem 1 implies several honest-party guarantees that are worth stating explicitly. We collect them here: completeness for the all-honest case (Section 6.5.1), designated-evaluator privacy (Section 6.5.2), and malicious-verifier resistance (Section 6.5.3). The first is immediate from the all-honest simulator (Algorithm 9 in Section 7); the second follows from the privacy of the garbling scheme combined with the seed-derived canonical-label policy; the third is the cryptographic content of the corrupt-V case and is the subject of four lemmas leading to Lemma 8.

6.5.1 Completeness

Definition 7. The protocol $\Pi_{\mathsf{DV\text{-}Groth16}}$ is *complete* if, whenever P holds a valid witness w with $C_{\mathsf{assertions}}(x; w) = 1$ and both parties follow the protocol honestly, Algorithm 8 outputs `refund` except with probability $\mathsf{negl}(\lambda)$.

Proposition 1. $\Pi_{\mathsf{DV\text{-}Groth16}}$ *is complete in the sense of Definition 7.*

Proof. Honest P's π_{in} satisfies the inner pairing equation, so $R_{\mathsf{outer}}(\mathsf{stmt}) = 1$ (Definition 5). Since V is honest, $\mathsf{SRS}^{\mathsf{out}}_{\mathsf{group}} = \mathsf{ExpandSRS}(\mathsf{seed})$, so the seed-expansion check of Algorithm 7 passes (Lemma 4, completeness direction) and her honestly-computed π_{out} verifies under $\mathsf{vk}_{\mathsf{outer}}(\mathsf{seed})$. Post seed-reveal she derives the outer trapdoors from the public seed; because the SRS she proved under is the expansion of exactly these trapdoors, the scalars she computes, e.g. $a_{\mathsf{out}} = \alpha_{\mathsf{out}} + p_{\mathsf{out}}(\tau_{\mathsf{out}}) + r_{\mathsf{out}}\delta_{\mathsf{out}}$ (with $r_{\mathsf{out}} = 0$) from her outer assignment, are the genuine discrete logarithms of her committed proof points, every quantity she must publish is computable from her own data plus the revealed seed. By Theorem 2 the genuine $(a_{\mathsf{out}}, b_{\mathsf{out}}, c_{\mathsf{out}})$ satisfy check 4 of $\mathcal{C}_{\mathsf{GC}}$, and they satisfy checks 1–3 by definition; the genuine Class (B) values and T_{ic} match the canonical bits and the $\mathcal{F}_{\mathsf{derive}}$ derivation by construction. Every surviving circuit evaluates to the accept label, so no k is released; all c_1 openings are consistent, signatures verify, and no evidence of Lemma 5's set exists, so no Tier-2 path is enabled. The refund-by-timelock path fires and P recovers D.

□

6.5.2 Designated-evaluator privacy

Proposition 2. *For an honest* P *in the both-honest execution, any party other than* V *observing the on-chain transcript and any off-chain leakage learns only the public input x, the proofs' public data, and the final outcome* out*. In particular the assertion witness w remains hidden, and before settlement no party other than* V *can evaluate the surviving garbled circuits or extract the slashing key. We call this property* designated evaluation/enforcement*. It is weaker than designated-verifier deniability: with a ceremony inner SRS, a revealed π_{in} is transferable evidence (anyone verifies it against the public $\mathsf{vk}_{\mathsf{in}}$, and no party holds a trapdoor with which it could have been forged). Deniability is achievable by the variant in which the per-session trapdoor SRS covers $C_{\mathsf{assertions}}$ itself, at per-proof setup cost linear in $|C_{\mathsf{assertions}}|$ and without compatibility with pre-existing proofs (Section 9.2).*

Proof. Witness-hiding is the ordinary zero-knowledge of inner Groth16 under the public $\mathsf{SRS}^{\mathsf{in}}$ (Remark 12), surfaced as the `simulate` interface of $\mathcal{F}_{\mathsf{wNIZK}}^{C_{\mathsf{assertions}}}$; no trapdoor simulation of π_{in} exists or is invoked.

The semantic opacity of the surviving garbled circuits follows from the BHR privacy property [BHR12] of the garbling scheme combined with the canonical-label policy of Algorithm 3. Concretely: the only party able to assemble canonical input labels for the Bob-authoritative wires of a surviving circuit $i \in [s] \setminus J$ is the holder of seed, which post-Phase-II is V. The Alice-authoritative input labels are determined by P's signed scalars, but evaluating $\widetilde{C}_i$ requires both, so any party without canonical labels cannot evaluate. Mere inspection of $\widetilde{C}_i$ reveals nothing beyond garbled-circuit privacy, which by [BHR12] is bounded by $\mathsf{Adv}^{\mathsf{garb}}_{\mathsf{BHR}}$.

The on-chain outcome is publicly visible by design (cf. Remark 3); this is the deliberate information leakage of $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$ to $\mathcal{Z}$ via $\mathcal{G}_{\mathsf{Ledger}}$. No further information leaks beyond this outcome. □

6.5.3 Malicious-verifier resistance

A malicious V might attempt to trigger slashing against an honest P by any of the following strategies. We show each is either precluded by the structure of the outcome-triggering script conditions or reduces to breaking an underlying primitive.

Attack surface. The strategies available to a malicious V are:

1. Refuse to open canonical hash-locks, preventing evaluation.
2. Publish a false canonical-bit mismatch claim (evidence item (iii) of Lemma 5), attempting to trigger `slash-commit`.
3. Refuse to transmit seed off-chain, preventing P from producing UTXO3.
4. Substitute non-canonical labels during evaluation, attempting to force the GC to produce a reject label and disclose k.
5. Publish an internally consistent outer SRS whose trapdoor scalars are *not* the expansion of the seed committed in hs, attempting to make honest P's signed scalars fail checks 1–3 of $\mathcal{C}_{\mathsf{GC}}$.

The protocol's defences against these, stated as cryptographic properties of the outcome-triggering conditions, are the following lemmas.

Lemma 4. *Suppose* V *publishes an internally consistent* $\mathsf{SRS}^{\mathsf{out}}_{\mathsf{group}}$ *whose trapdoor scalars are not* $\big(H(\mathsf{seed} \,\|\, \mathtt{tau}), \ldots, H(\mathsf{seed} \,\|\, \mathtt{delta})\big)$ *for the seed committed in hs. Then honest* P*'s seed-expansion check (Algorithm 7) fails, she publishes no* UTXO3*, and the session settles* ***refund*** *at* T_1*. Conversely, against an honest* V *the check passes, so the guard never harms completeness.*

Proof. Failure direction: the check compares the atomic points against the seed-derived scalars, so it fails under the hypothesis. Without the UTXO3 package, V holds neither the Class (A)/(C) signed values nor the Class (B) canonical reveals, so no surviving circuit is evaluable and the reject label (hence k) is unobtainable except with probability $\mathsf{Adv}^{\mathsf{garb}}_{\mathsf{BHR}}$ (Assumption 3, authenticity); each `slash-commit` evidence item (i)–(iv) of Lemma 5 presupposes UTXO3 contents, signatures, or reveals, none of which exist; the timelock-T_1 path of Lemma 6 then fires. This matches $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$'s clauses III/IV, which give honest P `refund` on a true statement regardless of V's behaviour. Completeness direction: atomic-point equality, combined with the Phase-I batched pairing checks (which chain every composite SRS element to the atomic points), implies $\mathsf{SRS}^{\mathsf{out}}_{\mathsf{group}} = \mathsf{ExpandSRS}(\mathsf{seed})$; an honest V's SRS satisfies this by construction. □

Remark 26. Lemma 4 protects P's deposit but leaves strategy (5) costless for V, who can force P to burn her Phase-I garbling work. A deployment may additionally bind $h_{\mathsf{SRS}} := H(\mathsf{SRS}^{\mathsf{out}}_{\mathsf{group}})$ into UTXO2 alongside hs; the predicate "the committed SRS is not the expansion of the revealed seed" then becomes decidable from on-chain data post-reveal, and can gate a forfeiture spend of $D_{\mathtt{V}}$ analogous to the T_2 path, adjudicated as in $\mathcal{F}^{+}_{\mathsf{HL}}/\mathcal{F}_{\mathsf{derive}}$. This adds no term to the bound of Theorem 1 and is not required for it; we leave the script realisation to the deployment companion.

Lemma 5. ***slash-commit*** *is enabled only by positive on-chain evidence from the following set: (i) an opening of* UTXO3 *contents that do not hash to* c_1*; (ii) failure or equivocation of* P*'s one-time signatures on the scalar package (two valid openings of one wire are script-checkable evidence,* $\mathcal{F}^{+}_{\mathsf{HL}}$*); (iii) a Class (B) canonical-label reveal whose bit differs from* $\mathsf{bit}_{\beta(j)}(H(\mathsf{seed} \,\|\, \mathtt{label}(j)))$ *under the publicly revealed seed, adjudicated through* $\mathcal{F}^{+}_{\mathsf{HL}}$*; (iv) a lost* $\mathcal{F}_{\mathsf{derive}}$ *dispute on the signed derived value* T_{ic}*. Refusal-to-evaluate is not such an event; it does not enable* ***slash-commit****.*

Proof. By construction of the UTXO1 script, which enables the `slash-commit` spend path only with a witness satisfying the positive-evidence condition. Refusal produces no transaction at all. □

Lemma 6. *If* V *never evaluates,* UTXO1*'s timelock-T_1 refund path fires and* P *recovers D. Further,* UTXO2 *carries a forfeiture path at timelock $T_2 < T_1$: if* V *has not made* seed *public by T_2,* P *claims* V*'s bond $D_{\mathtt{V}}$ and falls back to the timelock-T_1 refund. No script can compel revelation of a secret held by one party; the achievable guarantee is economic (bond forfeiture plus refund), and it is the one the protocol provides.*

Proof. Immediate from the timelock ordering $T_2 < T_1$ and the script conditions: the T_2 path pays $D_{\mathtt{V}}$ to P absent a prior seed-reveal spend, and the T_1 path refunds D. □

Lemma 7. *The capsule construction $u_i = k \oplus H(\mathtt{mask} \| i \| \lambda_0^{(i)})$ binds k to the specific reject label $\lambda_0^{(i)}$ of the honestly-garbled circuit i. If* V *evaluates circuit i with any input strings other than the labels released to it (the canonical Class (B) labels revealed by* P *and the labels selected by* P*'s signed Class (A)/(C) values), the output label Λ_i obtained satisfies $\Lambda_i \neq \lambda_0^{(i)}$ except with probability $\mathsf{Adv}^{\mathsf{garb}}_{\mathsf{BHR}}(\lambda)$, by garbling authenticity (Assumption 3). Hence $u_i \oplus H(\mathtt{mask} \| i \| \Lambda_i) \neq k$ and the hashlock check $H(k) \stackrel{?}{=} h$ fails.*

Proof. BHR garbling authenticity guarantees that evaluating an honestly-garbled circuit with non-canonical inputs produces, with overwhelming probability, a value outside the two canonical output labels $\{\lambda_0^{(i)}, \lambda_1^{(i)}\}$. The probability of producing $\lambda_0^{(i)}$ specifically is bounded by $\mathsf{Adv}^{\mathsf{garb}}_{\mathsf{BHR}}$. □

Lemma 8. *For the corrupt-*V *case, the probability that $\mathcal{A}$ drives $\mathcal{G}_{\mathsf{Ledger}}$ to emit* `slash-fast` *or* `slash-commit` *when* P *is honest is at most*

$$\mathsf{Adv}^{\mathsf{CR}}_{H}(\lambda) + \mathsf{Adv}^{\mathsf{garb}}_{\mathsf{BHR}}(\lambda) + \mathsf{Adv}^{\mathsf{HL+}}(\lambda) + \mathsf{Adv}^{\mathsf{derive}}(\lambda).$$

Proof. Let E_{slash} denote the event that $\mathcal{G}_{\mathsf{Ledger}}$ emits `slash-fast` or `slash-commit` against honest P, partitioned by wire class and evidence type (Definition 4).

E_0 **(SRS-expansion mismatch).** If V published a $\mathsf{SRS}^{\mathrm{group}}_{\mathrm{out}}$ that is not $\mathsf{ExpandSRS}(\mathsf{seed})$ for the revealed seed, honest P's seed-expansion check (Algorithm 7, Lemma 4) fails and she publishes no UTXO3. With no scalar package, no surviving circuit is evaluable and no evidence item of Lemma 5 exists, so $\mathcal{G}_{\mathsf{Ledger}}$ emits neither slashing outcome and the T_1 path settles `refund`: $\Pr[E_0 \wedge \mathsf{slash}] = 0$. Conditioning on the complement (the check passes), $\mathsf{SRS}^{\mathrm{group}}_{\mathrm{out}} = \mathsf{ExpandSRS}(\mathsf{seed})$, so honest P's committed π_{out} verifies under $\mathsf{vk}_{\mathsf{outer}}(\mathsf{seed})$; all remaining cases are analysed under this conditioning.

E_1 **(`slash-fast`: forging the reject label or k).** Under the conditioning of E_0, honest P's committed π_{out} verifies under $\mathsf{vk}_{\mathsf{outer}}(\mathsf{seed})$, so by Corollary 1 every honestly-garbled surviving circuit accepts on the genuine inputs. A corrupt V holds, per wire, only the canonical label revealed by P (class B) and the labels selected by P's signed scalars (classes A, C); evaluation on garbage strings yields the specific reject label $\lambda_0^{(i)}$ only with garbling-authenticity advantage (Lemma 7); embedding over the s circuits folds the factor s into $\mathsf{Adv}^{\mathsf{garb}}_{\mathsf{BHR}}$ as in Lemma 9. Direct inversion of $h = H(k)$ is absorbed into $\mathsf{Adv}^{\mathsf{HL+}}$. Hence $\Pr[E_1] \leq \mathsf{Adv}^{\mathsf{garb}}_{\mathsf{BHR}} + \mathsf{Adv}^{\mathsf{HL+}}$.

E_2 **(`slash-commit`: false positive evidence).** By Lemma 5, `slash-commit` requires positive evidence from the enumerated set. $E_{2,a}$: an opening of UTXO3 contents inconsistent with c_1 against honest P requires a hash collision: $\Pr[E_{2,a}] \leq \mathsf{Adv}^{\mathsf{CR}}_{H}$ (Lemma 1). $E_{2,b}$: a false wrong-bit ruling against honest P's Class (B) label reveals requires subverting the canonical-bit adjudication of $\mathcal{F}^{+}_{\mathsf{HL}}$, the bit of $H(\mathsf{seed} \| \mathtt{label})$ is a deterministic public value once seed is on-chain, so this is bounded by $\mathsf{Adv}^{\mathsf{HL+}}$. $E_{2,c}$: a false dispute outcome against honest P's signed T_{ic}, a deterministic public computation from $(\mathsf{seed}, \mathsf{stmt})$, is bounded

by $\mathsf{Adv}^{\mathsf{derive}}$ (the soundness of the $\mathcal{F}_{\mathsf{derive}}$ realisation). Equivocation evidence cannot exist against an honest signer. Hence $\Pr[E_2] \leq \mathsf{Adv}^{\mathsf{CR}}_{H} + \mathsf{Adv}^{\mathsf{HL}+} + \mathsf{Adv}^{\mathsf{derive}}$.

Refusal. By Lemmas 5 and 6, refusal triggers no slashing branch and contributes zero. Combining the disjoint events E_0, E_1, E_2 (and refusal) yields the stated bound; E_0 contributes no term. The embedding of this partition into the corrupt-verifier hybrid sequence is Lemma 25. □

Remark 27. Proposition 8 is invoked in the proof of Theorem 5 to bound the probability that the real-world transcript diverges from the ideal outcome in the $\mathcal{C} = \{\mathtt{V}\}$ case. Concretely: $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$'s decision rule under honest `P` and corrupt `V` emits `refund` whenever $\mathcal{A}$ does not produce on-chain slashing evidence within the protocol windows. The simulator $\mathcal{S}_{\mathtt{V}}$ is therefore required to produce a `refund` transcript; the probability that $\mathcal{A}$ instead drives $\mathcal{G}_{\mathsf{Ledger}}$ toward slashing (making $\mathcal{S}$'s task impossible) is exactly what Proposition 8 bounds. The four terms of the bound correspond to the four hybrid transitions of the corrupt-`V` sequence in Section 8.3.

7 The simulator

This section specifies the simulator $\mathcal{S}$ for each of the three non-trivial corruption patterns: both parties honest ($\mathcal{C} = \emptyset$, Section 7.2); prover corrupt ($\mathcal{C} = \{\mathtt{P}\}$, Section 7.3); verifier corrupt ($\mathcal{C} = \{\mathtt{V}\}$, Section 7.4). The case $\mathcal{C} = \{\mathtt{P}, \mathtt{V}\}$ requires only a translation simulator, in which $\mathcal{S}$ observes $\mathcal{A}$'s actions on $\mathcal{G}_{\mathsf{Ledger}}$ and translates them into the $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$-driving messages needed to align ledger-visible outcomes:

- On observing confirmation of $\mathsf{UTXO1}$: $\mathcal{S}$ reads x from $\mathcal{A}$'s commitment opening (via $\mathcal{G}_{\mathsf{oGG}}$ symbolic extraction, following the *Observing $\mathcal{A}$'s proofs* clause of Algorithm 10) and calls $(\mathtt{init}, \mathsf{sid}, x, D)$ on $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$ as corrupt `P`.
- On observing confirmation of $\mathsf{UTXO3}$ with contents consistent with c_1: $\mathcal{S}$ extracts w^{extr} from $\mathcal{A}$'s π_{in} (via Assumption 1) and calls $(\mathtt{prove}, \mathsf{sid}, w^{\mathsf{extr}})$.
- On observing T_1-timelock expiry with no slashing transaction: $\mathcal{S}$ does nothing; $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$'s refund timeout fires (Algorithm 1 clause IV).
- On observing a Tier-1 or Tier-2 slashing transaction: $\mathcal{S}$ calls $(\mathtt{evaluate}, \mathsf{sid})$ on $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$ and selects the matching $\mathsf{out} \in \{\mathtt{slash\text{-}fast}, \mathtt{slash\text{-}commit}\}$ per clause III of Algorithm 1.

Indistinguishability is immediate since no honest party's view is at stake. We therefore focus on the three non-trivial cases.

Throughout this section, "$\mathcal{S}$ simulates party X" means $\mathcal{S}$ produces, for each protocol message that honest X would send, a bit-string whose distribution is indistinguishable from the real-world message. Indistinguishability of the resulting transcripts is deferred to the hybrid-sequence argument of Section 8; here we specify only what $\mathcal{S}$ produces.

7.1 The simulator's interfaces and tools

Before specifying the three simulators, we catalogue the interfaces $\mathcal{S}$ has at its disposal.

$\mathcal{G}_{\mathsf{oGG}}$ **observability.** $\mathcal{S}$, domain-separated by sid, observes every group-operation query that $\mathcal{A}$ makes to $\mathcal{G}_{\mathsf{oGG}}$. In particular, $\mathcal{S}$ can determine the symbolic representation of every group element $\mathcal{A}$ computes. This underlies the extractor for π_{in} (see [[BFKT24], §4]); no outer extraction is used, R_{outer} is a deterministic predicate whose witness is public (Definition 5). We write $\mathsf{DL}_{\mathcal{G}_{\mathsf{oGG}}}(X)$ for the discrete log $\mathcal{S}$ reads off the symbolic representation of X in $\mathcal{A}$'s query transcript, a bookkeeping operation inside the simulation, not a discrete-log computation.

$\mathcal{G}_{\mathsf{oGG}}$ **simulation.** For honest-prover statements, $\mathcal{S}$ can, via the oblivious-sampling interface, produce Groth16-shaped triples (A, B, C) that verify against a given verification key and public input without knowledge of a witness. This is the $\mathcal{F}_{\mathsf{wNIZK}}$ interface exposed by $\mathcal{G}_{\mathsf{oGG}}$ [[BFKT24], Fig. 5].

$\mathcal{F}_{\mathsf{GC}}$ **simulation.** The malicious-secure garbling functionality allows $\mathcal{S}$ to produce garbled circuits whose execution behaviour can be programmed by $\mathcal{S}$, without committing to specific inputs. The `simulate` interface of Algorithm 2 is the formal handle for this capability.

Witness-hiding. The realigned protocol has no Pedersen layer; on-chain commitments are hash commitments on $\mathcal{G}_{\mathsf{Ledger}}$. Witness-hiding of $\pi_{\mathsf{in}}, \pi_{\mathsf{out}}$ is the standard Groth16 zero-knowledge surfaced as the `simulate` interface of $\mathcal{F}_{\mathsf{wNIZK}}$ (Remark 12); the simulator uses it on true statements without a witness.

$\mathcal{G}_{\mathsf{Ledger}}$ **write access.** $\mathcal{S}$ posts UTXOs and spends on the ledger on behalf of the simulated parties. $\mathcal{G}_{\mathsf{Ledger}}$ enforces that spends satisfy scripts but does not itself verify cryptographic semantics beyond what the scripts check.

$\mathcal{F}_{\mathsf{Beacon}}$ **control.** $\mathcal{S}$ plays the honest party's role in the two-party commit-reveal beacon, which gives it control over half the randomness. If $\mathcal{A}$ tries to abort, $\mathcal{S}$ invokes the default-loses path.

$\mathcal{F}_{\mathsf{HL}}^{+}$ **observation.** $\mathcal{S}$ sees all hash-lock preimages revealed on-chain.

7.2 Case 1: both parties honest ($\mathcal{C} = \emptyset$)

In this case, $\mathcal{S}$ internally simulates both `P` and `V`, driving them according to the honest protocol, but with one crucial exception: $\mathcal{S}$ does not know the witness w. It must produce a real-world transcript consistent with the ideal-world outcome announced by $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$.

Algorithm 9 Simulator $\mathcal{S}_\emptyset$ for the both-honest case ($\mathcal{C} = \emptyset$)

Input: session identifier sid and interaction with $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$.

Session start. On $(\texttt{init-leak}, \mathsf{sid}, x, D)$ from $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$:

1: mark $\texttt{P}, \texttt{V}$ as "simulated honestly"; record (x, D)
2: sample $\mathsf{seed}^{\mathcal{S}} \leftarrow \{0,1\}^\lambda$; derive only the outer trapdoors $(\tau_{\mathsf{out}}, \alpha_{\mathsf{out}}, \beta_{\mathsf{out}}, \gamma_{\mathsf{out}}, \delta_{\mathsf{out}})$, each as $H(\mathsf{seed}^{\mathcal{S}} \,\|\, \texttt{label})$
3: compute $\mathsf{SRS}^{\mathsf{out}}_{\mathsf{group}}$ (Definition 2), produce π_{SRS}; deliver $(\mathsf{SRS}^{\mathsf{out}}_{\mathsf{group}}, hs, \pi_{\mathsf{SRS}})$ from simulated $\texttt{V}$ to simulated $\texttt{P}$

Proof production. On $(\texttt{prove-leak}, \mathsf{sid}, b\,[, w])$ from $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$: ▷ w is leaked only when $b = 0$

4: **if** $b = 1$ **then** $\pi^{\mathcal{S}}_{\mathsf{in}} = (A^{\mathcal{S}}_{\mathsf{in}}, B^{\mathcal{S}}_{\mathsf{in}}, C^{\mathcal{S}}_{\mathsf{in}}) \leftarrow \mathcal{F}_{\mathsf{wNIZK}}.\texttt{simulate}(\mathsf{vk}_{\mathsf{in}}, x)$
5: **else** $\pi^{\mathcal{S}}_{\mathsf{in}} \leftarrow \mathsf{Prove}_{\mathsf{in}}(\mathsf{SRS}^{\mathsf{in}}, x, w)$ ▷ honest $\texttt{P}$'s failing artifacts, identically distributed
6: **end if**
7: the statement points are $\pi^{\mathcal{S}}_{\mathsf{in}} = (A^{\mathcal{S}}_{\mathsf{in}}, B^{\mathcal{S}}_{\mathsf{in}}, C^{\mathcal{S}}_{\mathsf{in}})$ themselves ▷ no Pedersen layer
8: $\pi^{\mathcal{S}}_{\mathsf{out}} \leftarrow \mathsf{Prove}_{\mathsf{out}}\big(\mathsf{SRS}^{\mathsf{out}}_{\mathsf{group}}, (x, A^{\mathcal{S}}_{\mathsf{in}}, B^{\mathcal{S}}_{\mathsf{in}}, C^{\mathcal{S}}_{\mathsf{in}});\ \mathsf{trace};\ 0\big)$ ▷ honest outer prover; the witness is public; verifies iff $b = 1$

Commitments and circuits.

9: $\{\widetilde{C}^{\mathcal{S}}_i\}_{i=1}^{s} \leftarrow \mathcal{F}_{\mathsf{GC}}.\texttt{simulate}$ programmed to output the label matching b (Algorithm 2, IV)
10: sample $k^{\mathcal{S}} \leftarrow \{0,1\}^{2\lambda}$; set $h^{\mathcal{S}} := H(k^{\mathcal{S}})$; build capsules $u^{\mathcal{S}}_i$ as in Algorithm 6
11: sample $\rho_1 \leftarrow \{0,1\}^\lambda$; compute $c^{\mathcal{S}}_1$; post $\mathsf{UTXO1}$ on $\mathcal{G}_{\mathsf{Ledger}}$ with deposit D

Phases II–III.

12: drive simulated $\texttt{P}, \texttt{V}$ through Phase II ($\mathcal{F}_{\mathsf{Beacon}}$, cut-and-choose, seed reveal, $\mathsf{UTXO3}$) ▷ all checks pass by construction
13: **if** $b = 1$ **then** surviving circuits evaluate to *accept*; no k released; after T_1, $\mathsf{UTXO1}$ refunds to $\texttt{P}$
14: **else** surviving circuits evaluate to *reject*; simulated $\texttt{V}$ reconstructs $k^{\mathcal{S}}$ and posts the $\texttt{slash-fast}$ spend
15: **end if**

Consistency with $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$.

16: on simulated $\texttt{V}$'s evaluation, send $(\texttt{evaluate}, \mathsf{sid})$ to $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$ ▷ honest $\texttt{P}$ called $\texttt{prove}$ directly; $\mathcal{S}$ never sees w
17: $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$ announces out ($\texttt{refund}$ iff $b = 1$) on $\mathcal{G}_{\mathsf{Ledger}}$, matching $\mathcal{S}$'s on-chain trace

Remark 28. With both parties honest, there is no adversarial strategy $\mathcal{S}$ needs to counter; indistinguishability reduces to the simulation qualities of $\mathcal{G}_{\mathsf{oGG}}$ and $\mathcal{F}_{\mathsf{GC}}$. On $(\texttt{prove-leak}, \mathsf{sid}, b\,[, w])$, $\mathcal{S}$ produces $\pi^{\mathcal{S}}_{\mathsf{in}}$ via $\mathcal{F}_{\mathsf{wNIZK}}.\texttt{simulate}$ when $b = 1$ and via the honest prover on the leaked w when $b = 0$ (Remark 30 applies verbatim); $\pi^{\mathcal{S}}_{\mathsf{out}}$ is computed by the honest outer prover on the public trace in both branches; the garbled circuits are produced via $\mathcal{F}_{\mathsf{GC}}$ programmed to the bit b. On-chain commitments are hash commitments and require no equivocation in the both-honest case.

7.3 Case 2: prover corrupt ($\mathcal{C} = \{\texttt{P}\}$)

This is the soundness case. $\mathcal{S}$ plays as honest $\texttt{V}$ and observes $\mathcal{A}$'s behaviour as corrupt $\texttt{P}$. The critical task is *extraction*: $\mathcal{S}$ must recover the witness w from $\mathcal{A}$'s protocol messages and forward it to $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$ via $\texttt{prove}$.

Algorithm 10 Simulator $\mathcal{S}_{\mathtt{P}}$ for the corrupt-prover case ($\mathcal{C} = \{\mathtt{P}\}$)

Session start. On $(\texttt{init-leak}, \mathsf{sid}, x, D)$ from $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$:

play as honest V: sample $\mathsf{seed} \leftarrow \{0,1\}^{\lambda}$; compute $\mathsf{SRS}^{\mathsf{out}}_{\mathsf{group}}$, π_{SRS}; send to $\mathcal{A}$

Observe and extract. ▷ $\mathcal{A}$ acts as corrupt P

record every $\mathcal{A}$-initiated $\mathcal{G}_{\mathsf{oGG}}$ query (domain-separated by sid)

when $\mathcal{A}$ opens UTXO3 consistently with c_1, revealing $\pi^{\mathcal{A}}_{\mathsf{in}} = (A_{\mathsf{in}}, B_{\mathsf{in}}, C_{\mathsf{in}})$ and $\pi^{\mathcal{A}}_{\mathsf{out}}$:

  if the committed $\pi^{\mathcal{A}}_{\mathsf{out}}$ verifies under $\mathsf{vk}_{\mathsf{outer}}(\mathsf{seed})$ on $(x, A_{\mathsf{in}}, B_{\mathsf{in}}, C_{\mathsf{in}})$, then $R_{\mathsf{outer}}(\mathsf{stmt}) = 1$ except with the soundness error of Lemma 3, i.e. $(A_{\mathsf{in}}, B_{\mathsf{in}}, C_{\mathsf{in}})$ is a valid inner proof under $\mathsf{vk}_{\mathsf{in}}$ (Definition 5); *no outer extraction is run*

  $w^{\mathsf{extr}} \leftarrow \mathcal{E}_{\mathsf{in}}(A_{\mathsf{in}}, B_{\mathsf{in}}, C_{\mathsf{in}})$ ▷ the `extract` interface of $\mathcal{F}^{C_{\mathsf{assertions}}}_{\mathsf{wNIZK}}$

send $(\texttt{prove}, \mathsf{sid}, w^{\mathsf{extr}})$ to $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$ ▷ $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$ sets $b := C_{\mathsf{assertions}}(x; w^{\mathsf{extr}})$

Phase II consistency checks. ▷ $\mathcal{S}$ plays honest V

engage in $\mathcal{F}_{\mathsf{Beacon}}$, producing J uniformly

for each circuit $i \in J$ **do**

  if the opening of i fails **then**

    send $(\texttt{evaluate}, \mathsf{sid})$ to $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$; select $\mathsf{out} \leftarrow \texttt{slash-commit}$ (clause III of Algorithm 1)

  end if

end for

reveal seed cooperatively; observe $\mathcal{A}$'s UTXO3

if c_1-consistency fails on the opened package, or a signature fails or equivocates, or a class-(B) wrong-bit reveal or lost $\mathcal{F}_{\mathsf{derive}}$ dispute occurs **then** $\mathsf{out} \leftarrow \texttt{slash-commit}$

end if

Phase III evaluation.

evaluate surviving circuits with canonical labels and $\mathcal{A}$'s signed scalars

if $\pi^{\mathcal{A}}_{\mathsf{out}}$ does not verify R_{outer} on statement $(x, A_{\mathsf{in}}, B_{\mathsf{in}}, C_{\mathsf{in}})$ **then** ▷ Corollary 1, contrapositive

  reject label produced; reconstruct k; post `slash-fast` transaction on $\mathcal{G}_{\mathsf{Ledger}}$

end if

send $(\texttt{evaluate}, \mathsf{sid})$ to $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$ ▷ $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$'s b matches the real outcome:

if $b = 1$ **then**

  $\mathsf{out} = \texttt{refund}$

else if π_{in} invalid, on-chain commitment-consistent, GC detects **then**

  $\mathsf{out} = \texttt{slash-fast}$

else $\mathsf{out} = \texttt{slash-commit}$ ▷ commitment, signature, canonical-bit, or $\mathcal{F}_{\mathsf{derive}}$ evidence on UTXO3 (Lemma 5)

end if

Remark 29. The extraction must occur *before* $\mathcal{S}$ sends $(\texttt{evaluate}, \mathsf{sid})$ to $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$, so that $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$'s state is `witnessed` when evaluation is triggered. Because $\pi^{\mathcal{A}}_{\mathsf{in}}$ is revealed in UTXO3 (the simulator sees this via $\mathcal{G}_{\mathsf{Ledger}}$), and the subsequent on-chain evaluation steps are gated behind UTXO3's posting, this sequencing is naturally satisfied. An edge case arises if $\mathcal{A}$ never posts UTXO3: then $\mathcal{S}$ submits nothing to $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$ and the ideal-world refund timeout fires, which matches the real-world outcome (honest V reclaims after T_1 elapses).

7.4 Case 3: verifier corrupt ($\mathcal{C} = \{\mathtt{V}\}$)

Here $\mathcal{S}$ plays as honest P and interacts with corrupt V, controlled by $\mathcal{A}$. The main challenges are: (a) $\mathcal{S}$ does not know w and must simulate π_{in} and the garbled circuits (the outer proof is honestly computable from the public trace); (b) $\mathcal{S}$ must correctly produce or withhold the slashing secret k based on $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$'s b.

Algorithm 11 Simulator $\mathcal{S}_{\mathtt{V}}$ for the corrupt-verifier case ($\mathcal{C} = \{\mathtt{V}\}$)

Session start. On $(\mathtt{init\text{-}leak}, \mathsf{sid}, x, D)$ from $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$:
wait for $\mathcal{A}$ (corrupt V) to send $\mathsf{SRS}^{\mathsf{out}}_{\mathsf{group}}$, hs, π_{SRS}
SRS verification.
if π_{SRS} is invalid **then** abort
end if
Producing π_{in}, π_{out}. On $(\mathtt{prove\text{-}leak}, \mathsf{sid}, b\,[, w])$ from $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$:
if $b = 1$ **then** $\pi^{\mathcal{S}}_{\mathsf{in}} \leftarrow \mathcal{F}_{\mathsf{wNIZK}}.\mathtt{simulate}(\mathsf{vk}_{\mathsf{in}}, x)$ ▷ no witness; Groth16 ZK
else $\pi^{\mathcal{S}}_{\mathsf{in}} \leftarrow \mathsf{Prove}_{\mathsf{in}}(\mathsf{SRS}^{\mathsf{in}}, x, w)$ ▷ w leaked in the $b = 0$ branch only; non-verifying, as honest P's
end if
let $\mathsf{stmt}^{\mathcal{S}} := (x, A^{\mathcal{S}}_{\mathsf{in}}, B^{\mathcal{S}}_{\mathsf{in}}, C^{\mathcal{S}}_{\mathsf{in}})$ with $(A^{\mathcal{S}}_{\mathsf{in}}, B^{\mathcal{S}}_{\mathsf{in}}, C^{\mathcal{S}}_{\mathsf{in}}) = \pi^{\mathcal{S}}_{\mathsf{in}}$
$\pi^{\mathcal{S}}_{\mathsf{out}} \leftarrow \mathsf{Prove}_{\mathsf{out}}\big(\mathsf{SRS}^{\mathsf{out}}_{\mathsf{group}}, \mathsf{stmt}^{\mathcal{S}};\ \mathsf{trace}(\mathsf{stmt}^{\mathcal{S}});\ 0\big)$ ▷ honest outer prover in *both* branches: the witness is the public trace (Definition 5); verifies iff $b = 1$; no trapdoor, no simulation, no discrete logarithm
Garbled-circuit programming.
if $b = 1$ **then** $\mathcal{F}_{\mathsf{GC}}.\mathtt{simulate}$ programmed to output *accept*
else $\mathcal{F}_{\mathsf{GC}}.\mathtt{simulate}$ programmed to output *reject*
end if
build capsules $u^{\mathcal{S}}_i$ and slashing secret $k^{\mathcal{S}}$ as in the real protocol
Phase II on-chain.
post UTXO1 with deposit D and $c^{\mathcal{S}}_1$ binding $\mathsf{stmt}^{\mathcal{S}}$, $(A^{\mathcal{S}}_{\mathsf{out}}, B^{\mathcal{S}}_{\mathsf{out}}, C^{\mathcal{S}}_{\mathsf{out}})$, $h^{\mathcal{S}} = H(k^{\mathcal{S}})$, $\mathsf{root}^{\mathcal{S}}_{\mathsf{GC}}$, ρ_1
$\mathcal{A}$ posts UTXO2; participate in $\mathcal{F}_{\mathsf{Beacon}}$ and cut-and-choose; for $i \in J$ open simulated garbling ▷ $\mathcal{F}_{\mathsf{GC}}$ malicious-secure simulation
$\mathcal{A}$ reveals seed and $H(\mathsf{seed}) = hs$ is verified, or the T_2 forfeiture path of Lemma 6 fires
Post-reveal package. ▷ every step below is a public computation
$\mathcal{S}_{\mathtt{V}}$: run honest P's seed-expansion check on the revealed seed against the published $\mathsf{SRS}^{\mathrm{group}}_{\mathrm{out}}$ (Algorithm 7)
if the check fails **then**
    emulate the honest halt: publish no UTXO3, forward no $\mathtt{evaluate}$ to $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$; the T_1 timeout (clause IV) settles $\mathtt{refund}$ ▷ matches Lemma 4; independent of b
else
    compute $(a_{\mathsf{out}}, b_{\mathsf{out}}, c_{\mathsf{out}})$ from $\mathcal{S}$'s outer assignment and the trapdoors, and T_{ic} from $(\mathsf{seed}, \mathsf{stmt}^{\mathcal{S}})$; sign via $\mathcal{F}^{+}_{\mathsf{HL}}$, reveal class-(B) canonical labels, publish UTXO3
end if
Phase III. On $(\mathtt{evaluate}, \mathsf{sid})$ from $\mathcal{A}$ (forwarded to $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$), $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$ returns out and (privately to V) b:
if $b = 1$ **then** do nothing; circuits evaluate to accept; after T_1, $\mathcal{G}_{\mathsf{Ledger}}$ refunds D to P
else circuits evaluate to reject; $\mathcal{A}$ reconstructs $k^{\mathcal{S}}$ and spends UTXO1 via the hashlock path; $\mathcal{S}$ does not interfere
end if

Remark 30. The corrupt-V simulator splits on the bit b that $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$ discloses, but only for the *inner* proof. **True statement ($b = 1$).** Witness-hiding is the only obligation

and concerns only π_{in}: $\mathcal{S}$ produces $\pi_{\mathsf{in}}^{\mathcal{S}}$ through $\mathcal{F}_{\mathsf{wNIZK}}.\mathtt{simulate}$ (Groth16 zero-knowledge against the public $\mathsf{vk}_{\mathsf{in}}$), which is exactly the simulation certifying that a corrupt V learns nothing about w. **False statement ($b = 0$).** Here $\mathcal{Z}$ has handed honest P an invalid w; an invalid assignment carries no witness to hide, so $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$ leaks w to $\mathcal{S}$ (clause II), and $\mathcal{S}$ reproduces honest P's actual non-verifying $\pi_{\mathsf{in}}^{\mathcal{S}} \leftarrow \mathsf{Prove}_{\mathsf{in}}(\mathsf{SRS}^{\mathsf{in}}, x, w)$, whose distribution matches honest-P-with-invalid-w exactly. **The outer proof needs no simulation in either branch:** its witness is the public verification trace of $\mathsf{stmt}^{\mathcal{S}}$, so $\mathcal{S}$ runs the honest outer prover $\mathsf{Prove}_{\mathsf{out}}(\mathsf{SRS}^{\mathsf{out}}_{\mathsf{group}}, \mathsf{stmt}^{\mathcal{S}}; \mathsf{trace}(\mathsf{stmt}^{\mathcal{S}}); 0)$, obtaining a $\pi_{\mathsf{out}}^{\mathcal{S}}$ that verifies iff $b = 1$ and is identically distributed to honest P's (the proof is deterministic). The garbled-circuit-programming step routes the $b = 0$ case to a reject output, matching the real protocol. In neither branch does $\mathcal{S}$ leak w to $\mathcal{Z}$.

Note that this works because the private message $(\mathtt{acceptance\text{-}bit}, \mathsf{sid}, b)$ from $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$ to corrupted V is delivered *before* the on-chain settlement, giving $\mathcal{S}$ time to align the garbled-circuit programming.

When V supplies a non-expansion SRS, the simulator's guard fails exactly as honest P's would, and it suppresses the `evaluate` call; $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$ then settles `refund` by the clause IV timeout, matching the real-world halt regardless of b. Thus the $b = 0$ leakage branch and the guard-failure branch do not interfere: the former routes to `slash-fast` only when P does publish UTXO3 (guard passed), the latter to `refund` when she does not.

Remark 31. In the hybrid design the simulator needs the outer trapdoors only *after* the seed is public, at which point they are derivable by anyone as $H(\mathsf{seed} \,\|\, \mathtt{label})$; no pre-reveal trapdoor extraction from $\mathcal{A}$'s $\mathcal{G}_{\mathsf{oGG}}$ queries is used anywhere in Algorithm 11. π_{SRS} guarantees knowledge of a seed behind hs that binds the committed scalar vector; the Phase-I batched pairing checks establish *internal consistency only* and are independent of seed (cf. the halt discussion in the proof of Lemma 3). The link between the published points and the seed is established only post-reveal, by the seed-expansion check of Algorithm 7: $\mathcal{S}$ runs it exactly as honest P would and, on mismatch, publishes no UTXO3, routing the session to the timelock-T_1 `refund` (Lemma 4), which matches $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$'s clause III/IV outcome for an honest prover. On a π_{SRS} or internal-consistency failure in Phase I, $\mathcal{S}$ aborts exactly as honest P would. The adversarial-seed-generation concern of earlier versions (an $\mathcal{A}$ deriving its SRS outside $\mathcal{G}_{\mathsf{oGG}}$'s observed operations) is therefore moot at this layer: observability is no longer load-bearing for the corrupt-verifier simulation.

7.5 Summary of proof obligations

Table 1: Tools used by the simulator in each corruption pattern.

Corruption	Main simulator tools used
$\mathcal{C} = \emptyset$	$(b = 1)$ $\mathcal{G}_{\mathsf{oGG}}$ simulation interface for $\pi_{\mathsf{in}}^{\mathcal{S}}$ / $(b = 0)$ honest prover on the leaked w; honest outer prover on the public trace; $\mathcal{F}_{\mathsf{GC}}$ garbling simulator; hash commitments.
$\mathcal{C} = \{\mathtt{P}\}$	$\mathcal{G}_{\mathsf{oGG}}$ observability and extraction [[BFKT24], §4]; c_1 binding via collision-resistance (for pre-reveal commitments); hash-lock binding (for canonical labels).
$\mathcal{C} = \{\mathtt{V}\}$	$(b = 1)$ $\mathcal{G}_{\mathsf{oGG}}$ simulation interface for $\pi_{\mathsf{in}}^{\mathcal{S}}$ (no witness); $(b = 0)$ honest prover on the leaked invalid w; honest outer prover on the public trace in both branches; $\mathcal{F}_{\mathsf{GC}}$ malicious-secure simulation (for opened cut-and-choose circuits); hash commitments.

Having specified the simulator for each corruption pattern, the proof obligations remaining for the main theorem (Theorem 1) are:

1. Show that in each corruption pattern, the real-world execution $\mathsf{EXEC}_{\Pi_{\mathsf{DV\text{-}Groth16}},\mathcal{A},\mathcal{Z}}$ is computationally indistinguishable from the ideal-world execution $\mathsf{EXEC}_{\mathcal{F}_{\mathsf{DV\text{-}Groth16}},\mathcal{S},\mathcal{Z}}$. This is done by a hybrid-sequence argument, one sequence per corruption pattern.
2. In the $\mathcal{C} = \{\mathtt{P}\}$ case, formally bound the probability that the extracted w^{extr} does not satisfy $C_{\mathsf{assertions}}(x; w^{\mathsf{extr}}) = 1$ when $\pi_{\mathsf{in}}^{\mathcal{A}}$ verifies. This is the content of [[BFKT24], Thm. 1] applied to the inner Groth16 circuit.
3. In the $\mathcal{C} = \{\mathtt{V}\}$ case, formally argue that the simulator's proofs, $\mathcal{F}_{\mathsf{wNIZK}}.\mathtt{simulate}$ for $\pi_{\mathsf{in}}^{\mathcal{S}}$ at $b = 1$, the honest prover on the leaked w at $b = 0$, and the honest outer prover on the public trace in both branches, are distributionally indistinguishable from a real-world honest-P trace.

8 Security proof

This section proves Theorem 1 by exhibiting, for each corruption pattern $\mathcal{C} \in \{\emptyset, \{\mathtt{P}\}, \{\mathtt{V}\}\}$, a sequence of hybrid executions $\mathsf{H}_0^{\mathcal{C}}, \mathsf{H}_1^{\mathcal{C}}, \ldots, \mathsf{H}_{n_{\mathcal{C}}}^{\mathcal{C}}$ such that $\mathsf{H}_0^{\mathcal{C}} = \mathsf{EXEC}_{\Pi_{\mathsf{DV\text{-}Groth16}},\mathcal{A},\mathcal{Z}}$ is the real-world execution and $\mathsf{H}_{n_{\mathcal{C}}}^{\mathcal{C}} = \mathsf{EXEC}_{\mathcal{F}_{\mathsf{DV\text{-}Groth16}},\mathcal{S}_{\mathcal{C}},\mathcal{Z}}$ is the ideal-world execution with the simulator from Section 7. Adjacent hybrids differ in exactly one cryptographic component (or, in two presentational compressions noted below, three components which are individually independent), and the distinguishing advantage of $\mathcal{Z}$ between adjacent hybrids is bounded by one or more named primitives' security.

The case $\mathcal{C} = \{\mathtt{P}, \mathtt{V}\}$ (both parties corrupt) is handled by the translation simulator described in Section 7: $\mathcal{S}$ maps $\mathcal{A}$'s on-chain actions into the corresponding $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$-driving messages, so that the ledger-visible outcome announced by $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$ matches the real-world transaction shape. Indistinguishability is immediate: no honest party's view is at stake, and the translation is deterministic in $\mathcal{A}$'s actions.

8.1 The corrupt-prover hybrid sequence

The corrupt-prover case is the most technically substantial of the three. We replace the real cryptographic components one at a time, ending in the ideal world where $\mathcal{S}_{\mathtt{P}}$ from Algorithm 10 drives $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$. The hybrid sequence is:

- $\mathsf{H}_0^{\{\mathtt{P}\}}$: the real-world execution. Honest V runs the real protocol against $\mathcal{A}$-as-P. All hash-locks, beacons, commitments, garbled circuits, and Groth16 proofs are real.
- $\mathsf{H}_1^{\{\mathtt{P}\}}$: replace the real hash-lock implementation (Lamport or Winternitz) with the ideal $\mathcal{F}_{\mathsf{HL}}^{+}$ (Algorithm 4).
- $\mathsf{H}_2^{\{\mathtt{P}\}}$: replace the real beacon (commit-reveal) with $\mathcal{F}_{\mathsf{Beacon}}$ (Algorithm 5).
- $\mathsf{H}_3^{\{\mathtt{P}\}}$: (vacuous in the realigned design: there is no Pedersen layer. Retained only to keep the hybrid indexing aligned with the corrupt-verifier sequence; the binding of on-chain commitments is collision-resistance of H, charged at Lemma 9.)
- $\mathsf{H}_4^{\{\mathtt{P}\}}$: replace the real garbled circuits with $\mathcal{F}_{\mathsf{GC}}$ (Algorithm 2).
- $\mathsf{H}_5^{\{\mathtt{P}\}}$: replace the real Groth16 verification of $\mathcal{A}$'s π_{in} with calls to $\mathcal{F}_{\mathsf{wNIZK}}^{C_{\mathsf{assertions}}}$ over $\mathcal{G}_{\mathsf{oGG}}$; the outer proof continues to be verified concretely, its soundness charged at the bridge (Lemma 14). This is the key transition: honest V now obtains w^{extr} from $\mathcal{F}_{\mathsf{wNIZK}}$'s `extract` interface applied to the symbolic representation of $\mathcal{A}$'s queries.

- $\mathsf{H}_6^{\{\mathtt{P}\}}$: replace the entire honest-$\mathtt{V}$ simulation (driven by extracted scalars and observed transcript) with $\mathcal{S}_{\mathtt{P}}$ driving $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$. The on-chain settlement is now announced by $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$ on the global ledger rather than by $\mathtt{V}$.
- $\mathsf{H}_7^{\{\mathtt{P}\}} = \mathsf{EXEC}_{\mathcal{F}_{\mathsf{DV\text{-}Groth16}},\mathcal{S}_{\mathtt{P}},\mathcal{Z}}$: the ideal world.

We bound each transition in turn.

Lemma 9 ($\mathsf{H}_0 \to \mathsf{H}_1$: hash-lock)**.** $\left|\Pr[\mathcal{Z}=1 \mid \mathsf{H}_0] - \Pr[\mathcal{Z}=1 \mid \mathsf{H}_1]\right| \le \mathsf{Adv}^{\mathsf{HL}+}(\lambda)$, *where* $\mathsf{Adv}^{\mathsf{HL}+}$ *is the advantage against the one-wayness of the hash function* H *underlying the chosen hash-lock mechanism.*

Proof. **Reduction $\mathcal{B}_{\mathsf{HL}}$.** On input a single hash-lock target $h^* = H(\mathtt{cl} \,\|\, i^* \,\|\, j^* \,\|\, b^* \,\|\, \lambda^*)$ from the one-wayness challenger (with hidden λ^*), $\mathcal{B}_{\mathsf{HL}}$ simulates the protocol to $\mathcal{Z}$ as follows:

1. Sample a random target slot $(i^*, j^*, b^*) \in [s] \times [|\mathsf{wires}|] \times \{0,1\}$ to embed.
2. Sample all other hash-lock targets honestly per the real protocol; for slot (i^*, j^*, b^*), plant h^* in place of $H(\mathtt{cl} \,\|\, i^* \,\|\, j^* \,\|\, b^* \,\|\, \lambda_{i^*,j^*,b^*})$.
3. Run the full protocol with $\mathcal{Z}$ playing the corrupt prover role. On any of $\mathcal{Z}$'s opening attempts for slot (i^*, j^*, b^*) that submit a preimage $\lambda' \neq \lambda^*$ with

$$H(\mathtt{cl} \,\|\, i^* \,\|\, j^* \,\|\, b^* \,\|\, \lambda') = h^*,$$

   output λ' as the preimage and halt.

Analysis. The number of hash-lock slots is $N = 2 \cdot s \cdot |\mathsf{wires}|$. If $\mathcal{Z}$'s distinguishing advantage is Δ and is achieved by opening some specific slot, thenconditioning on $\mathcal{B}_{\mathsf{HL}}$'s slot guess being correct (probability $1/N$), $\mathcal{B}_{\mathsf{HL}}$ wins with probability at least Δ/N. Hence $\Delta \le N \cdot \mathsf{Adv}_H^{\mathsf{HL}+}(\lambda)$, where $\mathsf{Adv}_H^{\mathsf{HL}+}$ is the per-instance one-wayness advantageof H. Folding the multiplicative slot factor N into the notation $\mathsf{Adv}^{\mathsf{HL}+}(\lambda) := N \cdot \mathsf{Adv}_H^{\mathsf{HL}+}(\lambda)$ recovers the stated form. (Equivalently: under the standard $H : \{0,1\}^* \to \{0,1\}^\ell$ instantiation with $\ell = 2\lambda$, $\mathsf{Adv}_H^{\mathsf{HL}+} \le 2^{-2\lambda}$ and $N \cdot 2^{-2\lambda}$ remains $\mathsf{negl}(\lambda)$.) Interfaces IV–V of $\mathcal{F}_{\mathsf{HL}}^{+}$ (equivocation slashing, canonical-bit adjudication) are deterministic public checks given the on-chain data; their realisation advantage is the same $\mathsf{Adv}^{\mathsf{HL}+}$. □

Lemma 10 ($\mathsf{H}_1 \to \mathsf{H}_2$: beacon)**.** $\left|\Pr[\mathcal{Z}=1 \mid \mathsf{H}_1] - \Pr[\mathcal{Z}=1 \mid \mathsf{H}_2]\right| \le \varepsilon_{\mathcal{R}}$, *where* $\varepsilon_{\mathcal{R}}$ *is the bias of the commit-reveal beacon, itself bounded by* $\mathsf{Adv}_H^{\mathsf{CR}}$ *under collision-resistant commitments (Section 5.7).*

Proof. **Reduction $\mathcal{B}_{\mathcal{R}}$.** Given a PPT environment $\mathcal{Z}$ distinguishing H_1 from H_2 with advantage Δ, construct $\mathcal{B}_{\mathcal{R}}$ playing as one party in the two-party commit-reveal beacon $\mathcal{R}$ (with $\mathcal{Z}$ as the other party):

1. $\mathcal{B}_{\mathcal{R}}$ embeds the beacon execution into the protocol's Phase-II step.
2. $\mathcal{B}_{\mathcal{R}}$ runs the rest of the protocol honestly with $\mathcal{Z}$, using the beacon's output J^* wherever H_1 uses a real J.
3. At the end, $\mathcal{B}_{\mathcal{R}}$ outputs $\mathcal{Z}$'s decision bit.

Analysis. If $\mathcal{R}$ behaves like an honest beacon (its output is uniform, modulo bias $\varepsilon_{\mathcal{R}}$), $\mathcal{B}_{\mathcal{R}}$'s simulation is statistically close to H_1. If $\mathcal{R}$ is replaced by an idealised uniform sampler (a $\mathcal{F}_{\mathsf{Beacon}}$-style abstraction), the simulation is exactly H_2. By the definition of beacon bias, the statistical distance between $\mathcal{R}$'s output distribution and uniform is at most $\varepsilon_{\mathcal{R}}$ per beacon call; with one beacon call per session, $|\Pr[\mathcal{Z}=1 \mid \mathsf{H}_1] - \Pr[\mathcal{Z}=1 \mid \mathsf{H}_2]| \le \varepsilon_{\mathcal{R}}$. The

two specific deviation modes a corrupt participant in $\mathcal{R}$ might attempt are: (a) commit-collision (find two openings of the commit phase hashing to the same digest, biasing J^*); (b) selective abort (refuse to open the commit when the resulting J^* is unfavourable, forcing the default-loses path). Both are folded into $\varepsilon_{\mathcal{R}}$ by the beacon's construction: (a) is bounded by $\mathsf{Adv}^{\mathsf{CR}}_{H}$, (b) by the default-loses probability $2^{-\Omega(\lambda)}$. The beacon's instantiation in Section 5.7 satisfies $\varepsilon_{\mathcal{R}} \le \mathsf{Adv}^{\mathsf{CR}}_{H} + 2^{-\Omega(\lambda)}$. □

Lemma 11 ($\mathsf{H}_2 \to \mathsf{H}_3$: vacuous)**.** $\left|\Pr[\mathcal{Z}=1 \mid \mathsf{H}_2] - \Pr[\mathcal{Z}=1 \mid \mathsf{H}_3]\right| \le 0$.

Proof. In the realigned protocol there is no Pedersen layer: the on-chain commitments are hash commitments, already idealised at the $\mathcal{F}^{+}_{\mathsf{HL}}/c_1$ steps, and the cross-circuit binding of inputs is provided by c_1 (collision-resistance of H, charged at Lemma 9). This transition is therefore vacuous and is retained only to keep the hybrid index aligned with the corrupt-verifier sequence. □

Remark 32. The bound contains one extraction term and one soundness term, for two genuinely distinct jobs. $\mathsf{Adv}^{\mathsf{sound},\mathcal{G}_{\mathsf{oGG}}}_{\mathsf{out}}$ (charged at Lemma 14 via Lemma 3) bounds a corrupt prover committing a verifying π_{out} on a false statement, plain soundness for $\mathcal{C}_{\mathsf{verif}}$, since its witness is the public verification trace and no outer extraction exists. $\mathsf{Adv}^{\mathsf{Groth16}/\mathcal{G}_{\mathsf{oGG}}}_{\mathsf{in}}$ (charged at Lemma 13) bounds the recovery of the assertion witness from $(A_{\mathsf{in}}, B_{\mathsf{in}}, C_{\mathsf{in}})$ via $\mathcal{F}^{C_{\mathsf{assertions}}}_{\mathsf{wNIZK}}$'s `extract`. Neither subsumes the other; there is no shared term to credit once.

Lemma 12 ($\mathsf{H}_3 \to \mathsf{H}_4$: garbled circuits)**.** $\left|\Pr[\mathcal{Z}=1 \mid \mathsf{H}_3] - \Pr[\mathcal{Z}=1 \mid \mathsf{H}_4]\right| \le \mathsf{Adv}^{\mathsf{garb}}_{\mathsf{BHR}}(\lambda)$.

Proof. H_3 uses the real BHR garbling scheme; H_4 replaces it with $\mathcal{F}_{\mathsf{GC}}$. The cut-and-choose soundness factor $\binom{s}{s/2}^{-1}$, the probability that a corrupt garbler passes all openings in J but leaves at least one malformed circuit in $[s] \setminus J$, is by Assumption 3 an internal parameter of $\mathcal{F}_{\mathsf{GC}}$'s ideal behaviour. It is accounted for at the hybrid step where cut-and-choose soundness is actually invoked, namely Lemma 14 below. The hybrid distinguishing advantage between the real BHR scheme and $\mathcal{F}_{\mathsf{GC}}$ at this step is therefore bounded only by the BHR privacy/authenticity loss $\mathsf{Adv}^{\mathsf{garb}}_{\mathsf{BHR}}$. For reasonable instantiations (free-XOR with AES-based hash), this is $\le 2^{-\lambda}$. □

Lemma 13 ($\mathsf{H}_4 \to \mathsf{H}_5$: Groth16 over $\mathcal{G}_{\mathsf{oGG}}$)**.**

$$\left|\Pr[\mathcal{Z}=1 \mid \mathsf{H}_4] - \Pr[\mathcal{Z}=1 \mid \mathsf{H}_5]\right| \le \mathsf{Adv}^{\mathsf{Groth16}/\mathcal{G}_{\mathsf{oGG}}}(\lambda, q, |C_{\mathsf{assertions}}|).$$

Proof. In H_4, honest $\mathtt{V}$ verifies $\mathcal{A}$'s π_{in} by running the standard Groth16 verification algorithm. In H_5, $\mathtt{V}$ instead invokes $\mathcal{F}^{C_{\mathsf{assertions}}}_{\mathsf{wNIZK}}$'s `verify` interface and additionally invokes the `extract` interface to recover w^{extr} from $\mathcal{A}$'s π_{in}. By [[BFKT24], Thm. 1], this replacement costs at most $\mathsf{Adv}^{\mathsf{Groth16}/\mathcal{G}_{\mathsf{oGG}}}(\lambda, q, |C_{\mathsf{assertions}}|)$.

The *outer* proof is not replaced at this transition: π_{out} continues to be verified concretely against $\mathsf{vk}_{\mathsf{outer}}(\mathsf{seed})$, and its soundness, the only outer property the analysis uses (Definition 5), is charged once, at the bridge transition, through Lemma 3's term $\mathsf{Adv}^{\mathsf{sound},\mathcal{G}_{\mathsf{oGG}}}_{\mathsf{out}}$.

In H_5, with overwhelming probability, the extracted w^{extr} either satisfies $C_{\mathsf{assertions}}(x; w^{\mathsf{extr}}) = 1$ when $\pi^{\mathcal{A}}_{\mathsf{in}}$ verifies, or is $\perp$ when $\pi^{\mathcal{A}}_{\mathsf{in}}$ does not verify. The extraction failure probability is part of the per-circuit advantage bound from [[BFKT24], Thm. 1]. □

Lemma 14 ($\mathsf{H}_5 \to \mathsf{H}_6$: the bridge)**.** *Under collision-resistance of H, $\mathcal{F}_{\mathsf{GC}}$'s cut-and-choose soundness, zero-knowledge of π_{SRS} (Assumption 2), and the realisation guarantees of $\mathcal{F}_{\mathsf{HL}}^{+}$ and $\mathcal{F}_{\mathsf{derive}}$,*

$$\begin{aligned}\left|\Pr[\mathcal{Z}=1 \mid \mathsf{H}_5] - \Pr[\mathcal{Z}=1 \mid \mathsf{H}_6]\right| \le \mathsf{Adv}_{\mathsf{out}}^{\mathsf{sound},\mathcal{G}_{\mathsf{oGG}}} + \mathsf{Adv}_{H}^{\mathsf{CR}}(\lambda) \\ + \binom{s}{s/2}^{-1} + \mathsf{Adv}_{\pi_{\mathsf{SRS}}}^{\mathsf{ZK}}(\lambda) + \mathsf{Adv}^{\mathsf{derive}}.\end{aligned}$$

Proof. In H_5 the garbling and the inner Groth16 machinery have been replaced by their ideal counterparts. It remains to establish the bridge: whenever the real protocol emits $\mathsf{out}^{\mathsf{real}} \in \{\texttt{refund}, \texttt{slash-fast}, \texttt{slash-commit}\}$ under corrupt $\mathcal{A}$, the ideal world H_6 emits the matching outcome with $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$ invoked on the correct extracted witness. We trace the conditions for $\mathsf{out}^{\mathsf{real}} = \texttt{refund}$; the two slashing outcomes are handled in Lemma 15.

(i) Commitment binding. $\mathcal{A}$'s Phase-III openings must hash to c_1; two distinct openings yield a collision (Lemma 1), probability at most $\mathsf{Adv}_{H}^{\mathsf{CR}}$, union-bounded with the per-circuit hash-lock collisions of (iii). Conditioning on the complement, the opened Phase-I tuple, including the proof points $(A_{\mathsf{in}}, B_{\mathsf{in}}, C_{\mathsf{in}})$ and $(A_{\mathsf{out}}, B_{\mathsf{out}}, C_{\mathsf{out}})$, is unique.

(ii) Outer soundness. Conditioning on the `refund` path, the committed π_{out} verifies under $\mathsf{vk}_{\mathsf{outer}}(\mathsf{seed})$ on the committed stmt. By Lemma 3 (the pre-reveal cutoff: the points were fixed in c_1 before the seed moved), $R_{\mathsf{outer}}(\mathsf{stmt}) = 1$ except with probability $\mathsf{Adv}_{\mathsf{out}}^{\mathsf{sound},\mathcal{G}_{\mathsf{oGG}}} + \mathsf{Adv}_{\pi_{\mathsf{SRS}}}^{\mathsf{ZK}}$, and by Definition 5 this *is* the statement that $(A_{\mathsf{in}}, B_{\mathsf{in}}, C_{\mathsf{in}})$ verifies under $\mathsf{vk}_{\mathsf{in}}$. No bridge theorem is invoked; the step is definitional.

(iii) Surviving-circuit acceptance decides outer verification. Conditioning on (i)–(ii) and on $\mathcal{F}_{\mathsf{GC}}$'s correctness-of-survivors guarantee (a malformed survivor slips through with probability at most $\binom{s}{s/2}^{-1}$, Assumption 3), Corollary 1 applies: the class-(B) canonical labels are the genuine seed-derived values (Lemma 2; wrong-bit reveals are caught by the deterministic public adjudication of $\mathcal{F}_{\mathsf{HL}}^{+}$, whose realisation advantage is charged at $\mathsf{H}_0 \to \mathsf{H}_1$), the signed T_{ic} is correct or the $\mathcal{F}_{\mathsf{derive}}$ dispute catches it (subversion probability at most $\mathsf{Adv}^{\mathsf{derive}}$), and check 4 on the unique scalars passing checks 1–3 (Theorem 2) reproduces exactly the verification verdict of the committed π_{out}. Acceptance therefore certifies the hypothesis of (ii); rejection releases k and routes to `slash-fast`.

(iv) Witness extraction. $(A_{\mathsf{in}}, B_{\mathsf{in}}, C_{\mathsf{in}})$ being a valid inner proof, the `extract` interface of $\mathcal{F}_{\mathsf{wNIZK}}^{C_{\mathsf{assertions}}}$ (replaced at Lemma 13, where its advantage is charged) yields w^{extr} with $C_{\mathsf{assertions}}(x; w^{\mathsf{extr}}) = 1$; $\mathcal{S}_{\mathsf{P}}$ passes it to $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$ via `prove`, and $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$ emits `refund`, matching H_5.

We assemble (i)–(iv) as an identical-until-bad argument: H_5 and H_6 agree on the `refund`/no-`refund` verdict unless one of four events fires, a second valid opening of c_1 ($\le \mathsf{Adv}_{H}^{\mathsf{CR}}$, (i)); the committed π_{out} verifying on a false statement ($\le \mathsf{Adv}_{\mathsf{out}}^{\mathsf{sound},\mathcal{G}_{\mathsf{oGG}}} + \mathsf{Adv}_{\pi_{\mathsf{SRS}}}^{\mathsf{ZK}}$, (ii)); a malformed surviving circuit ($\le \binom{s}{s/2}^{-1}$, (iii)); a lost $\mathcal{F}_{\mathsf{derive}}$ dispute on T_{ic} ($\le \mathsf{Adv}^{\mathsf{derive}}$, (iii)). On the complement of their union the opened tuple is unique, the committed π_{out} verifies iff $(A_{\mathsf{in}}, B_{\mathsf{in}}, C_{\mathsf{in}})$ is valid (Definition 5), every surviving circuit decides outer verification correctly on genuine inputs (Corollary 1), and $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$ emits `refund` exactly when the real protocol does. By the union bound the advantage is at most $\mathsf{Adv}_{H}^{\mathsf{CR}} + \mathsf{Adv}_{\mathsf{out}}^{\mathsf{sound},\mathcal{G}_{\mathsf{oGG}}} + \mathsf{Adv}_{\pi_{\mathsf{SRS}}}^{\mathsf{ZK}} + \binom{s}{s/2}^{-1} + \mathsf{Adv}^{\mathsf{derive}}$, the stated bound. The conditioning is non-circular: the bound of (ii) (Lemma 3) halts at c_1, before the seed reveal on which (iii) is defined. The terms $\mathsf{Adv}_{\mathsf{BHR}}^{\mathsf{garb}}$ and $\mathsf{Adv}^{\mathsf{HL}+}$ are not charged here, the corresponding primitives are already ideal at H_5. □

Lemma 15 (Outcome partition and slashing-branch matching)**.** *Fix corrupt* $\mathtt{P} = \mathcal{A}$ *and honest* $\mathtt{V}$*. The events* $\{\mathsf{Settle}(\mathsf{tr}) = o\}$ *for* $o \in \{\texttt{refund}, \texttt{slash-fast}, \texttt{slash-commit}\}$ *partition the probability space, and on each,* H_6 *emits outcome* o *with the same probability as* H_5*, conditioned on that event, up to the terms already charged in the* `refund` *case. In particular the slashing branches contribute no additional term.*

Proof. Partition. Settle is a function, so the three preimages are disjoint and exhaustive. We use the protocol's settlement priority: $\mathsf{Settle}(\mathsf{tr}) = \texttt{slash-commit}$ if tr contains any of the four script-checkable evidence items of the evidence set E (a c_1-inconsistent opening; an equivocated one-time signature, $\mathcal{F}_{\mathsf{HL}}^{+}$; a canonical-label reveal whose bit differs from $\mathsf{bit}_{\beta(j)}(H(\mathsf{seed}\|\texttt{label}(j)))$, $\mathcal{F}_{\mathsf{HL}}^{+}$; a lost $\mathcal{F}_{\mathsf{derive}}$ dispute on T_{ic}); else $\texttt{slash-fast}$ if some surviving copy released its reject label (capsule k recovered); else $\texttt{refund}$ by timelock. Each predicate is a deterministic, on-chain-checkable function of tr, so Settle is well defined.

`slash-fast`. Here no evidence in E is present, and some surviving honestly-evaluated copy emitted its reject label. By the enforcement corollary `cor:inner-outer`, on the genuine scalars a surviving copy emits reject iff the committed π_{out} fails check 4, i.e. iff $(A_{\mathsf{out}}, B_{\mathsf{out}}, C_{\mathsf{out}})$ does not verify under $\mathsf{vk}_{\mathsf{outer}}(\mathsf{seed})$; and on *signed* scalars $(a_{\mathsf{out}}, b_{\mathsf{out}}, c_{\mathsf{out}})$ inconsistent with the committed points, checks 1–3 fail and the copy emits reject. Both are reject paths that release k. In H_6, $\mathcal{S}_{\mathtt{P}}$ selects $\texttt{slash-fast}$ at $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$; clause III permits a corrupt prover to settle on $\texttt{slash-fast}$ at *either* bit (a false statement, $b = 0$, through the non-verifying π_{out}; a true statement, $b = 1$, through self-inconsistent scalars), so $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$ emits $\texttt{slash-fast}$ and reports the corresponding acceptance bit to $\mathtt{V}$. The only stochastic gap is whether a *malformed* surviving copy could have emitted accept where the genuine evaluation rejects; that probability is the cut-and-choose/garbling-soundness mass $\binom{s}{s/2}^{-1} + \mathsf{Adv}_{\mathsf{BHR}}^{\mathsf{garb}}$ already charged at $\mathsf{H}_3 \to \mathsf{H}_4$ and at step (iii) of the $\texttt{refund}$ analysis, and it is not recharged here. Conditioned on $\{\mathsf{Settle} = \texttt{slash-fast}\}$ the two worlds therefore emit $\texttt{slash-fast}$ with equal probability.

`slash-commit`. Here tr contains an evidence item $e \in E$. Each item is script-checkable and its presence is decided identically in both worlds, because the underlying objects are idealised: equivocation and wrong-bit reveals through $\mathcal{F}_{\mathsf{HL}}^{+}$ (charged $\mathsf{Adv}^{\mathsf{HL}+}$), the T_{ic} dispute through $\mathcal{F}_{\mathsf{derive}}$ (charged $\mathsf{Adv}^{\mathsf{derive}}$), and c_1-inconsistency through collision-resistance of H (charged $\mathsf{Adv}_{H}^{\mathsf{CR}}$). All three terms already appear in the bridge bound. In H_6, $\mathcal{S}_{\mathtt{P}}$ selects $\texttt{slash-commit}$ at $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$; the witness-less $\texttt{slash-commit}$ of the $\texttt{init}$ branch is available against a corrupt prover (clause III), so $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$ emits $\texttt{slash-commit}$. Conditioned on $\{\mathsf{Settle} = \texttt{slash-commit}\}$ the worlds agree except on the $\mathcal{F}_{\mathsf{HL}}^{+}/\mathcal{F}_{\mathsf{derive}}/H$ idealisation events, which are charged once.

No double counting. Every term invoked above ($\binom{s}{s/2}^{-1}$, $\mathsf{Adv}_{\mathsf{BHR}}^{\mathsf{garb}}$, $\mathsf{Adv}^{\mathsf{HL}+}$, $\mathsf{Adv}^{\mathsf{derive}}$, $\mathsf{Adv}_{H}^{\mathsf{CR}}$) is charged at the hybrid step that introduces the corresponding idealisation, and each slashing branch only *invokes* those already-charged events rather than introducing new ones; the soundness term $\mathsf{Adv}_{\mathsf{out}}^{\mathsf{sound},\mathcal{G}_{\mathsf{oGG}}}$ is consumed solely in the $\texttt{refund}$ branch (the only branch in which a corrupt prover gains by having a non-verifying π_{out} accepted). Summing over the partition reproduces the bridge bound of `lem:hyb-bridge-Pcorrupt` with no slashing-branch surcharge. □

Lemma 16 ($\mathsf{H}_6 \to \mathsf{H}_7$: trivial)**.** $\Pr[\mathcal{Z} = 1 \mid \mathsf{H}_6] = \Pr[\mathcal{Z} = 1 \mid \mathsf{H}_7]$.

Proof. In H_6, the simulator $\mathcal{S}_{\mathtt{P}}$ drives $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$ through its standard interface; the ledger-visible transcript is the simulated transcript $\mathcal{S}_{\mathtt{P}}$ produces. In H_7, the simulator is invoked in the standard ideal-world execution. The two are by construction the same. □

Theorem 3 (Corrupt-prover indistinguishability)**.** *For every PPT environment $\mathcal{Z}$ and every PPT adversary $\mathcal{A}$ with $\mathcal{C} = \{\mathtt{P}\}$,*

$$\begin{aligned}\left|\Pr[\mathcal{Z}^{\mathsf{H}_0} = 1] - \Pr[\mathcal{Z}^{\mathsf{H}_7} = 1]\right| \\ \leq \mathsf{Adv}^{\mathsf{HL}+} + \varepsilon_{\mathcal{R}} + \mathsf{Adv}^{\mathsf{garb}}_{\mathsf{BHR}} + \binom{s}{s/2}^{-1} + \mathsf{Adv}^{\mathsf{Groth16}/\mathcal{G}_{\mathsf{oGG}}}_{\mathsf{in}} \\ + \mathsf{Adv}^{\mathsf{sound},\mathcal{G}_{\mathsf{oGG}}}_{\mathsf{out}} + \mathsf{Adv}^{\mathsf{CR}}_{H} + \mathsf{Adv}^{\mathsf{derive}} + \mathsf{Adv}^{\mathsf{ZK}}_{\pi_{\mathsf{SRS}}}.\end{aligned}$$

Proof. Telescoping sum of Lemmas 9–16: $\mathsf{Adv}^{\mathsf{HL}+}$ from $\mathsf{H}_0 \to \mathsf{H}_1$ (Lemma 9), $\varepsilon_{\mathcal{R}}$ from $\mathsf{H}_1 \to \mathsf{H}_2$ (Lemma 10), 0 from $\mathsf{H}_2 \to \mathsf{H}_3$ (Lemma 11), $\mathsf{Adv}^{\mathsf{garb}}_{\mathsf{BHR}}$ from $\mathsf{H}_3 \to \mathsf{H}_4$ (Lemma 12), $\mathsf{Adv}^{\mathsf{Groth16}/\mathcal{G}_{\mathsf{oGG}}}_{\mathsf{in}}$ from $\mathsf{H}_4 \to \mathsf{H}_5$ (Lemma 13; the inner instance only), $\mathsf{Adv}^{\mathsf{sound},\mathcal{G}_{\mathsf{oGG}}}_{\mathsf{out}} + \mathsf{Adv}^{\mathsf{CR}}_{H} + \binom{s}{s/2}^{-1} + \mathsf{Adv}^{\mathsf{ZK}}_{\pi_{\mathsf{SRS}}} + \mathsf{Adv}^{\mathsf{derive}}$ from $\mathsf{H}_5 \to \mathsf{H}_6$ (Lemma 14), 0 from $\mathsf{H}_6 \to \mathsf{H}_7$ (Lemma 16). Summing yields the stated bound. □

8.2 The both-honest hybrid sequence

This sequence is shorter because there is no extraction; the simulator produces simulated cryptographic objects without needing to recover anything from $\mathcal{A}$. The hybrid sequence follows:

- $\mathsf{H}^{\emptyset}_0$: real execution with honest $\mathtt{P}, \mathtt{V}$ and a passive $\mathcal{A}$ (the environment corrupts no party but observes public messages).
- $\mathsf{H}^{\emptyset}_1$: (vacuous, no Pedersen layer; on-chain commitments are hash commitments).
- $\mathsf{H}^{\emptyset}_2$: replace real garbling with $\mathcal{F}_{\mathsf{GC}}$ in simulation mode.
- $\mathsf{H}^{\emptyset}_3$: replace honest $\mathtt{P}$'s real inner Groth16 prove with $\mathcal{F}_{\mathsf{wNIZK}}$'s `simulate` interface when $b = 1$, and with the honest prover on the leaked w when $b = 0$; the outer proof is computed by the honest outer prover on the public trace in both cases.
- $\mathsf{H}^{\emptyset}_4 = \mathsf{EXEC}_{\mathcal{F}_{\mathsf{DV\text{-}Groth16}},\mathcal{S}_{\emptyset},\mathcal{Z}}$: the ideal world with $\mathcal{S}_{\emptyset}$ from Algorithm 9.

The hash-lock, beacon, and cut-and-choose components do not contribute terms in this sequence because both parties are honest and these components only matter for catching adversarial deviations. Their underlying primitives are still invoked by the real protocol, but the transitions to their ideal counterparts produce zero distinguishing advantage when no party deviates.

Lemma 17 ($\mathsf{H}_0 \to \mathsf{H}_1$: vacuous)**.** $\left|\Pr[\mathcal{Z} = 1 \mid \mathsf{H}_0] - \Pr[\mathcal{Z} = 1 \mid \mathsf{H}_1]\right| \leq 0$.

Proof. There is no Pedersen layer in the realigned protocol; on-chain commitments are hash commitments idealised at the $\mathcal{F}^{+}_{\mathsf{HL}}/c_1$ steps. This transition is vacuous and is retained only to keep the hybrid index aligned with the corrupt-prover sequence. □

Lemma 18 ($\mathsf{H}_1 \to \mathsf{H}_2$: garbling privacy)**.** $\left|\Pr[\mathcal{Z} = 1 \mid \mathsf{H}_1] - \Pr[\mathcal{Z} = 1 \mid \mathsf{H}_2]\right| \leq \mathsf{Adv}^{\mathsf{garb}}_{\mathsf{BHR}}(\lambda)$.

Proof. In H_1, the real BHR garbled circuits are produced from honest $\mathtt{P}$'s input encoding and the circuit $\mathcal{C}_{\mathsf{GC}}$. In H_2, the simulator invokes $\mathcal{F}_{\mathsf{GC}}$'s `simulate` interface (clause IV of Algorithm 2) to produce garbled circuits of $\mathcal{C}_{\mathsf{GC}}$ that evaluate to the label matching the bit b leaked by $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$, without committing to specific inputs. The distinguishing advantage is bounded by BHR privacy. □

Lemma 19 ($\mathsf{H}_2 \to \mathsf{H}_3$: Groth16 simulation)**.**

$$\left|\Pr[\mathcal{Z} = 1 \mid \mathsf{H}_2] - \Pr[\mathcal{Z} = 1 \mid \mathsf{H}_3]\right| \leq \mathsf{Adv}^{\mathsf{Groth16}/\mathcal{G}_{\mathsf{oGG}}}(\lambda, q, |C_{\mathsf{assertions}}|).$$

Proof. In H_2, honest P's π_{in} is produced by the standard Groth16 prover on (x, w). In H_3, when $b = 1$ the simulator invokes $\mathcal{F}_{\mathsf{wNIZK}}$.`simulate` to produce a verifying $\pi^{\mathcal{S}}_{\mathsf{in}}$ without w; by [[BFKT24], Thm. 1] this costs at most $\mathsf{Adv}^{\mathsf{Groth16}/\mathcal{G}_{\mathsf{oGG}}}(\lambda, q, |C_{\mathsf{assertions}}|)$. When $b = 0$ the simulator runs the honest prover on the leaked w, zero advantage. The outer proof contributes nothing in either branch: its witness is the public trace and the simulator runs the honest, deterministic outer prover (Definition 5). □

Lemma 20 ($\mathsf{H}_3 \to \mathsf{H}_4$: ideal world)**.** $\Pr[\mathcal{Z} = 1 \mid \mathsf{H}_3] = \Pr[\mathcal{Z} = 1 \mid \mathsf{H}_4]$.

Proof. By this point the simulator's actions are exactly those of $\mathcal{S}_\emptyset$ as described in Algorithm 9: produce $\pi^{\mathcal{S}}_{\mathsf{in}}$ per the b-branch of Algorithm 9, $\pi^{\mathcal{S}}_{\mathsf{out}}$ via the honest outer prover on the public trace, and the garbled circuits via $\mathcal{F}_{\mathsf{GC}}$ programmed to b. The transition to the ideal world is the standard UC re-labelling. □

Theorem 4 (Both-honest indistinguishability)**.** *For every PPT environment $\mathcal{Z}$ and every PPT adversary $\mathcal{A}$ with $\mathcal{C} = \emptyset$,* $\left|\Pr[\mathcal{Z}^{\mathsf{H}_0} = 1] - \Pr[\mathcal{Z}^{\mathsf{H}_4} = 1]\right| \leq \mathsf{Adv}^{\mathsf{garb}}_{\mathsf{BHR}} + \mathsf{Adv}^{\mathsf{Groth16}/\mathcal{G}_{\mathsf{oGG}}}_{\mathsf{in}}$.

Proof. The result follows from telescoping sum of Lemmas 17–20. □

8.3 The corrupt-verifier hybrid sequence

This sequence requires the simulator to produce simulated proofs without a witness while interacting with corrupt V. Two specific technical challenges arise: (a) the simulator must produce a π_{in} that is identically distributed to honest P's, even when $\mathcal{Z}$ hands honest P an invalid witness w (handled by the Path-A leakage clause of Algorithm 1 and Remark 30); (b) the simulator must condition on the event that $\mathcal{A}$ cannot induce slashing against honest P (handled by Lemma 8, contributing $\mathsf{Adv}^{\mathsf{CR}}_H + \mathsf{Adv}^{\mathsf{derive}}$ to the bound beyond the per-hybrid contributions). The hybrid sequence follows:

- $\mathsf{H}^{\{\mathtt{V}\}}_0$: real execution with honest P and $\mathcal{A}$-as-V.
- $\mathsf{H}^{\{\mathtt{V}\}}_1$: replace $\mathcal{F}^{+}_{\mathsf{HL}}$, $\mathcal{F}_{\mathsf{Beacon}}$ with their ideal counterparts.
- $\mathsf{H}^{\{\mathtt{V}\}}_2$: replace real garbling with $\mathcal{F}_{\mathsf{GC}}$ in simulation mode (programmed to output the bit b that $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$ will deliver via the `acceptance-bit` private channel).
- $\mathsf{H}^{\{\mathtt{V}\}}_3$: replace honest P's real Groth16 prove of π_{in} with a simulator invocation that uses the witness leaked by $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$. Concretely, $\mathcal{S}_{\mathtt{V}}$ reads (b, w) from the `prove-leak` message of Algorithm 1 clause II and runs the honest Groth16 prover $\mathsf{Prove}_{\mathsf{in}}(\mathsf{SRS}^{\mathsf{in}}, x, w)$ to produce $\pi^{\mathcal{S}}_{\mathsf{in}}$. If $b = 1$, $\pi^{\mathcal{S}}_{\mathsf{in}}$ verifies; if $b = 0$, it does not, in both cases the distribution matches honest P's output exactly (Remark 30). The outer proof $\pi^{\mathcal{S}}_{\mathsf{out}}$ is produced by the honest outer prover on the public verification trace (Definition 5) in both branches; it requires neither the witness nor any simulation.
- $\mathsf{H}^{\{\mathtt{V}\}}_4 = \mathsf{EXEC}_{\mathcal{F}_{\mathsf{DV\text{-}Groth16}}, \mathcal{S}_{\mathtt{V}}, \mathcal{Z}}$: the ideal world.

Lemma 21 ($\mathsf{H}_0 \to \mathsf{H}_1$: HL + Beacon, combined)**.**

$$\left|\Pr[\mathcal{Z} = 1 \mid \mathsf{H}_0] - \Pr[\mathcal{Z} = 1 \mid \mathsf{H}_1]\right| \leq \mathsf{Adv}^{\mathsf{HL}+} + \varepsilon_{\mathcal{R}}.$$

Proof. The three replacements are independent of each other; the combined transition is a presentational compression of three sub-transitions whose bounds sum by the triangle inequality. The HL replacement contributes $\mathsf{Adv}^{\mathsf{HL+}}$ by the same argument as Lemma 9. The beacon replacement contributes $\varepsilon_{\mathcal{R}}$ by Lemma 10. There is no Pedersen replacement (the realigned protocol has no Pedersen layer). The Groth16 advantage is charged at the Groth16 transition (Lemma 23). □

Lemma 22 ($\mathsf{H}_1 \to \mathsf{H}_2$: garbled-circuit simulation)**.**

$$\left|\Pr[\mathcal{Z} = 1 \mid \mathsf{H}_1] - \Pr[\mathcal{Z} = 1 \mid \mathsf{H}_2]\right| \leq \mathsf{Adv}^{\mathsf{garb}}_{\mathsf{BHR}}(\lambda).$$

Proof. In H_1, honest `P` produces real BHR garbled circuits. In H_2, the simulator invokes $\mathcal{F}_{\mathsf{GC}}$'s `simulate` interface in malicious-secure mode: the simulated garbled handles must look like real garblings under cut-and-choose opening by $\mathcal{A}$, while their evaluation on the planned inputs in Phase III produces the bit b delivered by $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$ via `acceptance-bit`. The cut-and-choose factor $\binom{s}{s/2}^{-1}$ from Assumption 3 does not appear here because corrupt `V` does not garble; the cut-and-choose check is performed by $\mathcal{A}$ on the simulator's (correctly-emulated) garbled circuits and either succeeds (the protocol proceeds) or fails (the protocol aborts, which is equivalent across the two hybrids). The distinguishing advantage is therefore the BHR privacy/authenticity loss only. □

Lemma 23 ($\mathsf{H}_2 \to \mathsf{H}_3$: Groth16 simulation)**.**

$$\left|\Pr[\mathcal{Z} = 1 \mid \mathsf{H}_2] - \Pr[\mathcal{Z} = 1 \mid \mathsf{H}_3]\right| \leq \mathsf{Adv}^{\mathsf{Groth16}/\mathcal{G}_{\mathsf{oGG}}}(\lambda, q, |C_{\mathsf{assertions}}|).$$

Proof. π_{in} analysis: when $b = 0$, $\mathcal{S}_{\mathtt{V}}$ runs the honest prover on the leaked (x, w), identical distributions, zero advantage. When $b = 1$, $\mathcal{S}_{\mathtt{V}}$ invokes $\mathcal{F}_{\mathsf{wNIZK}}$.`simulate` in place of the honest prover; by [[BFKT24], Thm. 1] this simulation indistinguishability is bounded by $\mathsf{Adv}^{\mathsf{Groth16}/\mathcal{G}_{\mathsf{oGG}}}(\lambda, q, |C_{\mathsf{assertions}}|)$.

π_{out} analysis: $\mathcal{S}_{\mathtt{V}}$ runs the *honest* outer prover on the public trace of $\mathsf{stmt}^{\mathcal{S}}$ in both branches (Algorithm 11; the proof is deterministic with coins 0). The distribution is identical to honest `P`'s and contributes zero. No outer simulation, extraction, or trapdoor recovery occurs at this layer. □

Lemma 24 ($\mathsf{H}_3 \to \mathsf{H}_4$: slashing-resistance conditioning)**.**

$$\left|\Pr[\mathcal{Z} = 1 \mid \mathsf{H}_3] - \Pr[\mathcal{Z} = 1 \mid \mathsf{H}_4]\right| \leq \mathsf{Adv}^{\mathsf{CR}}_{H} + \mathsf{Adv}^{\mathsf{derive}}.$$

Proof. At H_3, the simulator's transcript matches honest `P`'s real-world transcript bit-for-bit on the cryptographic objects already replaced (garbling, π_{in}, π_{out}, hash commitments). What remains is to bound the probability that $\mathcal{A}$ (corrupt `V`) drives the on-chain settlement to a slashing outcome that $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$ would not authorise. $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$'s decision rule under honest `P` emits `refund` if $b = 1$ and `slash-fast` if $b = 0$; the simulator $\mathcal{S}_{\mathtt{V}}$ produces a real-world transcript whose on-chain settlement matches this outcome, provided $\mathcal{A}$ does not produce a non-matching slashing transaction.

By Lemma 8 (whose case E_0 routes a non-expansion SRS to `refund` via the guard of Algorithm 7, contributing nothing) and the outcome partition of Lemma 25, the probability that $\mathcal{A}$ drives $\mathcal{G}_{\mathsf{Ledger}}$ to emit a slashing outcome contradicting $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$'s decision is at most $\mathsf{Adv}^{\mathsf{CR}}_{H} + \mathsf{Adv}^{\mathsf{garb}}_{\mathsf{BHR}} + \mathsf{Adv}^{\mathsf{HL+}} + \mathsf{Adv}^{\mathsf{derive}}$.

The garbling and hash-lock/adjudication terms have already been paid at $\mathsf{H}_1 \to \mathsf{H}_2$ and $\mathsf{H}_0 \to \mathsf{H}_1$ respectively, so the marginal contribution at this step is $\mathsf{Adv}^{\mathsf{CR}}_{H} + \mathsf{Adv}^{\mathsf{derive}}$:

collision-finding against the slashing-key pin $h = H(k)$ and the Phase-III c_1-consistency check, and subversion of the $\mathcal{F}_{\mathsf{derive}}$ dispute on the public derivation of T_{ic}. □

Lemma 25 (Corrupt-verifier outcome partition)**.** *Fix honest* P *and corrupt* $\mathtt{V} = \mathcal{A}$*. The events* $\{\mathsf{Settle}(\mathsf{tr}) = o\}$ *for* $o \in \{\texttt{refund}, \texttt{slash-fast}, \texttt{slash-commit}\}$ *partition the space, and on each, the ideal execution* $\mathsf{H}_4^{\{\mathtt{V}\}}$ *emits* o *with the same probability as* $\mathsf{H}_3^{\{\mathtt{V}\}}$ *conditioned on that event, up to the terms of Lemma 8. No outcome contributes a term beyond those already charged in the corrupt-verifier sequence.*

Proof. Settle is a function under the settlement priority of Lemma 15, so the three preimages are disjoint and exhaustive. Honest P follows the protocol, including the seed-expansion guard of Algorithm 7. Write $b = C_{\mathsf{assertions}}(x; w)$ for the (possibly invalid) witness $\mathcal{Z}$ handed P.

refund, three disjoint sub-events. (a) The guard passes and $b = 1$: by Corollary 1 every surviving circuit accepts on P's genuine inputs, no reject label is emitted, no evidence exists, and the T_1 path fires; $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$ clause III emits refund at $b = 1$. (b) The guard *fails* (event E_0 of Lemma 8): P publishes no UTXO3, so no evaluate is answerable and no evidence exists; the real world settles refund at T_1 (Lemma 4), and in the ideal world $\mathcal{S}_{\mathtt{V}}$ emulates the halt and drives no outcome, so $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$ clause IV (timeout) emits refund. (c) is subsumed by (a)/(b). Both worlds agree on each.

slash-fast. Against honest P this arises in two ways. (i) $b = 0$ with the guard passing: π_{in} does not verify, so $R_{\mathsf{outer}}(\mathsf{stmt}) = 0$ and P's committed π_{out} fails check 4 under $\mathsf{vk}_{\mathsf{outer}}(\mathsf{seed})$; her genuine scalars pass checks 1–3 (they are the true discrete logarithms of her points) but the copy emits reject on check 4, releasing k. This is the honest reject path, matched by $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$ clause III ($b = 0$, honest P $\to$ slash-fast); it contributes no term. (ii) $\mathcal{A}$ forges the reject label or inverts $h = H(k)$: bounded by $\mathsf{Adv}^{\mathsf{garb}}_{\mathsf{BHR}} + \mathsf{Adv}^{\mathsf{HL+}}$ (event E_1 of Lemma 8), already charged. The guard is orthogonal to (i): it concerns V's SRS, not P's witness.

slash-commit. Against honest P this requires a false positive evidence item; by event E_2 of Lemma 8 its probability is at most $\mathsf{Adv}^{\mathsf{CR}}_H + \mathsf{Adv}^{\mathsf{HL+}} + \mathsf{Adv}^{\mathsf{derive}}$, of which the marginal charge at $\mathsf{H}_3 \to \mathsf{H}_4$ is $\mathsf{Adv}^{\mathsf{CR}}_H + \mathsf{Adv}^{\mathsf{derive}}$ (Lemma 24). $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$ under honest P never authorises this outcome, so the entire mass is distinguishing and equals the charged bound.

No new term. E_0 contributes 0; every other charge is one of $\mathsf{Adv}^{\mathsf{garb}}_{\mathsf{BHR}}, \mathsf{Adv}^{\mathsf{HL+}}, \mathsf{Adv}^{\mathsf{CR}}_H, \mathsf{Adv}^{\mathsf{derive}}$, each introduced once at its hybrid step. Summing over the partition reproduces Theorem 5 with no additional surcharge. □

Theorem 5 (Corrupt-verifier indistinguishability)**.** *For every PPT environment* $\mathcal{Z}$ *and every PPT adversary* $\mathcal{A}$ *with* $\mathcal{C} = \{\mathtt{V}\}$*,*

$$\left|\Pr[\mathcal{Z}^{\mathsf{H}_0} = 1] - \Pr[\mathcal{Z}^{\mathsf{H}_4} = 1]\right| \leq \mathsf{Adv}^{\mathsf{HL+}} + \varepsilon_{\mathcal{R}} + \mathsf{Adv}^{\mathsf{garb}}_{\mathsf{BHR}} + \mathsf{Adv}^{\mathsf{Groth16}/\mathcal{G}_{\mathsf{oGG}}}_{\mathsf{in}} + \mathsf{Adv}^{\mathsf{CR}}_H + \mathsf{Adv}^{\mathsf{derive}}.$$

Proof. Telescoping sum of Lemmas 21–24. □

Remark 33. The corrupt-verifier bound differs from the corrupt-prover bound (Theorem 3) by the absence of the cut-and-choose factor $\binom{s}{s/2}^{-1}$, the outer-soundness term $\mathsf{Adv}^{\mathsf{sound},\mathcal{G}_{\mathsf{oGG}}}_{\mathsf{out}}$, and $\mathsf{Adv}^{\mathsf{ZK}}_{\pi_{\mathsf{SRS}}}$: all three defend against a cheating prover/garbler, and in the corrupt-verifier case the prover is honest. All other terms appear in both bounds.

8.4 Combining the cases: proof of Theorem 1

Proof of Theorem 1. Define the simulator $\mathcal{S}$ as the dispatcher that, on session start, reads the corruption set $\mathcal{C} \subseteq \{\mathtt{P}, \mathtt{V}\}$ declared by $\mathcal{A}$ and invokes one of the four pattern-specific simulators: $\mathcal{S}_\emptyset$ (Algorithm 9) if $\mathcal{C} = \emptyset$; $\mathcal{S}_\mathtt{P}$ (Algorithm 10) if $\mathcal{C} = \{\mathtt{P}\}$; $\mathcal{S}_\mathtt{V}$ (Algorithm 11) if $\mathcal{C} = \{\mathtt{V}\}$; The translation simulator (Section 7) if $\mathcal{C} = \{\mathtt{P}, \mathtt{V}\}$.

By Theorems 3, 4, and 5 and the immediate indistinguishability of the both-corrupted case, the distinguishing advantage of $\mathcal{Z}$ is bounded by the maximum over the four patterns of the per-pattern bound: $\max\Big\{0, \mathrm{Bound}_\emptyset, \mathrm{Bound}_{\{\mathtt{P}\}}, \mathrm{Bound}_{\{\mathtt{V}\}}\Big\}$.

Comparing:

$$\begin{aligned}
\mathrm{Bound}_\emptyset &= \mathsf{Adv}^{\mathsf{garb}}_{\mathsf{BHR}} + \mathsf{Adv}^{\mathsf{Groth16}/\mathcal{G}_{\mathsf{oGG}}}_{\mathsf{in}}, \\
\mathrm{Bound}_{\{\mathtt{V}\}} &= \mathrm{Bound}_\emptyset + \mathsf{Adv}^{\mathsf{HL}+} + \varepsilon_{\mathcal{R}} + \mathsf{Adv}^{\mathsf{CR}}_{H} + \mathsf{Adv}^{\mathsf{derive}}, \\
\mathrm{Bound}_{\{\mathtt{P}\}} &= \mathrm{Bound}_{\{\mathtt{V}\}} + \binom{s}{s/2}^{-1} + \mathsf{Adv}^{\mathsf{sound},\mathcal{G}_{\mathsf{oGG}}}_{\mathsf{out}} + \mathsf{Adv}^{\mathsf{ZK}}_{\pi_{\mathsf{SRS}}}.
\end{aligned}$$

The corrupt-prover bound dominates. Equivalently:

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{HL}+} + \varepsilon_{\mathcal{R}} + \mathsf{Adv}^{\mathsf{garb}}_{\mathsf{BHR}} + \binom{s}{s/2}^{-1} + \mathsf{Adv}^{\mathsf{Groth16}/\mathcal{G}_{\mathsf{oGG}}}_{\mathsf{in}} + \\
\mathsf{Adv}^{\mathsf{sound},\mathcal{G}_{\mathsf{oGG}}}_{\mathsf{out}} + \mathsf{Adv}^{\mathsf{CR}}_{H} + \mathsf{Adv}^{\mathsf{derive}} + \mathsf{Adv}^{\mathsf{ZK}}_{\pi_{\mathsf{SRS}}},
\end{aligned}$$

which matches the bound stated in Theorem 1. □

9 Discussion and future work

9.1 Summary of the construction

We have presented $\Pi_{\mathsf{DV\text{-}Groth16}}$ and proved it UC-realises $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$ in the GUC-hybrid model $(\mathcal{G}_{\mathsf{Ledger}}, \mathcal{G}_{\mathsf{oGG}}\,;\, \mathcal{F}_{\mathsf{GC}}, \mathcal{F}_{\mathsf{Beacon}}, \mathcal{F}^{+}_{\mathsf{HL}}, \mathcal{F}_{\mathsf{derive}})$ against static corruption, with the concrete bound stated in Theorem 1. The proof structure separates the cryptographic content into building blocks (Section 5), protocol specification (Section 6), and a three-case hybrid argument (Sections 7, and 8), with no relegation to appendices.

Three architectural choices underpin the construction. *Native one-layer recursion*: the outer circuit is the inner Groth16 verifier itself, computed natively over a 2-chain, so the per-session trapdoor SRS is compact and independent of the assertion circuit, while the heavy inner setup is public and amortised across all sessions. *Garbled scalarization*: the garbled circuit verifies the *outer* proof in scalar form (Theorem 2), the only layer at which proof discrete logarithms ever become available to any party, with three fixed-base consistency checks binding the revealed scalars to pre-reveal commitments. *Complementary input authority*: each garbled-circuit input wire is bound by exactly one of three mechanisms (signed scalars checked in-circuit, seed-derived canonical labels, a dispute-adjudicated derived value), jointly covering both corruption patterns without per-case constructions.

9.2 Limitations

Static corruption. Our analysis is for static corruption: $\mathcal{Z}$ chooses the corruption set $\mathcal{C} \subseteq \{\mathtt{P}, \mathtt{V}\}$ at session start. Adaptive corruption (in which $\mathcal{Z}$ may corrupt parties mid-execution based on observed protocol messages) is not covered. The construction has the right

ingredients for adaptive security, the explainability property of $\mathcal{G}_{\mathsf{oGG}}$ supports late-witness-revelation of previously-simulated proofs; the hash commitments admit programmable-RO equivocation; the BHR garbling scheme has known non-committing variants, but the full proof requires additional machinery and is left to future work.

Single public input. We analyse the protocol with $n = 1$ public input. Multi-input statements ($n > 1$) are a straightforward extension: the QAP grows linearly in n, the commitment structure is unchanged (only the public-input portion of the inner key grows with n), so the bound is unchanged. We omit the multi-input write-up to keep the notation focused on the cryptographic content rather than the bookkeeping.

Off-chain bandwidth. The protocol's dominant off-chain cost is the cut-and-choose parameter s. To achieve $2^{-\lambda}$ soundness, s must satisfy $\binom{s}{s/2}^{-1} \leq 2^{-\lambda}$, which for $\lambda = 128$ is met at $s \approx 128$ *total* circuits (of which $s/2 \approx 64$ survive), since $\binom{s}{s/2} = \Theta(2^s/\sqrt{s})$ gives $s \approx \lambda + O(\log \lambda)$; each surviving circuit contains the full garbled $\mathcal{C}_{\mathsf{GC}}$. The on-chain cost is independent of s (only the master commitment c_1 and the Merkle root $\mathsf{root}_{\mathsf{GC}}$ are posted), but the off-chain transmission of garbled tables and openings is substantial, and we state the elephant plainly: $\mathcal{C}_{\mathsf{GC}}$ is dominated by three fixed-base scalar multiplications over the 761-bit outer base field. A naive in-circuit realization is on the order of 10^8–10^9 AND gates per copy under Boolean garbling; the windowed comb construction of Remark 20, together with an optimized field-multiplication gadget and shared subcircuits, reduces this by a further ~ 3–$4\times$, *engine-neutrally*, changing only the garbled circuit and no proof, to the order of a few hundred MB per copy (a working estimate, to be pinned by the deployment benchmarks), times s copies. On BW6-761's 761-bit coordinates the windowing gain is multiplexer-limited (Remark 20), so this $\sim$3–4$\times$ is close to the ceiling of engine-neutral optimization; the cost remains the dominant one of the construction. Reducing it by further orders of magnitude is not a circuit rewrite but an *engine* change: the protocol concentrates *all* elliptic-curve work into exactly three garbled fixed-base multiplications, precisely the operation addressed by EC-homomorphic garbling primitives, Argo MAC [EL26] is a direct candidate, Glock [Eag25] explores asymptotically smaller garblings, and the arithmetic garbling of OHMG [**ohmg2026**] is authenticity-preserving and UC-composable with the present construction. We treat these as future work (see below), as each replaces the Boolean garbling engine rather than the circuit.

Parametric instantiations. $\Pi_{\mathsf{DV\text{-}Groth16}}$ is parametric in the realisations of $\mathcal{F}_{\mathsf{HL}}^{+}$, $\mathcal{F}_{\mathsf{derive}}$, $\mathcal{F}_{\mathsf{Beacon}}$, and the hash function H. The security bound's terms $\mathsf{Adv}^{\mathsf{HL}+}$, $\mathsf{Adv}^{\mathsf{derive}}$, $\varepsilon_{\mathcal{R}}$, $\mathsf{Adv}_{H}^{\mathsf{CR}}$ absorb the realisation-specific advantages; specific instantiations (Lamport [Lam79] vs Winternitz [Mer87] vs ESSPI [LJF25] for hash-locks; commit-reveal vs block-hash-based vs Drand for beacon) tune the constants but do not affect the qualitative bound.

Designated enforcement, not deniability. With a ceremony inner SRS, π_{in} is transferable evidence: anyone holding it verifies it against the public $\mathsf{vk}_{\mathsf{in}}$, and no party holds a trapdoor with which it could have been forged. Post seed-reveal, the outer layer is likewise publicly checkable. What the construction designates is *enforcement*: only `V` can evaluate the surviving circuits before settlement, and only `V`'s script path consumes the slashing key (Proposition 2). Designated-verifier *deniability*, the trapdoor-holder-could-have-forged property, requires the trapdoor SRS to cover the proof of the statement itself: the variant with a per-session SRS for $C_{\mathsf{assertions}}$ achieves it, at per-proof setup cost linear in $|C_{\mathsf{assertions}}|$ and without compatibility with pre-existing proofs. The two constructions are the two points of this trade-off.

9.3 Future work

Adaptive corruption. The most natural extension is to lift the static-corruption restriction. The standard machinery for adaptive UC commitments and adaptive garbling (non-committing encryption [DN02], programmable-RO hash-commitment equivocation, programmable-RO simulators) is compatible with our construction; what remains is to verify that the joint enforcement of the complementary input authority does not interact badly with the simulator's equivocation power. This is a finite amount of additional analysis and we expect it to go through.

Alternative SNARKs. The outer/inner split is not specific to Groth16. The framework applies to any SNARK whose verification equation can be cast as an algebraic identity over scalars; this includes Plonk-style universal-setup SNARKs and KZG-based polynomial-commitment schemes. The scalarization identity (Theorem 2) and the native-recursion layer would need to be re-derived for the new SNARK's verification structure, but the cryptographic-content load is borne by the outer Groth16 (which would remain Groth16 in this paper's analysis); only the inner SNARK changes. The cost of the construction is dominated by the outer Groth16, so swapping the inner SNARK does not significantly affect concrete efficiency.

Concrete instantiation analysis. The bound is concrete but its terms contain implicit constants: $\mathsf{Adv}^{\mathsf{Groth16}/\mathcal{G}_{\mathsf{oGG}}}$ depends on the QAP structure of the inner and outer circuits, and the concrete factor $|\mathcal{C}_{\mathsf{verif}}|$ is unspecified. A full instantiation analysis, benchmarking $|\mathcal{C}_{\mathsf{verif}}|$ for the BLS12-377/BW6-761 2-chain with standard one-layer verifier gadgets (gnark, arkworks), measuring $\mathsf{SRS}^{\mathsf{out}}$ size, the $\mathcal{C}_{\mathsf{GC}}$ gate count under a named garbling scheme, and the implied cut-and-choose parameter, would replace the placeholders $\langle c_{\mathsf{verif}} \rangle$ of this paper with measured values; no numeric claim should be read as final until then. Inner curves outside a 2-chain (BN254, BLS12-381) are reachable only by non-native emulation of the verifier circuit, which inflates $|\mathcal{C}_{\mathsf{verif}}|$ by two to three orders of magnitude and inverts the compactness comparison for small assertion circuits; native chain-bottom instantiation is the recommended deployment. This is engineering work; we leave it to a deployment-focused companion paper.

Multi-prover and threshold variants. A natural variant lets $\Pi_{\mathsf{DV\text{-}Groth16}}$ accept a SNARK proof from a threshold-shared prover (e.g., a multi-party trusted setup). This requires extending $\mathcal{F}_{\mathsf{DV\text{-}Groth16}}$'s witness-submission interface to threshold-shared inputs and re-analysing the corrupt-prover hybrid sequence under threshold corruption. The construction itself is compatible.

Batched sessions. One seed and one $\mathsf{SRS}^{\mathsf{out}}$ may cover N inner proofs per session, N outer statements, or a single $\mathcal{C}_{\mathsf{verif}}$ instance verifying N proofs, amortising the verifier's per-session setup to $O(1/N)$, the natural regime for bridge and rollup applications. The UC analysis of the batched session (one reveal gating N settlements) is future work.

9.4 Open problems

Tightness of the cut-and-choose bound. The factor $\binom{s}{s/2}^{-1}$ is the standard cut-and-choose soundness amplification, but is it tight for our specific construction? The c_1-binding of inputs across circuits is stronger than the binding implicit in standard cut-and-choose analyses; this might support a tighter bound. We have not pursued this.

Beacon and hash-commitment realisations. The protocol is parametric in the realisations of $\mathcal{F}_{\mathsf{Beacon}}$ and the on-chain hash commitments. Standardised, formally analysed Bitcoin-script commitment and beacon constructions (with indifferentiability or bias guarantees) would tighten the realisation; we treat these abstractly via $\varepsilon_{\mathcal{R}}$ and $\mathsf{Adv}^{\mathsf{CR}}_{H}$.

Concurrent self-composition. Our analysis is per-session; the GUC composition theorem gives linear scaling in the session count. The empirical question of whether the bound's constants degrade with session count is interesting and depends on whether each session's $\mathcal{G}_{\mathsf{oGG}}$ query budget contributes additively (the [BFKT24] framework suggests yes, but a tighter analysis under specific game distributions might reveal sub-linear scaling in some settings).

References

[AAL+24] Lukas Aumayr, Zeta Avarikioti, Robin Linus, Matteo Maffei, Andrea Pelosi, Christos Stefo, and Alexei Zamyatin. BitVM: quasi-turing complete computation on bitcoin. Cryptology ePrint Archive, Paper 2024/1995, 2024. URL: https:/eprint.iacr.org/2024/1995.pdf.

[ABK+21] Michel Abdalla, Manuel Barbosa, Jonathan Katz, Julian Loss, and Jiayu Xu. Algebraic adversaries in the universal composability framework. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 311–341. Springer, 2021.

[AMLC25] Ramon Amela, Shreemoy Mishra, Sergio Demian Lerner, and Javier Álvarez Cid-Fuentes. Union: a trust-minimized bridge for rootstock. 2025. eprint: 2501.07435. URL: https://arxiv.org/pdf/2501.07435.pdf.

[ARS20] Behzad Abdolmaleki, Sebastian Ramacher, and Daniel Slamanig. Lift-and-shift: obtaining simulation extractable subversion and updatable snarks generically. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1987–2005, 2020.

[BDPT25] Carsten Baum, Bernardo David, Elena Pagnin, and Akira Takahashi. Universally composable interactive and ordered multi-signatures. In *IACR International Conference on Public-Key Cryptography*, pages 3–31. Springer, 2025.

[BFKT24] Jan Bobolz, Pooya Farshim, Markulf Kohlweiss, and Akira Takahashi. The brave new world of global generic groups and uc-secure zero-overhead snarks. In *Theory of Cryptography Conference*, pages 90–124. Springer, 2024.

[BHR12] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 784–796, 2012.

[Bit25] BitLayer. Bitvm bridge. 2025. URL: https://docs.bitlayer.org/docs/BitVMBridge/Concepts/overview/.

[BKSV21] Karim Baghery, Markulf Kohlweiss, Janno Siim, and Mikhail Volkhov. Another look at extraction and randomization of groth's zk-snark. In *International Conference on Financial Cryptography and Data Security*, pages 457–475. Springer, 2021.

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In *International conference on the theory and application of cryptology and information security*, pages 514–532. Springer, 2001.

[BMTZ24] Christian Badertscher, Ueli Maurer, Daniel Tschudi, and Vassilis Zikas. Bitcoin as a transaction ledger: a composable treatment: c. badertscher et al. *Journal of Cryptology*, 37(2):18, 2024.

[Can01] Ran Canetti. Universally composable security: a new paradigm for cryptographic protocols. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 136–145. IEEE, 2001.

[CDPW07] Ran Canetti, Yevgeniy Dodis, Rafael Pass, and Shabsi Walfish. Universally composable security with global setup. In *Theory of Cryptography Conference*, pages 61–85. Springer, 2007.

[CF24] Alessandro Chiesa and Giacomo Fenzi. Zksnarks in the rom with unconditional uc-security. In *Theory of Cryptography Conference*, pages 67–89. Springer, 2024.

[cit25] Citrea: bitcoin's first ZK rollup. Project whitepaper, https://citrea.xyz, 2025.

[CJS14] Ran Canetti, Abhishek Jain, and Alessandra Scafuro. Practical uc security with a global random oracle. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 597–608, 2014.

[CS25] Céline Chevalier and Éric Sageloli. Diving deep into UC: uncovering and resolving issues in universal composability. Cryptology ePrint Archive, Paper 2025/934, 2025. DOI: 10.62056/a6n56chdj. URL: https:/eprint.iacr.org/2025/934.pdf.

[DMS24] Michel Dellepere, Pratyush Mishra, and Alireza Shirzad. Garuda and pari: faster and smaller snarks via equifficient polynomial commitments. *Cryptology ePrint Archive*, 2024.

[DN02] Ivan Damgård and Jesper Buus Nielsen. Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor. In *Annual International Cryptology Conference*, pages 581–596. Springer, 2002.

[Eag25] Liam Eagen. Glock: garbled locks for bitcoin. *Cryptology ePrint Archive*, 2025.

[EL26] Liam Eagen and Ying Tong Lai. Argo mac: garbling with elliptic curve macs. *Cryptology ePrint Archive*, 2026.

[FBF+25] Ariel Futoransky, Fadi Barbara, Ramses Fernandez, Gabriel Larotonda, and Sergio Demian Lerner. TOOP: a transfer of ownership protocol over bitcoin. Cryptology ePrint Archive, Paper 2025/964, 2025. URL: https:/eprint.iacr.org/2025/964.pdf.

[Gro16] Jens Groth. On the size of pairing-based non-interactive arguments. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 305–326. Springer, 2016.

[JSI96] Markus Jakobsson, Kazue Sako, and Russell Impagliazzo. Designated verifier proofs and their applications. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 143–154. Springer, 1996.

[LAA+25] Robin Linus, Lukas Aumayr, Zeta Avarikioti, Matteo Maffei, Andrea Pelosi, Orfeas Thyfronitis Litos, Christos Stefo, David Tse, and Alexei Zamyatin. Bridging bitcoin to second layers via BitVM2. Cryptology ePrint Archive, Paper 2025/1158, 2025. URL: https:/eprint.iacr.org/2025/1158.pdf.

[LAM+24] Sergio Demian Lerner, Ramon Amela, Shreemoy Mishra, Martin Jonas, and Javier Álvarez Cid-Fuentes. Bitvmx: a cpu for universal computation on bitcoin. 2024. eprint: 2405.06842. URL: https://arxiv.org/pdf/2405.06842.pdf.

[Lam79] Leslie Lamport. Constructing digital signatures from a one way function, 1979.

[LF25] Sergio Demian Lerner and Ariel Futoransky. FLEX — capital-efficient optimistic bridges with on-demand security bonds for bitcoin. Cryptology ePrint Archive, Paper 2025/1392, 2025. URL: https:/eprint.iacr.org/2025/1392.pdf.

[Lin16] Yehuda Lindell. Fast cut-and-choose-based protocols for malicious and covert adversaries. *Journal of Cryptology*, 29(2):456–490, 2016.

[Lin25] Robin Linus. Bitvm 3s–garbled circuits for efficient computation on bitcoin, 2025.

[LJF25] Sergio Demian Lerner, Martin Jonas, and Ariel Futoransky. Esspi: ecdsa/schnorr signed program input for bitvmx. 2025. eprint: 2503.02772. URL: https://arxiv.org/pdf/2503.02772.pdf.

[LP15] Yehuda Lindell and Benny Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. *Journal of Cryptology*, 28(2):312–350, 2015.

[Mer87] Ralph C Merkle. A digital signature based on a conventional encryption function. In *Conference on the theory and application of cryptographic techniques*, pages 369–378. Springer, 1987.

[Rub25] Jeremy Rubin. Delbrag. Manuscript, 2025.

[Yao82] Andrew C Yao. Protocols for secure computations. In *23rd annual symposium on foundations of computer science (sfcs 1982)*, pages 160–164. IEEE, 1982.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets. In *27th annual symposium on foundations of computer science (Sfcs 1986)*, pages 162–167. IEEE, 1986.