

Slashable Secrecy for Witness Encryption over Ethereum Finality

Yihsuan Wu

Waseda University

yihswanwu84@suou.waseda.jp

Abstract. Witness encryption over blockchain state aims to release a secret on a ledger condition without a key custodian. Validators with enough stake to construct a valid fork witness can obtain the secret although the condition never holds on the canonical chain. They finalize a private fork that satisfies the condition, and keeping the fork hidden withholds the conflicting votes that slashing needs. We propose slashable secrecy to address this gap with Ethereum’s native slashing rules, without an additional key-custody committee or separate collateral. In executions where the condition never holds on the finalized canonical chain through completion, an information advantage yields evidence of those votes against keys the adversary controlled, or a solution to a designated hard problem. We give a construction from a witness KEM and prove slashable secrecy conditionally on its extraction interface. We bound the implicated historical stake under changing validator weights. For an archived Ethereum mainnet state, the historical evidence-weight floor exceeds 3.5 percent of its active stake, about 1.44 million ETH, under stated budget and coverage assumptions. Experiments on a synthetic Ethereum network confirm native processing of the evidence, without WE decryption. Two limitations remain. A witness KEM satisfying the full relation and extraction interface remains uninstantiated. Enforcement requires that the evidence is available, converted to native form and included while its signers remain eligible.

Keywords: Witness encryption · Accountability · Conditional disclosure · Ethereum

1 Introduction

Conditional disclosure is useful when an application needs to publish a secret only after a future event. Consider a borrower who escrows an identity that may be disclosed if a debt remains unpaid at expiry. The lender should not hold a decryption key that permits disclosure after payment. Witness encryption offers another approach. The borrower encrypts the identity to a public condition, and anyone who later obtains a witness for that condition can decrypt [GGSW13, GG17, CGJ⁺17]. The borrower knows the identity at encryption time but need not know the future witness.

A blockchain can supply such witnesses, but a valid witness need not describe the history accepted by its users. A private fork may omit a payment recorded on the finalized prefix of the canonical chain. If that fork satisfies the encryption relation, it can release the identity even though no eligible state on that prefix satisfies the disclosure predicate. Ethereum already assigns responsibility for conflicting finality votes. But a fork used only as a decryption witness can stay hidden, and hidden votes escape that responsibility [leo22]. We ask whether this responsibility can also support conditional confidentiality.

Our answer is a conditional yes. *Slashable secrecy* connects an information attack to historical misconduct evidence against keys controlled by the attacker, the same double or surround votes that Ethereum already punishes. The connection uses an extractable witness KEM and Ethereum’s finality rules, and it introduces no committee holding shares of a decryption key. It holds in executions where the canonical disclosure condition never becomes satisfied through completion, and under an extraction interface that no witness KEM is yet shown to meet. Turning the evidence into an actual penalty also requires compatible native evidence, eligible signers and timely inclusion.

1.1 Our contributions

Slashable secrecy. We define slashable secrecy. In executions where the canonical disclosure condition never holds through completion, an adversary’s distinguishing advantage yields historical double-vote or surround-vote evidence against validator keys it controlled, or a designated hardness solution, with a quantitative success bound. We prove that a construction from an extractable witness KEM satisfies it, conditionally on the premises of Section 4. The extraction premise in Section 4.2 adapts extractable witness-KEM security [FHAS24] to bind extraction to the same ledger execution, require joint success with admissibility, and specify the quantitative loss. Extraction is a stipulated cryptographic interface, not an algorithm available to every public observer. The construction adds no committee, no validator duty and no separate collateral. T+1-eWEB proves an analogous leakage-to-conflict implication for a modified next-block chain with bit-signed commitments [Sta25]. Our analysis handles complete native certificates, a conflicting canonical certificate formed only at completion, dynamic evidence weight and native enforcement.

Economic and temporal bounds. We bound the historical stake weight of the implicated keys. The bound extends Gasper’s dynamic-set accounting [BHK⁺20] to per-key effective balances, native denominator and support deficits, and a later evidence measure. For an archived Ethereum mainnet anchor, a uniform budget certificate gives a floor exceeding 3.5 percent of anchor stake, about 1.44 million ETH, under full weight-state coverage. An example eligible horizon is 3500 epochs, about 15.6 days. This is the period in which a predicate state is eligible, not a guaranteed period without leakage. We then state when Ethereum’s own mechanism can enforce the evidence. It must be converted to native form and included while its signers remain eligible, and we distinguish evidence weight,

slashing eligibility and monetary loss. We formalize the withdrawal-delay and long-range-revision boundaries described by Casper [BG17], through native eligibility-retention conditions and a correctness-based limit on revoking recovery from archived ciphertexts.

Native validation. On a synthetic Ethereum network, we verify an execution-state disclosure witness and include the resulting evidence, which the pinned consensus rules process as native slashings. A second execution produces a late historical witness after the implicated keys lose eligibility. These experiments exercise native consensus and execution rules. They do not implement WE decryption. We release a reproducibility artifact containing the verification code, saved native histories and archived-state measurements.¹

1.2 Technical overview

Encryption fixes an authenticated anchor, a predicate and a bounded history-verification relation. A legal witness supplies a valid chain, an eligible state satisfying the predicate, and records verifying finalization of a descendant checkpoint. The witness KEM encapsulates a key using this public statement alone, and authenticated encryption protects the payload.

Security follows in three steps. The extraction interface first supplies a witness from an information attack. If the canonical disclosure condition remains unsatisfied through completion, the witness’s finalized checkpoint conflicts with the canonical completion checkpoint. Complete vote records then identify offending signers, and weight transport bounds their distinct-key intersection. An authorship premise excludes framing honest keys. Native enforcement is a subsequent step because historical aggregates, validator indices and slashing eligibility can differ at the processing state.

1.3 Related work

Hioki’s WE-based Bitcoin bridge proposal already connects Ethereum finality witnesses to slashing and discusses the danger of withholding the incriminating history [leo22]. An MEV-Boost proposal punishes unauthorized key-share release through a contract and, in a collusion case, exposes a proposer’s equivocation to native Ethereum slashing. It explicitly requires access to the private shares [jan22]. A later proposal ties decryption to slashable proposer commitments but leaves proposer–Keyper collusion unresolved [and25]. These proposals already connect confidentiality and economic responsibility.

Other release mechanisms include distributed custodians [BGG⁺20,CDK⁺22] and threshold beacons [GMR23]. Ferveo assigns stake-weighted threshold decryption to existing BFT validators [BO22], and weighted batched encryption improves committee-based release [ABD⁺26]. Silent-setup threshold encryption [GKPW24]

¹ <https://anonymous.4open.science/r/sswe-artifact-D613/>

and BEAST-MEV [BCF⁺25] avoid interactive setup, with the latter also supporting batched decryption. Our conditional construction derives evidence of violations of Ethereum’s existing finality-voting rules, without adding application-specific decryption duties to validators. The pinned Ethereum rules do not slash conflicting sync-committee messages [Eth].² Table 1 compares release mechanisms and accountability under each work’s stated model.

Goyal and Goyal extract a blockchain fork using WE under distinguishable forking security [GG17]. Our conditional result derives native misconduct evidence and its historical weight. KZG-opening WE [FHAS24], functional-commitment WE [CFK24] and linearly verifiable SNARKs [GHK⁺25] provide techniques, but no derivation of our native relation and joint interface is established here.

Threshold traitor tracing handles ciphertext-specific decoders under its admissibility and black-box tracing-access conditions [BPR24a,DDP⁺26]. CCA-secure traceable threshold encryption adds stronger privacy and consistency guarantees [BBF⁺25]. Snitching already relates distinguishing advantage to punishment probability with unframability [DFLM24,BDF⁺25]. Strong snitching covers trusted-party-assisted collusion under restricted interaction [BDF⁺25]. Neither quantitative leakage accountability nor economic deterrence is new. Our stipulated extraction access is not a black-box tracing guarantee or a public method for obtaining hidden evidence. Traceable sharing [BPR24b,GJP26], dealer-free traceable encryption [BHO25] and whistleblowing [KGP⁺25] provide further approaches to attribution under their respective models. Accountable decryption [LLW⁺24] also has a different system model.

BFT forensics [SWN⁺21], accountability gadgets [NTT22] and accountable light clients [CSSV22] study attribution in consensus and state verification. Gasper already bounds offending quorum weight under activations and exits [BHK⁺20, Section 8.6, Theorem 8.3]. Formal verification and weak-subjectivity analysis provide further dynamic-validator results [ALP⁺20,PA20]. Our accounting specializes these ideas to native weight tests and evolving effective balances. We use Casper’s accountable safety [BG17] while separately requiring an extraction interface that connects a confidentiality attack to a native history.

Services such as Shutter API³ and TACo⁴ already support conditional release or policy-controlled access. Nillion’s Covenants roadmap plans slashing for falsely released keys in its 2027 Neon phase.⁵ Such release and penalty mechanisms do not by themselves establish extraction of native consensus evidence from private information advantage. Our full native witness KEM also remains uninstantiated, so this comparison does not establish a deployable replacement for those services.

Table 1 shows what existing work leaves open. Committee-based release, as in eWEB [GKM⁺22], Cassiopeia [SZ23] and McFly [DHMW23], gives no punishable

² Etan Kissling, *EIP-7657: Sync committee slashings*, 2024, Stagnant. <https://eips.ethereum.org/EIPS/eip-7657>.

³ Shutter Network, *Shutter API: How It Works*, https://docs.shutter.network/docs/protocol/api/how_it_works. Accessed 27 September 2026.

⁴ *Introduction to TACo*, <https://docs.taco.build/>. Accessed 27 September 2026.

⁵ Nillion, *Roadmap*, <https://nillion.com/roadmap/>. Accessed 27 September 2026.

Table 1. Conditional release and accountability under the stated models. Evidence availability, punishable misconduct and implementation scope are distinct properties.

Work	Release mechanism	Evidence of misconduct	Collateral and enforcement	Established scope
eWEB [GKM ⁺ 22]	Blockchain shares with dynamic proactive secret sharing	Release-witness extraction, no misconduct proof guaranteed	No native Ethereum penalty established	Protocol and proof of concept; modified blockchain
Hioki bridge [leo22]	WE on Ethereum finality; witness block recorded on Bitcoin	Recorded witness data; no extraction from leakage	Native slashing of published fork signers; no derived bound	Informal proposal; ADP-based WE candidate
Cassiopeia [SZ23]	Committee releases on a witness, or timeout in the incentivized variant	Private off-chain disclosure is not contract-detectable	Incentivized variant slashes collateral for failed release	Security under committee assumptions. Base-protocol gas evaluation
McFly [DHMW23]	SWE using committee signatures on a future reference message	Honest-majority model; no punishable-evidence extraction for early decryption	No Ethereum-native slashing guarantee from release signatures alone	Formal SWE and protocol analysis; SWE benchmarks
T+1-eWEB [Sta25]	Modified signatures and KZG-opening WE for the next block’s state	Conflicting commitment-bit signatures in its model	Proposed penalty for conflicting commitment-bit signatures, not native attestations	Formal next-block construction; modified blockchain
Nihilium [ova26]	Staked processors release keys after validating client proofs	Observed leaked keys and challenges; detectability is open	Separate processor collateral in application contracts	Informal proposal; author-reported test implementations
This work	Conditional WKEM for native Ethereum history	Specified extraction yields controlled-key historical evidence or a hardness solution	Existing validator stake, subject to conversion, eligibility and inclusion	Conditional quantitative theorem; native tests; full WKEM uninstantiated

The protected events differ. McFly proves pre-release secrecy under its committee corruption assumptions. Our game excludes executions in which the canonical disclosure condition becomes satisfied by completion.

evidence when the committee decrypts early. Contract-based penalties depend on a leaked key or share being observed and on separate collateral [jan22,ova26]. The closest leakage-to-conflict result, T+1-eWEB, holds on a modified chain with modified signatures, not for native Ethereum attestations [Sta25]. Hioki’s proposal points to native slashing and addresses withholding by recording witness data on Bitcoin, but gives no security definition or proof for that connection and no derived stake bound [leo22]. None of the prior works in Table 1 derives evidence of a violation of Ethereum’s native slashing rules from leakage. Our goal is that derivation, with a definition, a conditional proof and a bound. An information advantage in an execution where the canonical disclosure condition never holds through completion should yield double-vote or surround-vote evidence against validator keys the adversary controlled, with a lower bound on their historical stake weight. The penalty mechanism is then Ethereum’s own, subject to evidence conversion, eligibility and inclusion, with no committee, no new validator duty and no separate collateral.

2 Preliminaries

Let λ be the security parameter. All polynomials, probabilities and advantages are functions of λ . PPT means probabilistic polynomial time in λ . An algorithm that takes an adversary as input is PPT if its running time is polynomial in λ and in the running time of that adversary. A function is negligible if it is eventually smaller than $1/q(\lambda)$ for every positive polynomial q . We write $\text{negl}(\lambda)$ for a negligible function, which may depend on the adversary. A polynomial-time relation verifier $R(x, w)$ checks a witness w for a public statement x . Its language is $L_R = \{x : \exists w, R(x, w) = 1\}$, with witness length bounded polynomially in the instance length.

2.1 Witness encryption and extractable witness KEMs

Witness encryption encrypts a message to a statement and permits recovery using a satisfying witness [GGSW13]. A witness key-encapsulation mechanism, or WKEM, encapsulates a symmetric key instead [FHAS24]. For a fixed relation R , its algorithms are

$$\text{pp} \leftarrow \text{Setup}(1^\lambda), \quad (\text{ct}_K, K) \leftarrow \text{Encap}(\text{pp}, x), \quad K' \leftarrow \text{Decap}(\text{pp}, x, \text{ct}_K, w).$$

Encapsulation needs no witness. Correctness requires legal witnesses for the same statement to recover the encapsulated key from the same ciphertext, except with negligible probability. Authenticated encryption under K then protects the message. Invalid-witness decapsulation need not return an explicit error.

Ordinary WE confidentiality protects false statements $x \notin L_R$. It does not ensure secrecy for a true statement whose witness is merely hard to obtain. Extractable security relates an information attack to recovery of a satisfying witness under a specified access model [FHAS24]. General extractability with

auxiliary input faces conditional impossibility results [GGHW17]. This matters here because a valid fork witness may exist even when no eligible state satisfying the disclosure predicate lies on the finalized prefix of the canonical chain. Section 4 states the precise quantitative interface required by our construction, including its permitted hardness-output branch. The extractor belongs to the security analysis, not the decryption API.

Signature-based WE uses signatures as witnesses; KZG-WE uses polynomial-commitment openings. These are particular relations and do not by themselves verify Ethereum transitions. We also use a binding commitment $\text{Com}(m; \rho)$ and a sound zero-knowledge proof of ciphertext consistency for adversarial escrow creation. Their roles and additional correctness requirements are specified in Section 3.3.

2.2 Ethereum finality and accountability

Slots index block opportunities, and epochs group slots for voting and state processing. A checkpoint is identified by its epoch and block root. Casper FFG votes link a source checkpoint to a target checkpoint. Starting from the protocol’s justified base, a link from a justified source supported by at least two thirds of the applicable active weight can justify its target. Finalization uses the native patterns of justified checkpoints and links. Double votes have the same target epoch and different attestation data. A surround vote strictly encloses the source and target epochs of another vote. Both can support attester slashing [BG17,Eth]. Accountable safety derives offending votes from conflicting finalizations. Our construction additionally needs a cryptographic route from an information attack to the private history containing those votes.

We identify checkpoints and states by authenticated ancestry and epoch, slot and processing phase. Skipped slots can leave two checkpoint epochs with the same block root. A root alone therefore does not identify the state used to evaluate a predicate or count stake. Historical evidence retains full signed data, signing domains and the branch’s index-to-public-key mapping. Keys are compared across branches by public key and counted once.

Liability. A validator remains eligible for a new native slashing while it is unslashed and its current epoch lies between activation and withdrawability. Exit does not immediately end eligibility. Once withdrawability is reached, even valid historical evidence cannot cause a new slashing of that key. A consolidation transfers balance but does not transfer responsibility for the source key’s signatures.

3 The slashable-secrecy protocol

The protocol encrypts to a ledger condition. Its decryption witness supplies a valid chain and the records needed to verify finalization. We first describe the statement and the user operations. Section 4 then analyzes leakage.

3.1 Model and disclosure condition

The public statement is $x = (\nu, a, \phi, H, \mathbf{B})$. Here ν fixes the native rules and encodings, a is an authenticated anchor at epoch e_0 , ϕ specifies the predicate Φ , H limits the eligible state epochs, and $\mathbf{B}$ bounds the verification data and work. The common trust context authenticates the anchor independently of WE setup. The canonical chain is the chain selected in the designated observer’s view under the fixed consensus rules and trust context. It may have an unfinalized head. Its finalized prefix consists of the ancestry of the observer’s accepted finalized checkpoint and grows monotonically in our execution model. We call this prefix the canonical history. A supplied chain is valid if it passes the prescribed anchor, signature, state-root and consensus and execution transition checks. Validity does not imply selection as the canonical chain [Eth]. The canonical disclosure condition holds when an eligible state satisfying Φ lies on the finalized prefix of the canonical chain.

Witness design. Write the witness as

$$w = (P_a, \mathcal{L}, \omega_u, i_u, i_f, i_t).$$

Here P_a supplies the anchor openings, execution context and necessary prefix provenance. The bounded history $\mathcal{L}$ contains the supplied chain, transition-validation material and original signed attestations. We use “history” for this larger package of blocks, states and supporting records. The indices select a disclosure state u , its descendant checkpoint f , and a replay event t that finalizes f . The opening ω_u authenticates the state data read by Φ . For an execution predicate, this includes the payload-header linkage to the execution state root and the relevant contract code and storage.

Opening relation. The deterministic verifier accepts $R_{\text{LC}}(x, w) = 1$ exactly when all of the following checks pass.

1. P_a authenticates the anchor and initial provenance. The witness respects the encodings and resource bounds fixed by $(\nu, \mathbf{B})$.
2. Exact replay of $\mathcal{L}$ from that anchor accepts every consensus and execution transition, including signatures and state roots.
3. The selected state satisfies $e_0 \leq \text{ep}(u) \leq e_0 + H$ and $\Phi(u) = 1$, as checked using ω_u against its authenticated state.
4. The selected event t natively finalizes f , with u on its ancestry. Replay retains complete signed support for every relevant justification record, including provenance carried from P_a .

Finalization may occur after the eligible state horizon, but all material must fit the verification bounds. An unchecked execution-engine response does not satisfy the second check. Appendix A.1 specifies the exact interfaces and certificate format.

This opening rule checks the supplied chain and its finalization records. It does not establish that the chain is canonical. A valid private fork may therefore

satisfy R_{LC} while the canonical disclosure condition remains unsatisfied. This distinction motivates the accountability analysis.

3.2 Encryption and disclosure

Fix a witness KEM for R_{LC} , or an exact encoding with polynomial-time witness compilation and projection. Its approved parameters $\mathbf{pp}$ bind this relation. The encryptor needs only x and m , not a future history or witness, and computes

$$(\mathbf{ct}_K, K) \leftarrow \text{Encap}(\mathbf{pp}, x), \quad \mathbf{ct} = (x, \mathbf{ct}_K, \text{AE.Enc}_K(m)).$$

We write $\text{Enc}(\mathbf{pp}, x, m)$ for this computation. The statement, ciphertext encodings and key derivation are bound consistently. Setup uses the backend’s prescribed distribution. A claim about malicious setup or revealed setup secrets would require additional backend security.

Once the canonical disclosure condition holds, anyone with a complete bounded canonical package can run `DeriveWit` and obtain w . The decryption algorithm $\text{Dec}(\mathbf{pp}, x, \mathbf{ct}, w)$ computes $K' \leftarrow \text{Decap}(\mathbf{pp}, x, \mathbf{ct}_K, w)$ and opens the authenticated payload under K' . Correctness must cover adaptively selected witnesses, as Definition 4 in Appendix A states. Decapsulation uses only $(\mathbf{pp}, x, \mathbf{ct}_K, w)$ and needs no approval from a key custodian. It assumes availability of the history and openings, not merely their public roots.

3.3 Identity escrow

For the running application, Φ checks that a fixed obligation has expired and remains unpaid. The contract code, obligation parameters and paid flag are authenticated by the witness. The application must also authenticate the escrowed identity and approve this condition. A well-formedness proof π_{wf} can bind an adversarial borrower’s ciphertext to the approved message commitment by proving

$$c_m = \text{Com}(m; \rho), \quad (\mathbf{ct}_K, K) = \text{Encap}(\mathbf{pp}, x; r), \quad c = \text{AE.Enc}_K(m; s).$$

Soundness, binding and correctness for every accepted choice of coins then imply that a legal witness recovers the committed message, except with negligible probability. This proof does not establish the identity’s authenticity or the future availability of a witness. Appendix A.3 gives the argument.

Timely compliance prevents disclosure only if its valid discharge reaches the history eventually finalized before expiry. A broadcast transaction or payment on a discarded branch is insufficient. This inclusion assumption belongs to the application, since the encryption predicate can test only the authenticated ledger state.

4 Security analysis

We now ask what happens when the canonical disclosure condition remains unsatisfied but an adversary nevertheless obtains information. Our result is conditional on extracting a chain witness and authenticating its signed support. The consensus argument then supplies the misconduct evidence and its weight.

4.1 Security goal

The goal is accountability for leakage, not secrecy against every adversary. Any information advantage gained while the canonical disclosure condition stays unsatisfied through completion must yield verifiable misconduct evidence against keys the adversary controlled, or a solution to a designated hard problem. A fixed execution descriptor $\mathcal{D}$ specifies the setup, observer, signing and corruption interfaces, extraction record, resource bounds and completion procedure. We write E for one execution of the experiment and $\text{Rec}(E)$ for its extraction record. The adversary chooses a statement with an authenticated anchor and equal-length messages m_0, m_1 , receives an honest encryption of m_b , and outputs a guess b' . The same execution then completes to a canonical finalized checkpoint c beyond $e_0 + H$. Completion must specify scheduling and corrupted-key behavior; it does not silently restore honest participation.

Let **bad** mean that an eligible Φ -satisfying state lies on the finalized prefix of the canonical chain at any time through completion. Thus **bad** records satisfaction of the canonical disclosure condition. Let **Valid** mean that the challenge is accepted and the guess is returned within the resource bounds. Let **Done** mean that completion reaches such a checkpoint c . Set $\text{Adm} = \text{Valid} \wedge \text{Done} \wedge \neg \text{bad}$. The game returns a fair bit on all other paths, giving the adversary's advantage

$$\epsilon = \Pr[b' = b \wedge \text{Adm}] - \frac{1}{2} \Pr[\text{Adm}].$$

The ever-controlled set $\mathcal{C}(E)$ is fixed by the key-generation and corruption interfaces in this execution. Merely funding or replaying an honest key's deposit does not add that key to the set.

For each execution the descriptor also fixes the historical evidence context a_E , the evidence-counting measure μ_e^E , the floor $\beta(E)$ and the hardness instance I_E with verifier HardVf . Slashable secrecy requires a PPT extractor, using $\text{Rec}(E)$, to return a solution z with $\text{HardVf}(I_E, z) = 1$ or historical evidence ev satisfying

$$\text{Adm}(E), \quad \text{HistVf}(a_E, \text{ev}) = (1, T), \quad \emptyset \neq T \subseteq \mathcal{C}(E), \quad \mu_e^E(T) \geq \beta(E),$$

with combined probability at least $\epsilon/p(\lambda) - \text{negl}(\lambda)$, for a fixed polynomial p . The evidence verifier checks the signed-data pairs and returns their deduplicated selected offending keys. The counting measure and floor are fixed independently of extractor outputs. The formal game and definition appear in Appendix A.

This target excludes leakage that is followed by satisfaction of the canonical disclosure condition by completion. Completion and uniform weight coverage are premises on permitted executions, not filters applied after observing the attack. Stalls or private histories outside that class receive no positive-floor guarantee. The extraction record is also stronger than public observation and may permit creation of fresh signatures for controlled keys. The definition promises verifiable attribution, not a public detector or proof that misconduct was causally necessary for decryption.

4.2 Cryptographic premises

The interface below adapts standalone witness-KEM extractability [FHAS24, Definition 16]. It requires extraction within the same completed execution and a joint admissible-success bound with the stated loss. The permitted record, execution class and hardness alternative also differ, so we do not assert a strict strengthening of the standalone definition. These are backend requirements.

Definition 1 (Real-execution joint extraction). *For the fixed relation and completed experiment, a single PPT algorithm $\text{JExt}(\mathcal{A}, \text{Rec}(E))$ returns a witness, a tagged designated hardness solution, or $\perp$. Let $\mathcal{G}$ and $\mathcal{H}$ be the disjoint valid-witness and valid-solution events. For every permitted PPT adversary with advantage ϵ , require*

$$\Pr[\mathcal{H}] + \Pr[\mathcal{G} \wedge \text{Adm}] \geq \epsilon/p(\lambda) - \text{negl}(\lambda),$$

for a fixed polynomial p with $p(\lambda) \geq 1$. The allowed record, hardness instance and public verifier are specified in advance. Extraction preserves the execution, statement, completion and corruption set.

The record excludes setup trapdoors, undisclosed honest signing secrets, honest encryption coins and the challenge key or bit. A backend in the algebraic group model [FKL18] and random-oracle model must justify its permitted representations and queries without revealing those secrets indirectly. Ordinary WE security or a marginal extraction guarantee in a different game does not by itself establish this joint bound. A backend needs an execution-preserving reduction that also establishes joint success with admissibility. Section 6.3 assesses candidate backends.

Definition 2 (Signed-data authorship). *For every PPT evidence producer with the permitted extraction access, the following event has negligible probability. The producer outputs a verifying aggregate that names a never-controlled key on signed data absent from that key’s honest log. Honest keys preserve anti-slashing state and never sign an offending pair.*

This premise must cover the actual registration and auxiliary-input interfaces. It is not inferred from the mere acceptance of a native deposit signature. Appendix B.4 gives a sufficient registered-signature reduction.

4.3 From a witness to responsibility

On an admissible execution, an extracted witness finalizes a checkpoint that conflicts with canonical completion. The theorem below bounds the weight of the signers who supported both sides.

For the weight calculation, let W_0 be anchor weight. Uniformly bound active-weight growth by $W \leq (1 + g)W_0$, anchor-relative loss by oW_0 , and the previous-target support removed at a native weight test by $\bar{\eta}W_0$. Weight transport gives

$$b_0 = W_0 \left[\frac{1}{3} - 2o - \frac{2}{3}g - 2\bar{\eta} \right]_+, \quad \beta = W_0 \left[\frac{1}{3} - 3o - \frac{2}{3}g - 2\bar{\eta} \right]_+. \quad (1)$$

The first bound is in the anchor measure; the second counts evidence under a later measure with the same loss budget. Appendix B proves the native certificate and general weight statements used below.

Theorem 1 (Conditional slashable secrecy). *Assume Definitions 1 and 2, exact interfaces and binding authenticated encodings for R_{LC} (Appendix A.1), complete replay certificates (Definition 3) for the witness and completion histories, and canonical completion in polynomial time. Suppose the weight budgets hold uniformly. That is, for every accepted witness and every pair of conflicting supporting links that Lemma 3 selects against completion, Lemma 4 gives anchor intersection at least $b_0(E) > 0$ and evidence weight at least $\beta(E)$. Both floors are fixed independently of extraction. Then the protocol satisfies slashable secrecy (Definition 5). Its extractor succeeds with probability at least $\epsilon/p(\lambda) - \text{negl}(\lambda)$, with the polynomial p of Definition 1.*

Proof. Forward the hardness branch of JExt. On its admissible witness branch, project the history by Lemma 1. By Lemma 5 its finalized checkpoint conflicts with completion, and Lemma 3 selects offending links. Keep the full groups covering their supporting-set intersection, deduplicating offending keys. Weight transport gives the two floors and authorship excludes never-controlled keys. This postprocessing is deterministic and runs in polynomial time by Lemmas 1 and 3. It loses only the authentication and authorship failures, which have negligible probability. Appendix B.3 supplies the details.

If the ever-controlled keys have anchor weight less than the uniform positive anchor floor $b_0(E)$, the evidence branch succeeds with at most negligible probability. Assume also that every PPT algorithm finds a designated hardness solution with at most negligible probability. Then $\epsilon/p(\lambda)$ is at most negligible. The advantage ϵ is therefore at most negligible, since p is a polynomial. This small-coalition consequence uses the anchor measure on both sides. A zero evidence-weight floor is permitted only when a positive anchor intersection still ensures a nonempty offending set.

5 Economic responsibility and time

5.1 Enforcing the evidence

Historical evidence identifies signed misconduct. To submit it natively, a compiler must preserve the signed data while mapping every aggregate participant to the enforcement registry and checking the signing domains and container bounds. A private-fork participant absent from that registry can prevent full-aggregate conversion. The aggregate cannot simply be split to remove it.

Appendix D proves conversion under these checks. An alternative uses archived individual signatures for anchor-weight keys on both sides, preserving the weight floor without registering unrelated fork-only participants. This requires signature availability in the witness and completion packages, as well as extraction for the augmented relation. Neither conversion route guarantees inclusion.

At processing, the native operation newly slashes exactly the eligible keys in its full intersection. Prior slashings, withdrawability or domain changes can prevent enforcement. Penalties also use effective balances at their assessment states. The initial nominal penalty is $\lfloor E/4096 \rfloor$ in Gwei; a later correlation assessment depends on the rolling slashing sum. Available balances cap debits, and rewards to proposers or whistleblowers affect net coalition loss. The exact formula appears in Appendix D.3. Thus historical evidence weight measures implicated stake rather than a guaranteed burn.

5.2 Duration and shared exposure

We formalize the liability boundary under the pinned native rules. Casper already identifies the historical-key risk after withdrawal in its discussion of long-range revisions [BG17, Section 4.1]. For keys active and unslashed at epoch e_* , exits and consolidations cannot end slashing eligibility through $e_* + 256$ under the pinned rules. If B_t denotes keys already slashed in the meantime, their remaining eligible reference weight is the original weight minus that of B_t . Lemma 12 proves this interval.

For archived ciphertexts, the corresponding recovery observation follows from encryption correctness. If later-acquired secrets allow an attacker to construct a legal witness for the original statement, correctness permits recovery from that original ciphertext. Publishing a new ciphertext does not revoke this computation. A stale-key attack must still construct the native history and satisfy its predicate, proposer schedule and certificate bound. Section 6 demonstrates this witness path after eligibility expires. The formal archived-ciphertext argument is in Proposition 3.

Liability is shared across disclosures supported by the same conflict. A legal fork omitting unpaid obligations' discharges may satisfy several compatible escrow statements, but cannot erase payments already in its prefix. Applications must aggregate the value exposed across such statements. Historical stake does not become a separate compensation fund for each ciphertext.

5.3 An archived mainnet anchor

We use a root-verified Fulu state at epoch 461568, slot 14770176, with $W_0 = 40,689,877$ ETH and 881,322 active keys. The era file, independent root check and decoder are pinned to consensus-specs commit `8c12cae` [era26,Eth]. Appendix C derives the following uniform budgets from the full state and the transition rules.

Parameter or result	Value
Accounting interval N	3600 epochs
Cumulative leak cap L	256 epochs per chain
Bootstrap corridor (m, M)	(0.90, 1.05)
Derived slashing budget $\bar{\sigma}$	1.2/4096
Growth and loss (g, o)	(0.0405212560, 0.0886009221)
Previous-target deficit $\bar{\eta}$	$50 \cdot 2048/W_0 \simeq 0.00251659645$
Anchor intersection b_0/W_0	0.1240841256
Historical evidence floor β/W_0	0.0354832035

The 50-key bound includes 42 newly assigned exits and at most 8 preassigned exits per epoch. The evidence floor is about 1.4438 million ETH in historical weight. Future slashing is allowed by the derived budget, rather than excluded by a zero-slashing assumption.

Application of this number requires compatible budgets for every selectable weight test and counting state, including prefix provenance and completion. The anchor’s prefix-cohort count does not itself prove prefix growth/loss coverage. With $H = 3500$, about 15.6 days, and full certification and completion by $e_0 + 3599$, the remaining 99 epochs include normal finality processing. This is not 99 additional stall epochs or a complete scheduling descriptor. The pinned weak-subjectivity function gives 3537 epochs, about 15.7 days [Bee,Eth].⁶ Membership in that interval does not replace these coverage conditions.

6 Validation and backend assessment

The reproducibility artifact, linked in footnote 1, provides the code and saved data for reproducing the numerical certificate, native witness verification, evidence inclusion and delayed-recovery experiment. It includes pinned dependencies, reproduction instructions and a SHA-256 manifest.

6.1 Native disclosure and evidence

We implement a `DeadlineObligation` contract and replay it with the pinned Electra minimal consensus program and Prague execution using `py-evm 0.12.1b1`⁷ and `Solidity 0.8.30`.⁸ The synthetic network has 96 validators with 2048 ETH each. The contract accepts an exact payment by the debtor before the deadline and permanently records it. Disclosure requires expiry with the paid flag unset. The code, parameters and storage opening are authenticated.

The anchor is at epoch 3, the eligible horizon is $H = 10$, and certificate processing is bounded through epoch 26. Replay checks native signatures and

⁶ Background documentation: Phase 0 Weak Subjectivity Guide and its Electra extension. The calculation uses the pinned revision cited in the text.

⁷ Ethereum Foundation, *py-evm v0.12.1-beta.1*, <https://github.com/ethereum/py-evm/releases/tag/v0.12.1-beta.1>.

⁸ Solidity Team, *Solidity 0.8.30 Release Announcement*, 7 May 2025, <https://www.soliditylang.org/blog/2025/05/07/solidity-0.8.30-release-announcement/>.

consensus transitions, EVM execution, state roots, receipts, withdrawals and system calls. Blobs are excluded and generated execution requests are empty. Public verification uses saved signatures without signing keys, but shares reference implementations with generation.

History	Witness	Finalized epoch
Canonical timely payment	Rejected	15
Private omitted payment	Accepted	9
Canonical nonpayment, separate run	Accepted	11

The conflicting histories implicate 32 keys. Archived individual signatures allow singleton evidence, included one operation per signed block between slots 137 and 177. Public replay confirms exactly 32 new slashings. Altered state roots, false storage claims and missing signature archives are rejected. The result validates one supplied schedule, not arbitrary-network inclusion. The mainnet budget certificate is not applied to this synthetic trace.

6.2 A late witness after retirement

A second scenario extends the paid history without early slashings. Operators of keys 0 through 63 exit. Only after all reach withdrawability does the generator model acquisition of their keys and construct a private unpaid history from the old prefix. The generator enforces this ordering; the signatures alone do not timestamp key acquisition.

The total exit charge is 131072 ETH. The minimal preset uses churn quotient 32, minimum balance churn 64 ETH and activation/exit cap 128 ETH. Replay confirms the 128 ETH cap at every charge. Requests begin at epoch 64, the first bucket is 69, the last exit is $69 + \lceil 131072/128 \rceil - 1 = 1092$, and release is at 1348. With eight slots per epoch and six seconds per slot, the wait from publication position 40 to release slot 10784 is

$$(10784 - 40) \cdot 6 = 64464 \text{ seconds} \simeq 0.7461 \text{ days.}$$

This is simulated protocol time, not a mainnet attack or measured decryption time.

After the common prefix, the paid branch uses voters $X \cup Z$ and the private branch $X \cup Y$, where X, Y, Z are consecutive groups of 32 keys. Both contain source-4, target-6 links with different target roots. A complete cross-history scan finds exactly the 32 attester offenders in X and four proposer offenders already in X . All 36 evidence objects pass static checks and fail native processing at release because eligibility has expired. The private witness finalizes epoch 9 and authenticates the unpaid state at slot 56.

Both experiments include actual execution-state verification and supplemental signature archives. Neither includes a WE ciphertext, decapsulation, identity authentication or π_{wf} . Late acquisition occurs after the bounded security experiment’s completion and therefore illustrates the archived-witness limitation, not a violation of Theorem 1.

6.3 Witness-encryption candidates

The AADP candidate [SRG⁺26] and the earlier determinant-program construction have been broken [EGRSS26,Jo26].

Constructions for KZG openings [FHAS24] do not provide a native chain witness. We do not establish that the separate arithmetic-circuit proposal [WZ25] meets our interface. Timings for other relations do not establish the cost of a secure native instance. The conditional theorem requires a backend with the exact relation and extraction interface, rather than a particular implementation. Constructing and proving a witness KEM for this relation and extraction interface remains the main cryptographic instantiation problem.

7 Conclusion and future work

Slashable secrecy connects information advantage while the canonical disclosure condition remains unsatisfied through completion to controlled-key misconduct under a specified witness-extraction interface. Native certificate analysis and weight transport quantify the historical responsibility. Conversion, eligibility and inclusion determine whether it can be enforced. The archived mainnet state gives a positive conditional weight floor, and synthetic executions validate both evidence processing and a late witness after liability expires.

The next task is a witness KEM for the full relation and access model. Other concrete tasks are to remove supplemental-signature availability, prove timely evidence inclusion, and connect the retirement experiment to actual WE decryption. Rolling escrow also needs consistency across updates and security for every archived ciphertext. A stronger future goal is punishable premature disclosure even when the canonical disclosure condition becomes satisfied later. That requires evidence of early access and an enforceable misconduct rule, since signatures on the eventual canonical history need not conflict.

A Formal model and native relation

A.1 Bounded native verification

The fixed suite ν contains the consensus and execution programs, network parameters, fork schedule, domains, canonical encodings and predicate evaluator. The authenticated anchor gives a slot-and-phase tagged consensus position, its linked execution parent state and context, and initial justification provenance. The trust context is common to the compared histories. Resource bounds $\mathbf{B}$ cover encoded data, slot transitions and verification work, with caps polynomial in λ . Missing inputs, overruns, overflow and failed native assertions reject.

AnchorExact verifies the anchor state and execution openings and the provenance of relevant initial justified checkpoints, bits and participation flags. This requires replay from a complete trusted provenance base or an equivalent verifier. **NativeExact** then performs all pinned slot, epoch and block transitions in native order, including skipped slots, signatures, registry and balance changes, participation, finality and execution. It validates parent linkage and post-state roots. Execution validity includes the matching engine inputs, transactions, EVM and system transitions, withdrawals, requests, gas and blob rules, and historical inputs. Stateless witnesses must authenticate every required read, write, code object and absence claim. Exactness means accepting precisely the bounded native-valid continuations with truthful records. Availability and observer import checks, where used, also require their authenticated inputs rather than Boolean attestations.

Definition 3 (Complete replay certificate). *For every relevant successful native quorum test, retain its source and target, original test state and denominator, distinct supporting keys, and full signed-data groups. Each counted key is attributed to the first included, signature-verified attestation responsible for its matching target flag. Groups retain full attestation data, aggregate signatures, domains and branch-specific index maps. Shifted justification bits and intermediate justified checkpoints retain their original test provenance. The authenticated initial prefix supplies the same material where needed.*

Native key uniqueness and sorted unique signer indices are checked. Repeated participants are counted once. Keeping the original test matters because a support set need not pass under a later registry’s weights.

An elementary historical evidence item retains two full groups and a selected subset of their key intersection. **HistVf** authenticates each branch mapping and signing domain, checks both full signatures and the double-vote or strict surround condition, and verifies the selected membership. On a bounded list of such items it returns the union of selected keys, rejecting empty unions. Historical verification does not require the branch mappings to equal the current enforcement registry.

For $x = (\nu, a, \phi, H, \mathbf{B})$, use the witness tuple $w = (P_a, \mathcal{L}, \omega_u, i_u, i_f, i_t)$ of Section 3.1. The anchor openings, execution context and prefix material form P_a ; the continuation and its validation material form $\mathcal{L}$; and ω_u supplies the predicate openings. The verifier runs the exact interfaces and checks that u is

an anchor descendant within the eligible horizon, event t actually finalizes a descendant f of u , and the authenticated state at u satisfies Φ . For an execution predicate, it checks the payload-header-to-state-root linkage as well as the state opening. This defines R_{LC} .

A binding failure is a pair of distinct inputs with the same authenticated encoding. The encodings are binding if every PPT algorithm outputs a binding failure with at most negligible probability. The proofs call such an event an authentication failure.

Lemma 1 (Projection and canonical completeness). *Suppose AnchorExact and NativeExact are exact and no binding failure occurs among the authenticated encodings.*

- (a) *Every accepted witness determines, in polynomial time, a native-valid history, an eligible state that satisfies Φ , and a finalized descendant of that state with complete signed support.*
- (b) *Once the canonical disclosure condition holds, DeriveWit constructs an accepted witness from every complete canonical package within the bounds, in deterministic polynomial time.*

Proof. Anchor acceptance fixes the initial state and provenance. Induction over exact replay fixes subsequent states, signature groups and finalization records. The final checks select actual positions and predicate values, giving projection. For construction, replay the available bounded package and enumerate triples (u, f, t) in fixed lexicographic order. Build the predicate openings by the stipulated deterministic procedure and return the first accepted encoding. Admissibility of the package supplies such a triple. There are polynomially many positions and every check is polynomially bounded, so this procedure runs in polynomial time. Binding failures are excluded by the hypothesis.

An exact Boolean encoding follows by unrolling the bounded verifier into AND/NOT gates, imposing $z(z-1) = 0$ on every wire, $z = uv$ on AND gates and $z = 1 - u$ on NOT gates, and binding public inputs and acceptance. An accepting run supplies an assignment; induction over gates projects any satisfying assignment back to a legal witness. This establishes relation equivalence, not WE extraction or practical circuit cost.

A.2 Completed execution and record

The descriptor $\mathcal{D}$ fixes the setup and auxiliary-input distributions, native interfaces and observer, polynomial resources, completion procedure and extraction record. For each execution E it also fixes the historical evidence context a_E , the evidence-counting measure μ_e^E , the floor $\beta(E)$, and the designated hardness instance I_E with its sampler and verifier HardVf . Completion and weight restrictions hold throughout the permitted class. The experiment is the following.

1. Sample setup and the authenticated initial history. Initialize honest signing state and logs. Give the adversary its declared inputs and initialize $\mathcal{C}$ with the adversarially generated and initially corrupted keys.

2. Run its permitted messages, transaction submissions, scheduling and adaptive corruptions. Corruption releases only the declared state and adds the key to C. Honest signers retain anti-slashing state.
3. Accept one challenge (x, m_0, m_1) with an authenticated anchor, allowed bounds and equal-length messages. Return a fresh honest encryption of m_b for uniform b , and continue until the guess b' . Set **Valid** if the challenge is accepted and b' is returned within the resource bounds. Invalid requests, missing outputs and resource overruns leave **Valid** unset and follow the neutral outcome below.
4. Run the prescribed completion on the same state to a canonical finalized checkpoint c beyond $e_0 + H$. No new corruption query is accepted. Scheduling, queued-message delivery and participation of already corrupted keys follow the descriptor, not an implicit honest reset. Record completion signatures and transitions. Set **Done** if completion reaches such a checkpoint c , with $\text{ep}(c) > e_0 + H$, within the resource bounds.
5. Set **bad** if an eligible satisfying state lies on the finalized prefix of the canonical chain by completion. It need not be named by a separate finalization update. Set $\text{Adm} = \text{Valid} \wedge \text{Done} \wedge \neg \text{bad}$. Return $[b' = b]$ on this event and a fresh fair bit otherwise. On aborted paths, unused bits are assigned arbitrarily and $\text{bad} = 0$ by convention.

This gives the unnormalized advantage in Section 4. The final corruption set, completion and evidence context all belong to this one execution.

The polynomial-size read-only record $\text{Rec}(E)$ includes the public challenge and completed execution history with the records required for verification. Any adversary coins, internal state, algebraic representations, oracle queries or postprocessing access must be expressly specified. It excludes setup trapdoors, undisclosed honest signing secrets, honest encryption randomness, K , m_b and b , and seeds reconstructing them. Recording the already chosen pair (m_0, m_1) is allowed. Algebraic metadata may not reveal an honest secret by expanding a signature as $\text{sk } H(m)$. Postprocessing preserves the same oracle and execution. The experiment may keep private logs for judging success without giving them to the extractor.

Definition 4 (Honest-encryptor correctness). *For every permitted adaptive PPT selector of a statement x and message m , and every subsequent PPT selector of a witness w , with honest $\text{ct} \leftarrow \text{Enc}(\text{pp}, x, m)$ as in Section 3.2,*

$$\Pr[R_{\text{LC}}(x, w) = 1 \wedge \text{Dec}(\text{pp}, x, \text{ct}, w) \neq m] \leq \text{negl}(\lambda).$$

*The probability is over the setup, encryption and selector coins. **DeriveWit** must construct a legal witness from every available canonical package satisfying the declared history and resource bounds.*

Definition 5 (Slashable secrecy). *A scheme has slashable secrecy for $(\mathcal{D}, \beta)$ if a single PPT extractor and a fixed polynomial p with $p(\lambda) \geq 1$ work for every permitted PPT adversary. Given the adversary's code and $\text{Rec}(E)$, the extractor returns $\perp$, **(hard, z)** or **(evidence, ev)**. The hard event requires $\text{HardVf}(I_E, z) = 1$.*

The evidence event requires, in the same execution,

$$\text{Adm}(E), \quad \text{HistVf}(a_E, \text{ev}) = (1, T), \quad \emptyset \neq T \subseteq \mathcal{C}(E), \quad \mu_e^E(T) \geq \beta(E).$$

Over experiment and extractor coins, the sum of these disjoint success probabilities is at least $\epsilon/p(\lambda) - \text{negl}(\lambda)$, where ϵ is the adversary's advantage in the experiment above. The descriptor fixes β , the counting context, hardness instance and verifier independently of extraction. The hard branch is not restricted to admissible runs. Rejected evidence and empty sets never succeed, even when $\beta = 0$.

A.3 Consistency of an adversarial escrow

For the proof relation in Section 3.3, assume statistical binding of the approved commitment, soundness of π_{wf} and backend correctness for every accepted encapsulation choice and legal decryption witness. Soundness supplies values satisfying the commitment, encapsulation and payload equations. Binding fixes their message to the approved m , backend correctness recovers their K , and symmetric correctness recovers m . No extraction of the well-formedness witness is needed for this implication. It holds except with the sum of the soundness and binding errors, which is negligible.

Average correctness for honest coins alone is insufficient here, because the encryptor chooses accepted coins. A seed commitment must therefore be accompanied by checks of the full equations. The main secrecy game uses honest encryption without this optional proof; adding it requires zero knowledge and extraction-record compatibility.

B Security proofs

B.1 Native finalization and conflicting links

Lemma 2 (Native finalizing structure). *For the four finalizing branches of the pinned native rule [Eth], a newly finalized checkpoint a of epoch e has a supporting link $a \rightarrow a^+$ with $\text{ep}(a^+) = e + k$, where $k \in \{1, 2\}$. If $k = 2$, a checkpoint at epoch $e + 1$ on the same branch is justified as well. A complete replay certificate retains its justification support.*

Proof. Write P and C for the previous and current justified checkpoints saved before the transition at epoch E . The native procedure first copies C into the previous field, shifts the justification bits, and tests previous- and current-target weights. Its four finalizing branches are shown below. These are the branches of the fixed specification, not new acceptance rules.

Tested bits and epoch condition	Finalized	Link target	Intermediate
$[1, 4]$ and $\text{ep}(P) + 3 = E$	P	$E - 1$	$E - 2$
$[1, 3]$ and $\text{ep}(P) + 2 = E$	P	$E - 1$	none
$[0, 3]$ and $\text{ep}(C) + 2 = E$	C	E	$E - 1$
$[0, 2]$ and $\text{ep}(C) + 1 = E$	C	E	none

A newly set bit zero has a quorum whose source is C . Bit one is either set by the new previous-target check, with source P , or shifted from the preceding transition's bit zero. In the shifted case its original source is again P , since that transition copied its old current checkpoint into the previous field. Higher bits preserve earlier successful justification records. Induction from the authenticated initial provenance proves the table. The matching-source and matching-target tests establish the link endpoints. Fulu inherits the relevant participation rule from Deneb; the proof does not impose the older timely-target inclusion-delay condition.

Lemma 3 (Conflicting finality yields offending links). *Two conflicting finalized checkpoints with complete certificates satisfying Lemma 2 yield, in polynomial time, two supporting links whose intersecting signers have double-vote or surround-vote evidence.*

Proof. Let a, b be conflicting finalized checkpoints with $e = \text{ep}(a) \leq \text{ep}(b)$. Take a complete justification path to b from the shared authenticated prefix. Every common prefix checkpoint at or above epoch e would lie after a on its branch and make b a descendant of a , a contradiction. Therefore the path crosses epoch e in a link $s \rightarrow t$ with $\text{ep}(s) < e \leq j = \text{ep}(t)$.

Let $a \rightarrow a^+$ be the finalizing link of gap k . If $j = e$, the justification of a supplies the other equal-target-epoch link. If $e < j < e + k$, integrality forces $k = 2$ and $j = e + 1$, where Lemma 2 supplies an intermediate justified checkpoint. If $j = e + k$, use $a \rightarrow a^+$. The targets in each equal-epoch comparison are different, since their equality would put a on b 's branch. If $j > e + k$, then

$$\text{ep}(s) < e < e + k < j,$$

so the b -side link surrounds $a \rightarrow a^+$. This covers every integer target epoch and retains strictness at both surround endpoints.

Each selected supporting set is covered by native signed-data groups. Select the relevant key subset within each group and deduplicate keys. Retain each full aggregate, including signers outside the selected subset. For every pair of groups whose selected-key intersection is nonempty, retain the two complete signed messages, their aggregates, and their authenticated branch mappings. Historical verification uses each mapping to recover the public keys. The union of the intersections equals the intersection of the selected supporting sets. In particular, if $S_A = \bigcup_i S_{A,i}$ and $S_B = \bigcup_j S_{B,j}$ are their partitions among retained full groups, then

$$S_A \cap S_B = \bigcup_{i,j} (S_{A,i} \cap S_{B,j}).$$

The weight bound applies to this union, not to each group pair. This construction does not split aggregate signatures and does not assert direct acceptance by a native enforcement state.

B.2 Weight transport

This argument adapts the dynamic-set bound in Gasper’s Theorem 8.3 [BHK⁺20], alongside related dynamic-set analyses [ALP⁺20, PA20]. The additional terms track changes in individual effective balances, native denominator and support deficits, and a later evidence-counting measure. Let μ_0 be the anchor’s nonnegative public-key weight measure, with $W_0 = \sum_k \mu_0(k) > 0$. Missing keys have weight zero. For a measure μ_j , write $W_j = \sum_k \mu_j(k)$ and

$$I_j = \frac{\sum_k [\mu_j(k) - \mu_0(k)]_+}{W_0}, \quad O_j = \frac{\sum_k [\mu_0(k) - \mu_j(k)]_+}{W_0}.$$

For each selected supporting set S_j , let $D_j > 0$ be the actual quorum denominator and suppose $\mu_j(S_j) \geq q_j D_j - r_j W_0$, with $q_j \in [0, 1]$ and $r_j \geq 0$. Define $\gamma_j = [W_j - D_j]_+ / W_0$ and $a_j = q_j O_j + (1 - q_j) I_j + q_j \gamma_j + r_j$.

Lemma 4 (Weight transport). *For $T = S_A \cap S_B$ and any evidence-counting measure μ_e ,*

$$\begin{aligned} \mu_0(T) &\geq W_0 [q_A + q_B - 1 - a_A - a_B]_+, \\ \mu_e(T) &\geq W_0 [q_A + q_B - 1 - a_A - a_B - O_e]_+. \end{aligned}$$

Proof. For any S , $\mu_0(S) \geq \mu_j(S) - I_j W_0$. Also $W_j / W_0 = 1 + I_j - O_j$ and $D_j \geq W_j - \gamma_j W_0$, so $\mu_0(S_j) \geq (q_j - a_j) W_0$. Inclusion-exclusion in μ_0 gives the first bound. The inequality $\mu_e(T) \geq \mu_0(T) - O_e W_0$ gives the second. Nonnegativity gives the positive parts.

For two-thirds quorums with equal measure and denominator, $I_A, I_B \leq i$ and $O_A, O_B, O_e \leq o$ give loss $D \leq \frac{7}{3}o + \frac{2}{3}i + r_A + r_B$. If $W_A, W_B \leq (1 + g)W_0$, one may choose $i = o + g$, giving

$$D \leq 3o + \frac{2}{3}g + r_A + r_B.$$

At a previous-target test in state U at epoch E , let S be the native supporting indices. They are previous-active, unslashed participants, and their weights are the effective balances stored in U . The threshold denominator is the current active total. Let ν_E be the ordinary current-active measure and define

$$S' = S \cap \text{Active}(E), \quad \eta = \frac{\sum_{k \in S \setminus \text{Active}(E)} E_k(U)}{W_0}.$$

Provided $\frac{2}{3}D$ exceeds the effective-balance increment used as the native minimum, a successful check implies actual support at least $\frac{2}{3}D$. Hence

$$\nu_E(S') \geq \frac{2}{3}W_E - \eta W_0.$$

All retained keys still support the same link. Applying Lemma 4 to the two restricted sets, with ordinary active measures, yields

$$\mu_e(S'_A \cap S'_B) \geq W_0 \left[\frac{1}{3} - 3o - \frac{2}{3}g - \eta_A - \eta_B \right]_+$$

under the stated uniform budgets. In very-low-weight states the native minimum can itself pass the threshold. Such states need a separate clamp deficit. The numerical certificate in Appendix C stays above that regime. Its exit-cohort bound supplies the previous-target deficits.

B.3 Composition and authorship

Lemma 5 (Maturation). *On an admissible execution, any accepted witness’s eligible state u conflicts with the canonical completion c . Its finalized descendant f therefore conflicts with c as well.*

Proof. If u and c were comparable, their epochs would force u to lie on the ancestry of c . Exact state authentication would then place the eligible Φ -satisfying state on the finalized prefix of the canonical chain, triggering **bad**. Conflicting branches cannot reconverge in the cryptographically bound history tree, so the descendant f also conflicts with c .

For the evidence branch of Theorem 1, apply Lemma 5 and Lemma 3, then restrict native supports as in the weight transport of Appendix B.2. Their positive anchor intersection is nonempty. The full aggregate pairs cover it as a deduplicated union, whose evidence weight is at least $\beta(E)$. A never-controlled key in that union would have signed an offending pair. Honest anti-slashing means at least one message is absent from its log, contradicting Definition 2 except with negligible probability.

Thus every event $\mathcal{G} \wedge \text{Adm}$ except the authentication and authorship exceptions gives successful evidence. Forwarding $\mathcal{H}$ unchanged and using Definition 1 proves the success bound without conditioning extraction on admissibility. The postprocessor changes neither completion nor the controlled set. The uniform bound is chosen before witness selection; a favorable extracted intersection cannot be used to set it afterward.

B.4 A sufficient authorship reduction

Consider a registered-signature game with one honest challenge key, a chosen-message signing oracle, and a registration interface for all other aggregate keys [RY07]. A forgery is a verifying full same-message aggregate containing the challenge key on a message absent from its signing-query log.

Lemma 6 (Registered-signature reduction). *Suppose there are at most $n_h(\lambda)$ honest key-creation positions, for a polynomial n_h . A single PPT simulator, the same for every position, embeds any selected position as the challenge key and simulates the permitted experiment and evidence producer. It preserves, within statistical error δ , the unnormalized joint subdistribution on runs where that key remains uncorrupted, including all signing logs, registrations, permitted records, and producer inputs and outputs. It may abort on target corruption, but makes no signing query absent from the target’s real honest log. Every other aggregate key meets the game’s registration requirements. If the game’s success probability is at most ϵ_{reg} for each resulting reduction, and ϵ_{enc} bounds any collision of full signed-data/domain encodings, aggregate authorship failure has probability at most*

$$n_h(\epsilon_{\text{reg}} + \delta) + \epsilon_{\text{enc}}.$$

Here failure means accepted full aggregate evidence naming a never-controlled key on signed data absent from its honest log. This bound is negligible when ϵ_{reg} , δ and ϵ_{enc} are negligible. It then gives Definition 2.

Proof. For each position j , let F_j be its never-controlled authorship-failure event. Embed that key and scan the producer's output for the first verifying aggregate containing it on a fresh message. Outside encoding collisions, this is a forgery satisfying every registration requirement. The simulator does not abort on F_j , so $\Pr[F_j] \leq \epsilon_{\text{reg}} + \delta$ outside that global exception. Union-bound over positions and add ϵ_{enc} . The reduction uses the full aggregate and extracts no individual signature.

Its registration and record-simulation hypotheses must be established for the chosen native suite. It does not make a signature-game forgery a designated same-run hardness solution of the WE game.

C Uniform native weight budgets

All amounts in this subsection are in ETH. Use the pinned Fulu rules without a protocol upgrade. Anchor-active records must satisfy the hysteresis band at the observation point. Every record not yet withdrawable must satisfy $E_k \leq c_k + 0.25$. Already-withdrawable records may have stale effective balances only if they have exited and have no pending unslashed outgoing consolidation. They cannot reactivate or transfer uncredited balances to active records. Let N cover every accounting epoch touched by a continuation, including partial endpoint epochs. Every selected weight test and evidence-counting state must lie in this covered interval. Prefix provenance predating the anchor needs its own authenticated budgets. Likewise, finality completion beyond the interval requires a longer certificate, even when the predicate window is shorter.

Write $c_k = \min(b_k, E_{\text{max},k})$ and $\delta_k = E_k - c_k$ at the anchor. The effective-balance update gives $-1.25 \leq \delta_k \leq 0.25$ at completed update phases. Define the following normalized anchor inputs:

$$\begin{aligned} A^+ &= W_0^{-1} \sum_{k \text{ active}} (0.25 - \delta_k), & A^- &= W_0^{-1} \sum_{k \text{ active}} (1.25 + \delta_k), \\ J_0 &= W_0^{-1} \sum_{k \text{ inactive}} (c_k + 0.25). \end{aligned}$$

Let X_0 count anchor weight of active keys already scheduled to exit within the interval, and U_0 the entire pending partial-withdrawal amount, both divided by W_0 . Let B_D, B_E, B_C be the deposit, exit and consolidation remaining credits in ETH. Let s_0 be the maximum inactivity score of anchor-active keys. On each allowed continuation, assume at most L leak epochs and at most σW_0 in nominal immediate and correlation slashing penalties. Nominal assessed penalties, rather than actual debits from depleted balances, also bound the associated slashing rewards. A zero anchor slashings vector does not establish $\sigma = 0$ for future

executions. Lemma 8 below replaces this premise by a derived positive budget when the covered interval contains no correlation-penalty assessment.

For $0 < m < 1 < M$, define the following envelopes, retaining the integer square root and the churn rounding of the specification:

$$C(M) = \lfloor \max(128, MW_0/65536) \rfloor, \quad c_* = \frac{64}{\lfloor \sqrt{10^9 m W_0} \rfloor},$$

$$a = \frac{B_D + 256N}{W_0}, \quad q(M) = \frac{B_E + B_C + (N+1)C(M)}{W_0},$$

$$r(M, m) = c_* N \left(\frac{54M^2}{64m} + \frac{108M}{448} + \frac{2M}{56} \right),$$

$$p(M, m) = \frac{42}{64} c_* MN,$$

$$A(s, L) = \frac{s^2}{24} + \frac{4}{3}sL + \frac{8}{3}L^2 + 2L,$$

$$\ell(M, s, L) = \frac{MA(s, L)}{2^{26}}.$$

Here a bounds churn-consuming deposits and q bounds new exit and partial withdrawal reservations. The terms r, p bound ordinary rewards and penalties. Put

$$\begin{aligned} g &= \Lambda^+ + J_0 + \frac{129}{128}a + r(M, m) + \sigma, \\ o &= X_0 + U_0 + q(M) + p(M, m) + \ell(M, s_0, L) + \sigma + \Lambda^-. \end{aligned} \tag{2}$$

Lemma 7 (Anchor budget certificate). *If $g < M - 1$ and $o < 1 - m$, then every covered active registry on every allowed continuation from that full anchor satisfies*

$$O \leq o, \quad W \leq (1 + g)W_0, \quad I \leq o + g.$$

The same bounds apply independently on the canonical continuation and each allowed private fork. They do not require that the encryptor predict the continuation.

Proof. At update endpoints, effective-balance loss for an anchor-active key is at most its capped-balance loss plus $1.25 + \delta_k$. Partition those keys into still-active keys, keys with pre-anchor scheduled exits and keys whose exits were scheduled later. Charge the second group to X_0 . For the third group, stop its balance-loss accounting at the last update before exit scheduling and charge the scheduled effective balance. This avoids charging both the full anchor balance and later decreases. The remaining capped-balance losses are partial withdrawals, ordinary penalties, leak and slashing. Full withdrawals and consolidation-source transfers occur after the relevant stopping point. Automatic above-cap sweeps do not decrease capped balance.

For either reservation counter, initial credit is consumed once and at most $N + 1$ new epoch buckets contribute within the interval. The activation-exit and

consolidation capacities together are bounded by $C(M)$, giving $q(M)$. Deposit-credit conservation gives a . These arguments retain initial reservations and pending withdrawals rather than treating the queues as empty.

For growth, assign every anchor record auxiliary credit $c_k + 0.25$. Increase it by churn-consuming deposits and consensus rewards, ignore penalty debits, and transfer credit with each processed consolidation. Credit upper-bounds stored effective balance of current and previous active keys. A consolidation source has exited before its transfer and has only one outgoing consolidation. Deposits to already withdrawn records that bypass churn cannot reactivate them or fund a later outgoing consolidation. Under the pinned ordering, an unslashed pending source is processed before that bypass applies. Each new active record needs at least 32 ETH of processed deposits and receives one additional 0.25 margin. Thus total auxiliary credit is at most $(1 + \Lambda^+ + J_0 + 129a/128 + r + \sigma)W_0$.

Temporarily assume active weight at least mW_0 and auxiliary credit at most MW_0 . The base-reward ratio is at most c_* . Flag rewards are bounded by $(54/64)c_*M^2W_0/m$, attestation proposer rewards by $(108/448)c_*MW_0$ and sync rewards including proposer shares by $(2/56)c_*MW_0$ per accounting epoch. The two-target factor in the attestation term covers current and previous target epochs. Ordinary penalties use source and target weight 40 and sync weight 2. This gives r, p without assuming the desired final weight bound.

For inactivity, dominate scores by $u \mapsto u + 4$ on leak epochs and $u \mapsto \max(u - 12, 0)$ otherwise, starting at s_0 . With potential $F(u) = u^2/24$, a non-leak step satisfies $u' \leq F(u) - F(u')$, while a leak step satisfies $u' + F(u') - F(u) = 4u/3 + 14/3$. Before leak step j , $u \leq s_0 + 4(j - 1)$. Summation gives score area at most $A(s_0, L)$, including recovery tails, and hence the debit bound ℓ .

If the temporary corridor first failed, all preceding reward and queue queries would obey these bounds. The growth potential and partitioned loss bound would then give $g < M - 1$ and $o < 1 - m$ at that operation, contradicting the first failure. Between update phases, use the last completed update for stored effective balances, rather than imposing the hysteresis band on every intermediate actual balance. Finally $I - O = W/W_0 - 1 \leq g$, proving the claims. Each bound is derived from the transition rules and the declared per-chain limits [Eth].

Lemma 8 (Initial-slashing budget). *Let the difference between any covered epoch and the anchor epoch be strictly less than 4096. Suppose no already-slashed anchor record has `withdrawable_epoch` - 4096 equal to the epoch of a covered `process_slashings` call. This includes the anchor epoch when its epoch processing remains pending. Put*

$$A_N = 1 + \Lambda^+ + J_0 + \frac{129}{128}a + r(M, m).$$

If $\bar{\sigma} \geq A_N/4095$ and the strict bootstrap inequalities hold with $\sigma = \bar{\sigma}$, the anchor certificate holds without a restriction on future slashing events. Their total nominal penalties are at most $\bar{\sigma}W_0$ on each covered continuation.

Proof. A fresh slashing at epoch s sets withdrawability to at least $s + 8192$. Its correlation assessment is therefore no earlier than $s + 4096$. The stated midpoint

check excludes assessments for older records as well. Only initial penalties occur in the interval.

Use a second copy of the auxiliary-credit accounting in Lemma 7. Retire a record's entire credit at its first slashing and retain that retired credit in a cumulative counter. Each charged effective balance is at most the credit retired at that call. A record can be slashed only once. It cannot transfer the retired credit to another eligible record by consolidation, since `process_pending consolidations` skips slashed sources. Conversely, a processed unslashed source reaches withdrawability by the next epoch, before another block can slash it; its transferred credit cannot also finance a later source charge. Later credits arriving at a retired record cannot create another charge there. Withdrawn funds deposited to another eligible record consume the deposit budget again; churn-bypassing deposits to a withdrawn record neither reactivate it nor supply a later outgoing consolidation. Switching to compounding credentials queues excess balance as deposits rather than releasing uncharged effective-balance credit. Thus internal transfers never finance two first-slashing charges. The initial margins, newly created records and ordinary rewards are bounded by $A_N W_0$, exactly as in the auxiliary-credit proof.

Let Z be the cumulative first-slashing effective balances divided by W_0 , and let σ_* be the resulting nominal penalties divided by W_0 . Slashing rewards in aggregate are no greater than those penalties under the pinned quotients. Consequently

$$Z \leq A_N + \sigma_*, \quad \sigma_* \leq Z/4096, \quad \sigma_* \leq A_N/4095.$$

These bounds hold at every prefix before any first corridor failure. Using $\bar{\sigma}$ in the strict bootstrap then excludes that failure, so the argument does not assume its desired weight or penalty bound. This bounds assessed amounts, not the number of newly slashed keys or the weight retaining fresh native eligibility.

Anchors with pending correlation assessments. For another anchor, let $\mathcal{S}_{\text{old}}$ contain its already-slashed records whose stored withdrawability minus 4096 is a covered assessment epoch, including pending anchor-epoch processing. This set is computable from the anchor records. The future penalty amount also depends on the then-current effective balances, active total and rolling slashing sum, which can include new slashings. The anchor slashing vector alone does not fix that amount. For any uniform assessment-balance bounds $\bar{E}_k$, the native multiplier and integer rounding give

$$\sigma_{\text{old}} = \frac{1}{W_0} \sum_{k \in \mathcal{S}_{\text{old}}} \bar{E}_k, \quad \bar{\sigma} \geq A_N/4095 + \sigma_{\text{old}}.$$

One may use $\bar{E}_k = 2048$ ETH under the pinned rules. Each old record is assessed at most once, its nominal correlation debit is at most its effective balance, and this debit creates no slashing reward. It therefore only reduces available credit in the preceding first-slashing argument. Add this allowance to the total penalty budget and recheck the strict bootstrap inequalities. This conservative extension does not preserve a previous anchor's numerical floor automatically.

For the archived anchor of Section 5.3, the measured inputs, rounded here only for display, are

$$\begin{aligned}(A^+, A^-, J_0) &= (0.0056497551, 0.0268394804, 0.0101225404), \\ (X_0, U_0) &= (0.0002705341, 0.0000021356), \\ (B_D, B_E, B_C) &= (1427.117479885, 250.42546, 204), \quad s_0 = 0.\end{aligned}$$

With $N = 3600$, $L = 256$, $m = 0.90$ and $M = 1.05$, the integer-Gwei inputs give $A_N/4095 \simeq 0.0002540240$. The full-record audit `window-anchor-audit.json` finds no old slashing midpoint in the closed interval $[461568, 465168]$, including the anchor epoch. This audit interval is deliberately larger than the 3600 accounting epochs used by the certificate. The balance audit finds one stale, already-withdrawable record, index 611789, with zero actual balance, no pending outgoing consolidation and 32 ETH stored effective balance. All other records satisfy the band. The exception meets the exclusion above and changes no budget input. Thus the midpoint premise of Lemma 8 holds for this window. The conservative choice $\bar{\sigma} = 1.2/4096 = 0.00029296875$ gives

$$g \simeq 0.0405212560, \quad o \simeq 0.0886009221.$$

Both strict bootstrap inequalities hold. The leak limit and the covered interval remain class conditions; future slashing is allowed within the derived budget.

C.1 Previous-target exit cohorts

Fix a replay state U immediately before a finality weight test at epoch $E > 0$. Define the cohort

$$\begin{aligned}X_E(U) &= \{k \mid \text{activation}_k \leq E - 1, \text{exit}_k = E\}, \\ x_E(U) &= W_0^{-1} \sum_{k \in X_E(U)} E_k(U).\end{aligned}$$

Here $E_k(U)$ is stored effective balance, including for validators that are no longer active. This is not the ordinary current-active measure.

Lemma 9 (Exit-cohort deficit). *For the previous-target supporting set S at this test,*

$$\eta = W_0^{-1} \sum_{k \in S \cap X_E(U)} E_k(U) \leq x_E(U).$$

In a replay without a protocol upgrade, a further conservative bound is $x_E(U) \leq n_E(U)E_{\max}/W_0$, where $n_E(U) = |X_E(U)|$ and $E_{\max}$ is the maximum stored effective balance allowed by the pinned rules. For the pinned Fulu mainnet rules $E_{\max} = 2048$ ETH.

Proof. Native activity is the interval $\text{activation}_k \leq e < \text{exit}_k$. A key active at $E - 1$ but not at E must therefore have exit epoch exactly E . Conversely, each member

of $X_E(U)$ was active at $E - 1$ and is inactive at E . The native previous-target set contains only previous-active, unslashed keys with the target participation flag. Consequently $S \setminus \text{Active}(E) = S \cap X_E(U)$. Substitution in the definition of η proves the equality, and nonnegative balances prove the two inequalities. The final cap follows from the effective-balance update rule and the two permitted withdrawal credential caps in the pinned specification [Eth].

The next two lemmas bound cohort size before applying the per-key cap. This avoids equating a scheduling charge with the later stored balance. All amounts in their proofs are in ETH. Retain the certified corridor $mW_0 \leq W \leq MW_0$, the leak cap L , and c_* from Appendix C. For zero initial inactivity scores on records whose exits are unscheduled, put

$$d_* = \frac{2}{64}c_*MW_0 + \frac{40}{64}c_*E_{\max} + \frac{E_{\max}4(L+1)}{2^{26}}.$$

The extra score increment covers an endpoint conservatively.

Lemma 10 (Minimum first-exit charge). *Suppose $d_* < 3/4$, anchor-active effective balances are at least 17 and satisfy the anchor band, and inactive records already scheduled for future activation have effective balance at least 32. Under the pinned rules, every first assignment of a finite exit epoch within the covered interval charges at least 16 ETH.*

Proof. Until exit is first assigned, a record cannot be a processed consolidation source, receive a full withdrawal, or incur a slashing penalty. Slashing calls `initiate_validator_exit` before its initial debit; correlation penalties concern records whose exits are already assigned. Partial withdrawals and a switch to compounding leave at least 32 ETH when they reduce balance. Automatic sweeps leave the applicable cap, which is at least 32. Deposits, incoming consolidations and rewards cannot decrease balance. It remains to bound ordinary and inactivity penalties between successive effective-balance updates.

Even if all sync positions repeat the same key, its total sync penalty over at most 32 block slots is bounded by the full participant penalty pool, $(2/64)c_*MW_0$. Source and target penalties contribute at most $(40/64)c_*E_{\max}$ at the epoch transition. Starting at score zero, the score is at most $4(L+1)$ under the cumulative leak cap; non-leak updates do not increase it. The Bellatrix denominator is 2^{26} . Integer divisions only decrease these three bounds. Thus their sum is at most d_* per update interval. New, never-active records start at score zero and receive no such penalties before becoming eligible.

At a completed update, stored balance $E \geq 17$ has actual balance at least $E - 0.25$. Before the next update, balance is therefore at least $\min\{E - 0.25, 32\} - d_* > 16$, including any balance-reducing operation that leaves the 32 ETH floor. Its updated effective balance is at least 16. Registry ejection checks precede effective-balance updates. If an update first produces 16, the next registry check assigns exit before any further effective-balance update. An intervening voluntary, forced or consolidation exit reads the same stored balance. If no exit was assigned

at a registry check, its stored effective balance exceeded 16. Induction establishes the claimed minimum. The anchor premises start this induction; newly activated records start at 32 or above.

Lemma 11 (Uniform exit-cohort size). *Let K_E, K_C bound the per-epoch exit and consolidation capacities, respectively, and their remaining credits at the anchor. Suppose each new exit assignment charges at least $b > 0$. Let $n_0(E)$ count anchor records already assigned exit epoch E and active at $E - 1$. Then every covered continuation satisfies*

$$n_E(U) \leq n_0(E) + \lfloor K_E/b \rfloor + \lfloor K_C/b \rfloor + 2.$$

Proof. Each finite exit assignment uses one of the two native cursors. A cursor never returns to an earlier epoch. Among requests assigned exactly E , at most one can advance it from an earlier epoch into E . Such a request may span several buckets, but the unused capacity in its last bucket is smaller than that bucket's capacity. All remaining requests assigned E consume the remaining credit without advancing. If E is reached by a time-based reset, its credit is the current capacity; if it is the anchor cursor epoch, use the stated initial-credit bound. In every case the remaining requests consume at most K_E or K_C . Capacity changes do not replenish an already-open bucket. A request advancing beyond E cannot contribute to its cohort. Each exit charge is at least b ; partial withdrawals sharing the exit cursor only consume credit. Add the two counts and the already assigned anchor records. An assigned finite exit epoch is not changed, and a source cannot request another consolidation after assignment. These arguments apply to every replayed continuation, not just to an observed queue trace.

For the archived anchor, `cohort-certificate-audit.json` verifies minimum active effective balance 27, zero inactivity scores on all records with unscheduled exits, and one already scheduled future activation with effective balance 1536. The parameters above give

$$d_* \leq 0.446521 + 0.000429 + 0.031373 = 0.478323 < 3/4.$$

The displayed component bounds are rounded upward. The corridor implies activation-exit capacity 256 and consolidation capacity at most $C(M) - 256 = 395$. Initial credits 250.42546 and 204 fit these envelopes. Lemma 10 supplies $b = 16$, so at most $17 + 25 = 42$ newly scheduled keys share an exit epoch. The same audit finds $\max_{e_0 \leq E \leq e_0 + 3599} n_0(E) = 8$, including the anchor epoch. Consequently

$$n_E \leq 50, \quad \bar{\eta} = \frac{50 \cdot 2048}{40,689,877} \simeq 0.00251659645.$$

For authenticated pre-anchor provenance, the inherited rules retain validator records and never revise an assigned finite exit epoch. Every historical member exiting at $E < e_0$ therefore remains in the anchor's corresponding cohort. The full anchor audit finds a maximum of 19 over all these historical cohorts. Since earlier effective-balance caps were no larger than 2048, this gives a smaller prefix deficit

bound than $\bar{\eta}$, without assuming a historical count of sixteen exits per epoch. It applies to the authenticated ancestor prefix, not to an unrelated historical fork. This membership argument does not extend the growth/loss certificate to pre-anchor weight measures.

Apply Lemma 7 over all required replay phases and suppose the exit-cohort bounds establish $\eta_A, \eta_B \leq \bar{\eta}$ uniformly for every selectable record. Lemma 4 then gives the two floors of (1). Substitution gives the numerical values reported in Section 5.3. The main theorem also requires its separate strictly positive anchor-intersection premise.

D Native evidence and archived ciphertexts

D.1 Full-aggregate conversion

A historical group $g = (d, \sigma, K, \delta)$ contains full attestation data, a verifying aggregate signature, distinct public keys and the original domain. In an enforcement state s with injective key map κ_s , define $I_s(K) = \text{sort}\{\kappa_s^{-1}(k) : k \in K\}$.

Proposition 1 (Static and stateful conversion). *Suppose every participant of two full groups occurs in s , the converted indices and data fit its native types, and native verification computes the original signing roots. Suppose their data satisfy the native double-vote or strict surround predicate, oriented as required, and $J = K_1 \cap K_2 \neq \emptyset$. Then*

$$A_j = \text{IndexedAttestation}(I_s(K_j), d_j, \sigma_j)$$

passes static verification. The resulting slashing operation has intersection J . At its actual processing state, let

$$L_s(J) = \{k \in J : \neg v_k.\text{slashed}, \quad v_k.\text{activation} \leq \text{ep}(s) < v_k.\text{withdrawable}\}.$$

Subject to the native transition's numeric and state preconditions, processing succeeds exactly when $L_s(J)$ is nonempty and newly slashes exactly that set.

Proof. Reindexing preserves the full public-key set and signing root. Indices are not signed fields of `AttestationData`, and aggregate verification is invariant under key permutation. Injectivity gives $I_s(K_1) \cap I_s(K_2) = I_s(J)$. After checking signatures and slashable data, the native operation iterates over this intersection, calling `slash_validator` exactly on eligible indices. Processing one index changes no other index's activation, withdrawability or slashed flag. The final `slashed.any` assertion proves the equivalence.

The pinned domain function retains current and previous fork versions, not an arbitrary historical schedule. Domain/root equality must therefore be checked for each group. Rewriting a legacy signed field changes its meaning and is not a conversion. The pinned Fulu indexed-attestation limit is $2048 \cdot 64$ indices, while a block permits one attester slashing. These bounds are distinct.

Historical evidence may select $T \subseteq J$, but a native operation has no selection mask. Exact preservation of newly slashed keys requires $L_s(J) = T$. Full-intersection enlargement remains non-framing under the same authorship premise, since each added key appears in the same offending pair. For a batch, simulate each actual pre-state, omit operations with no eligible intersection and verify that the union of newly slashed keys is the desired set.

If an aggregate participant is absent from the enforcement registry, full-participant reindexing is unavailable. Deleting its index while retaining the aggregate is not justified by the original verification. This is a limitation of this compiler, not an impossibility result for compilers supplied with other signatures.

D.2 Conversion using individual signatures

Define R_{LC}^{sig} by augmenting the base witness with verified individual signatures for every anchor-positive-weight key in each retained full-data group, including prefix provenance. Completion supplies the same augmentation. Each signature covers the original full data and domain, and original aggregates remain checked. The experiments store all participants' individual signatures, which suffices. An aggregate alone does not supply this archive.

Proposition 2 (Singleton conversion). *Apply Theorem 1 to the augmented relation and record. Let $K_0 = \{k : \mu_0(k) > 0\}$ and $T_0 = S_A \cap S_B \cap K_0$. The weight bounds still hold for T_0 . If each selected key occurs in the enforcement registry with compatible signed-data types and domains, the archives compile to singleton native operations naming exactly T_0 . If each key is eligible at its operation's processing state and native preconditions hold, the operations newly slash exactly T_0 .*

Proof. Removing keys outside K_0 changes no anchor weight, and

$$\mu_e(T_0) \geq \mu_0(T_0) - O_e W_0 = \mu_0(S_A \cap S_B) - O_e W_0.$$

For each remaining key, use its archived signatures for the offending full-data pair in two singleton indexed attestations. The preceding proposition applies to their singleton intersection. Emit each key once.

This route avoids registration of unrelated fork-only participants. It requires available signatures on both sides and joint extraction for the augmented relation, not merely for the base relation.

D.3 Eligibility and monetary debits

The following lemma formalizes retention under the pinned withdrawal and consolidation rules. It is not a new long-range defense.

Lemma 12 (Retention of eligibility). *Let T be active and unslashed at epoch e_* , with fixed nonnegative reference weights μ_* . Let B_t contain its subsequently slashed keys and R_t those reaching withdrawability by processing epoch t . Under the pinned rules without an upgrade, the eligible reference weight is $\mu_*(T \setminus (B_t \cup R_t))$. For $e_* \leq t \leq e_* + 256$, $R_t = \emptyset$.*

Proof. Activation epochs do not change. Activity at e_* implies exit no earlier than $e_* + 1$. Ordinary and consolidation-source exits set withdrawability to exit plus 256; slashing only extends it. Thus withdrawability is at least $e_* + 257$. Consolidation retains the source record and signed identity. The native eligibility predicate now gives both claims.

This concerns reference weights, not unchanged balances. A small initial-penalty budget can coexist with many already-slashed keys, so it does not ensure that $\mu_*(B_t)$ is small.

Write balances in Gwei and $\Delta = 10^9$. For a key newly slashed at s_k , the pinned nominal initial penalty and extended withdrawability are

$$P_k^{\text{init}} = \lfloor E_k(s_k)/4096 \rfloor, \quad w_k = \max\{w_k^{\text{exit}}, s_k + 8192\}.$$

Here w_k^{exit} is the value after initiating exit. Correlation assessment occurs at $t_k = w_k - 4096$. With the then-current $S = \sum \text{slashings}$ and active total W , its nominal amount is

$$P_k^{\text{corr}} = \left\lfloor \frac{E_k(t_k)}{\Delta} \right\rfloor \left\lfloor \frac{\min\{3S, W\}}{\lfloor W/\Delta \rfloor} \right\rfloor.$$

Each actual debit is capped by the balance at that call. Initial processing credits total whistleblower/proposer rewards of $\lfloor E_k(s_k)/4096 \rfloor$, with the proposer share included in that total. Net coalition loss depends on the recipients. A historical weight cannot replace either assessment balance without a transport bound [Eth].

D.4 Recovery from an archived ciphertext

The proposition is a correctness consequence for immutable ciphertexts.

Proposition 3 (Archived recovery). *Consider a family of polynomially many immutable published ciphertexts. Suppose a PPT algorithm using later-available secrets supplies accepted witnesses for their original statements except with negligible probability. If correctness covers that adaptive witness selection, running the original recovery algorithm on those witnesses recovers the payloads except with negligible probability. New ciphertexts do not revoke this computation.*

Proof. Apply correctness to each synthesized witness and union-bound over the polynomial family and synthesis failures. The computation uses only the original ciphertexts and statements.

For stale Ethereum keys, the premise requires an actual bounded continuation with valid proposer slots, included votes, native queued changes and a satisfied predicate. A stake fraction alone specifies no such schedule. The retirement execution in Section 6 supplies one synthetic witness-producing schedule after eligibility expires.

References

- ABD⁺26. Amit Agarwal, Kushal Babel, Sourav Das, Babak Poorebrahim Gilkalaye, Arup Mondal, Benny Pinkas, Peter Rindal, and Aayush Yadav. Weighted batched threshold encryption with applications to mempool privacy. In *2026 IEEE Symposium on Security and Privacy (SP)*, pages 2462–2481. IEEE, 2026.
- ALP⁺20. Musab A. Alturki, Elaine Li, Daejun Park, Brandon Moore, Karl Palm-skog, Lucas Peña, and Grigore Roşu. Verifying Gasper with dynamic validator sets in Coq. Technical report, Runtime Verification, Inc., June 8, 2020. Sections 4 and 6. <https://github.com/runtimeverification/beacon-chain-verification/blob/master/casper/report/report.pdf>.
- and25. and882. An anti-collusion mechanism for threshold encrypted mempools. Ethereum Research, 2025. Research proposal, 10 April. <https://ethresear.ch/t/an-anti-collusion-mechanism-for-threshold-encrypted-mempools/22129>.
- BBF⁺25. Rishiraj Bhattacharyya, Jan Bormet, Sebastian Faust, Pratyay Mukherjee, and Hussien Othman. CCA-secure traceable threshold (ID-based) encryption and application. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*, pages 2324–2338, 2025.
- BCF⁺25. Jan Bormet, Arka Rai Choudhuri, Sebastian Faust, Sanjam Garg, Hussien Othman, Guru-Vamsi Policharla, Ziyang Qu, and Mingyuan Wang. BEAST-MEV: Batched threshold encryption with silent setup for MEV prevention. *Cryptology ePrint Archive*, 2025.
- BDF⁺25. Jan Bormet, Stefan Dziembowski, Sebastian Faust, Tomasz Lizurej, and Marcin Mielniczuk. Strong secret sharing with snitching. In *Annual International Cryptology Conference*, pages 295–327. Springer, 2025.
- Bee. Carl Beekhuizen. Electra weak subjectivity calculations. https://notes.ethereum.org/@CarlBeek/electra_weak_subjectivity. Undated note, accessed September 2026.
- BG17. Vitalik Buterin and Virgil Griffith. Casper the friendly finality gadget. *arXiv preprint arXiv:1710.09437*, 2017.
- BGG⁺20. Fabrice Benhamouda, Craig Gentry, Sergey Gorbunov, Shai Halevi, Hugo Krawczyk, Chengyu Lin, Tal Rabin, and Leonid Reyzin. Can a public blockchain keep a secret? In *Theory of Cryptography Conference*, pages 260–290. Springer, 2020.
- BHK⁺20. Vitalik Buterin, Diego Hernandez, Thor Kamphefner, Khiem Pham, Zhi Qiao, Danny Ryan, Juhyeok Sin, Ying Wang, and Yan X Zhang. Combining GHOST and Casper. *arXiv preprint arXiv:2003.03052*, 2020.
- BHO25. Jan Bormet, Jonas Hofmann, and Hussien Othman. Traceable threshold encryption without a trusted dealer. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 471–505. Springer, 2025.
- BO22. Joseph Bebel and Dev Ojha. Ferveo: Threshold decryption for mempool privacy in BFT networks. *Cryptology ePrint Archive*, 2022.
- BPR24a. Dan Boneh, Aditi Partap, and Lior Rotem. Accountability for misbehavior in threshold decryption via threshold traitor tracing. In *Annual International Cryptology Conference*, pages 317–351. Springer, 2024.
- BPR24b. Dan Boneh, Aditi Partap, and Lior Rotem. Traceable secret sharing: strong security and efficient constructions. In *Annual International Cryptology Conference*, pages 221–256. Springer, 2024.

- CDK⁺22. Matteo Campanelli, Bernardo David, Hamidreza Khoshakhlagh, Anders Konring, and Jesper Buus Nielsen. Encryption to the future: a paradigm for sending secret messages to future (anonymous) committees. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 151–180. Springer, 2022.
- CFK24. Matteo Campanelli, Dario Fiore, and Hamidreza Khoshakhlagh. Witness encryption for succinct functional commitments and applications. In *IACR International Conference on Public-Key Cryptography*, pages 132–167. Springer, 2024.
- CGJ⁺17. Arka Rai Choudhuri, Matthew Green, Abhishek Jain, Gabriel Kaptchuk, and Ian Miers. Fairness in an unfair world: Fair multiparty computation from public bulletin boards. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 719–728, 2017. ePrint 2017/1091.
- CSSV22. Oana Ciobotaru, Fatemeh Shirazi, Alistair Stewart, and Sergey Vasilyev. Accountable light client systems for proof-of-stake blockchains. *Cryptology ePrint Archive*, 2022.
- DDP⁺26. Sourav Das, Pratish Datta, Aditi Partap, Swagata Sasmal, and Mark Zhandry. Optimal threshold traitor tracing. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 486–516. Springer, 2026.
- DFLM24. Stefan Dziembowski, Sebastian Faust, Tomasz Lazurek, and Marcin Mielniczuk. Secret sharing with snitching. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 840–853, 2024.
- DHMW23. Nico Döttling, Lucjan Hanzlik, Bernardo Magri, and Stella Wognig. McFly: verifiable encryption to the future made practical. In *International Conference on Financial Cryptography and Data Security*, pages 252–269. Springer, 2023.
- EGRSS26. Muhammad El Gebali, Yaroslav Rebenko, Markus Schofnegger, and Lev Soukhanov. Breaking ADP-based witness encryption. *Cryptology ePrint Archive*, 2026. Paper 2026/1594.
- era26. Ethereum mainnet state at epoch 461,568. Nimbus era archive, <https://mainnet.era.nimbus.team/>, file `mainnet-01803-86416acc.era`. Verification metadata are recorded in the artifact, 2026.
- Eth. Ethereum Foundation. Ethereum consensus specifications. Pinned source revision. Commit 8c12cae, including the Fulu fork. The artifact records the full hash.
- FHAS24. Nils Fleischhacker, Mathias Hall-Andersen, and Mark Simkin. Extractable witness encryption for KZG commitments and efficient laconic OT. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 423–453. Springer, 2024.
- FKL18. Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In *Annual International Cryptology Conference*, pages 33–62. Springer, 2018.
- GG17. Rishab Goyal and Vipul Goyal. Overcoming cryptographic impossibility results using blockchains. In *Theory of Cryptography Conference*, pages 529–561. Springer, 2017. ePrint 2017/935.
- GGHW17. Sanjam Garg, Craig Gentry, Shai Halevi, and Daniel Wichs. On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. *Algorithmica*, 79(4):1353–1373, 2017.

- GGSW13. Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*, pages 467–476, 2013.
- GHK⁺25. Sanjam Garg, Mohammad Hajiabadi, Dimitris Kolonelos, Abhiram Kothapalli, and Guru Vamsi Policharla. A framework for witness encryption from linearly verifiable SNARKs and applications. In *Annual International Cryptology Conference*, pages 504–539. Springer, 2025.
- GJP26. Vipul Goyal, Abhishek Jain, and Aditi Partap. Traceable secret sharing revisited. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 33–63. Springer, 2026.
- GKM⁺22. Vipul Goyal, Abhiram Kothapalli, Elisaweta Masserova, Bryan Parno, and Yifan Song. Storing and retrieving secrets on a blockchain. In *IACR International Conference on Public-Key Cryptography*, pages 252–282. Springer, 2022.
- GKPW24. Sanjam Garg, Dimitris Kolonelos, Guru-Vamsi Policharla, and Mingyuan Wang. Threshold encryption with silent setup. In *Annual International Cryptology Conference*, pages 352–386. Springer, 2024.
- GMR23. Nicolas Gailly, Kelsey Melissaris, and Yolan Romainier. tlock: practical timelock encryption from threshold BLS. *Cryptology ePrint Archive*, 2023.
- jan22. jannikluhn. Removing trusted relays in MEV-Boost using threshold encryption. Ethereum Research, 2022. Research proposal, 19 August. <https://ethresear.ch/t/removing-trusted-relays-in-mev-boost-using-threshold-encryption/13449>.
- Jo26. Sunghyeon Jo. Cryptanalysis of a candidate witness encryption scheme for affine determinant programs. *Cryptology ePrint Archive*, Paper 2026/1583, 2026.
- KGP⁺25. Mahimna Kelkar, Aadityan Ganesh, Aditi Partap, Joseph Bonneau, and S Matthew Weinberg. Breaking Omertà: On threshold cryptography, smart collusion, and whistleblowing. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*, pages 3505–3519, 2025.
- leo22. leohio (Leona Hioki). Trustless Bitcoin bridge creation with witness encryption. Ethereum Research, February 6, 2022. <https://ethresear.ch/t/11953>.
- LLW⁺24. Rujia Li, Yuanzhao Li, Qin Wang, Sisi Duan, Qi Wang, and Mark Ryan. Accountable decryption made formal and practical. *IEEE Transactions on Information Forensics and Security*, 20:620–635, 2024.
- NTT22. Joachim Neu, Ertem Nusret Tas, and David Tse. The availability-accountability dilemma and its resolution via accountability gadgets. In *International Conference on Financial Cryptography and Data Security*, pages 541–559. Springer, 2022.
- ova26. ovanwijk. Slashable conditional key release: A deployable crypto-economic approximation of witness encryption. Ethereum Research. <https://ethresear.ch/t/24336>, 2026. March 10. Informal proposal for Nihilium.
- PA20. Daejun Park and Aditya Asgaonkar. Analysis on weak subjectivity in Ethereum 2.0. Technical report, Runtime Verification, Inc. and Ethereum Research, December 7, 2020. <https://github.com/runtimeverification/beacon-chain-verification/blob/master/weak-subjectivity/weak-subjectivity-analysis.pdf>.

- RY07.

Thomas Ristenpart and Scott Yilek. The power of proofs-of-possession: Securing multiparty signatures against rogue-key attacks. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 228–245. Springer, 2007.
- SRG⁺26.

Lev Soukhanov, Yaroslav Rebenko, Muhammad El Gebali, Mikhail Komarov, Handan Kilinc-Alper, Michel Abdalla, and Patrick Towa. Implementable witness encryption from arithmetic affine determinant programs. *Cryptology ePrint Archive*, Paper 2026/175, 2026.
- Sta25.

Lev Stambler. Achieving blockchain-secured cryptographic primitives from signature-based witness encryption. *Cryptology ePrint Archive*, 2025.
- SWN⁺21.

Peiyao Sheng, Gerui Wang, Kartik Nayak, Sreeram Kannan, and Pramod Viswanath. BFT protocol forensics. In *Proceedings of the 2021 ACM SIGSAC conference on computer and communications security*, pages 1722–1743, 2021.
- SZ23.

Schwinn Saereesitthipitak and Dionysis Zindros. Cassiopeia: Practical on-chain witness encryption. In *International Conference on Financial Cryptography and Data Security*, pages 385–404. Springer, 2023.
- WZ25.

Yuzhu Wang and Mingwu Zhang. Extractable witness encryption for arithmetic circuit satisfiability problem. *IEEE Internet of Things Journal*, 12(11):18137–18148, 2025.