Sequence-Level Security for Active Weighted Signature Reconfiguration

Sunghyeon Jo
sjo65@gatech.edu
Georgia Institute of Technology

May 20, 2026

Abstract

Active weighted threshold signatures support dynamic changes to signer weights, thresholds, and committee membership. We show that local validity of weighted update operations is not a compositional security abstraction: a sequence of individually valid updates can move an initially sub-threshold coalition into an authorized reachable state. We introduce rank-exposure guards, a compiler that enforces a reconstruction-safety invariant over live, stale, derivative, public, and transient signing material. The compiler wraps ledger-sound one-step update engines with atomic activation and old-epoch digest-bound transition certificates, lifting fixed-state weighted unforgeability and update soundness to sequence-level active unforgeability. We instantiate the compiler as REG-ADAPT, a guarded GLI reconfiguration scheme built around ADAPT-style local updates, and implement it on top of the public ADAPT Go artifact. Our evaluation shows that the artifact detects and rejects unsafe update sequences, while adding only microsecond-scale metadata and rank-audit overhead.

1 Introduction

Threshold signatures reduce the communication and verification cost of distributed authorization. In proof-of-stake systems, bridge committees, oracle networks, custody systems, and off-chain voting protocols, signer weights represent stake, reputation, token ownership, or policy authority. These weights change over time. Validators join and exit, delegations move, slashing changes effective stake, custody policies rotate signers, and bridge or oracle committees reconfigure by epoch. A useful weighted threshold-signature system needs efficient signing and secure reconfiguration.

Recent work on Active Weighted Signatures and the ADAPT framework gives an efficient local update engine for weighted FROST-style signing using generalized Lagrange interpolation [22]. A signer of weight w_i is represented by derivative shares

$$f(i),\ f'(i), \ldots,\ f^{(w_i-1)}(i)$$

at one evaluation point, instead of by w_i virtual signers. This gives an appealing way to change weights and thresholds without full re-setup. Active weighted reconfiguration also needs a compositional state-machine security layer: a sequence of locally valid weight and threshold updates can move the system into a reachable state in which a previously sub-threshold coalition is authorized. This identifies the right security object: the reachable transition system, not an isolated final-state signature instance or a list of locally valid update calls.

This paper develops that state-machine security layer and instantiates it around the public ADAPT artifact. The security invariant is enforced over the reachable state machine, including live,

stale, derivative, public, and transient signing material; local update correctness is one component of this invariant, not the invariant itself.

In a static Shamir-style weighted threshold signature, the adversary's power is often summarized by current corrupted weight. In a dynamic weighted system, the effective power is the rank accumulated across old shares from prior epochs, derivative shares from GLI weight assignments, public coefficient information released during threshold decrease, and temporary handoff data generated while two states coexist. These objects are all linear information about the secret-sharing state. The right accounting unit is reconstruction-relevant rank. We introduce a public rank-exposure ledger whose entries are row descriptors tagged by owner, source, and expiry condition. A row interpretation map turns live descriptors into linear rows. Reconstruction safety means that the target secret row is not in the span of the rows available to the adversary. Exposure rank is the computable measure used to enforce this reconstruction condition in Shamir/GLI states. A public rank-exposure envelope upper-bounds the exposure rank of every coalition allowed by the system's environmental or economic corruption assumption. A reconfiguration is valid only if this envelope remains below threshold in the activated next state and in every transient state created by the update transcript. An update is ledger-sound if every scalar value learned during the update is recorded in the exposure ledger or is linearly dependent on recorded rows.

The guard compiler is a wrapper for practical systems that want a stable public key while changing the authorization policy. Examples include PoS state proofs, sync committees, bridge or oracle committees, and threshold custody with changing user policies. In all of these systems, there is a gap between the moment a participant's current weight changes and the moment all old signing material becomes irrelevant. The rank-exposure ledger makes that gap explicit. For PoS, ledger expiry can be tied to unbonding periods, slashing windows, and finalized refresh transcripts. For custody or governance, it can be tied to policy activation delays and key-rotation windows.

REG combines the linear span view of secret sharing with the reconfiguration problem studied in dynamic threshold systems. Shamir and general linear secret-sharing schemes formulate reconstruction as a span condition [1, 7]; Schnorr, FROST, verifiable secret sharing, polynomial commitments, and DKG provide the fixed-state signing and public verification layer [2, 3, 4, 5, 6, 10, 9, 11]. Dynamic threshold and committee-handoff systems, including CHURP and Dynamic-FROST, study changing committees and proactive refresh [13, 15, 14, 17, 12, 21]. ADAPT contributes the GLI-based active weighted update engine [22]; weighted, silent, and succinct threshold-signature works study complementary access-policy and verification axes [25, 26, 27, 30, 31, 28, 29]. The contribution here is the reachable-state invariant needed when weighted signing material changes over a sequence of accepted updates.

1.1 Our Results

Our results are as follows.

1. **Reachable-state obstruction.** Local update validity does not compose into sequence-level active security. If accepted transitions can move a sub-threshold coalition into an authorized reachable state, active unforgeability fails by correctness of the final weighted threshold signature.

2. **Reconstruction-rank accounting.** The exposure ledger accounts for live, stale, derivative, public, and transient signing information as linear functionals of the secret-sharing state. This yields a public envelope condition that generalizes current-weight safety.

3. **REG compiler theorem.** The REG compiler exposes a single atomic reconfiguration interface.

Local operations such as weight increases, weight decreases, and threshold changes become internal subroutines. The compiler theorem shows that fixed-state weighted unforgeability, update-transcript soundness, ledger-sound refresh/handoff, and a sound rank-exposure envelope imply active unforgeability over arbitrary valid reconfiguration sequences.

4. REG-ADAPT **construction theorem.** Instantiating the compiler around ADAPT/GLI gives a guarded GLI reconfiguration construction: derivative shares become ledger rows, removed derivatives become stale rows, threshold-decrease coefficients become public rows, and old-epoch certificates bind activation.

5. **Artifact and evaluation.** The implementation on top of the public ADAPT Go artifact adds epoch states, exposure ledgers, stale-row accounting, owner-$\perp$ public rows, certifier-set-weighted digest-bound transition certificates, conservative activation guards, exact-rank audit mode over GLI derivative rows, and benchmark drivers. It reproduces the unsafe reachable-state rejection, demonstrates a conservative-versus-exact rank tightness gap, and measures metadata, local update, certificate, and audit costs. We provide the open-source REG-ADAPT artifact at `https://github.com/ainta/ADAPT_imple`.

2 From ADAPT Updates to Reachable-State Security

2.1 Weighted Threshold Signatures and Virtualization

A weighted threshold-signature state consists of a participant set $\mathcal{U}$, a weight function $w : \mathcal{U} \to \mathbb{N}$, a threshold τ, and a public key pk. A coalition $A \subseteq \mathcal{U}$ is authorized if

$$W(A) := \sum_{i \in A} w(i) \geq \tau.$$

Correctness requires every authorized coalition to be able to produce a valid signature, and fixed-state unforgeability requires coalitions below threshold to be unable to sign fresh messages except with negligible probability. Shamir secret sharing gives the standard linear-algebraic model for such threshold access structures [1]; Schnorr/FROST-style signatures give the fixed-state signing layer used by ADAPT and by our artifact [2, 10, 9].

Virtualization realizes weight w_i by giving signer i the role of w_i separate virtual signers. This is conceptually simple but expensive when weights are large: key generation, aggregation, and signing work can scale with total virtual weight rather than the number of real participants. A dynamic PoS-style system amplifies this cost because deposits, exits, and threshold changes repeatedly alter the virtualized access structure.

2.2 ADAPT and GLI Derivative Shares

ADAPT's central algebraic move is to replace virtual signer expansion with derivative rows. Instead of assigning signer i separate Shamir points for each unit of weight, ADAPT assigns one evaluation point and the derivative sequence

$$f^{(0)}(i), f^{(1)}(i), \ldots, f^{(w_i-1)}(i).$$

Generalized Lagrange interpolation reconstructs the signing secret from enough function and derivative evaluations. This is the algebraic reason ADAPT can represent large weights compactly and avoid work that scales linearly with virtualized weight.

2.3 Local Weighted Updates

ADAPT uses GLI to implement local update routines for weight increases, weight decreases, threshold increases, and threshold decreases. Weight increases add higher-order derivative shares for the target signer. Weight decreases remove derivative rows from the current policy. Threshold changes transform the sharing polynomial or expose correction information so that the new threshold is represented without running a full distributed key generation protocol. These routines are the local cryptographic engine: they update shares and commitments efficiently while preserving the stable public key used by the signing system. ADAPT's implementation results emphasize this local efficiency: dynamic updates take a small fraction of re-setup time, and highly uneven weights avoid the linear blowup of virtual signer expansion [22].

2.4 The Reachable-State Gap

Consider a coalition F with weight 99 in a state with threshold 100. The coalition is below threshold. Suppose the update interface accepts the following locally valid operations:

$$\mathsf{TIncrease} : 100 \mapsto 150, \qquad \mathsf{WIncrease} : 99 \mapsto 149, \qquad \mathsf{TDecrease} : 150 \mapsto 100.$$

The final state has $W(F) = 149 \geq 100$. The coalition can now sign by correctness. This final signature follows from the final state's authorization policy. The lesson is structural: active security must be stated over reachable states and transition invariants.

2.5 Reachability Obstruction

Definition 1 (Accepted transition relation)**.** *Let* Trans *be the transition relation induced by an update interface. We write*

$$S \xrightarrow{\mathsf{Trans}} S'$$

when the interface accepts a transcript that moves the system from S to S'. For independent local APIs, Trans *is the union of accepted operations such as* WIncrease*,* WDecrease*,* TIncrease*, and* TDecrease*.*

Definition 2 (Unsafe reachable path)**.** *The transition relation* Trans *admits an F-unsafe path if there are states*

$$S_0, S_1, \ldots, S_k$$

such that $S_j \xrightarrow{\mathsf{Trans}} S_{j+1}$ for every j, while

$$W_{S_0}(F) < \tau_{S_0} \qquad \text{and} \qquad W_{S_k}(F) \geq \tau_{S_k}.$$

Theorem 1 (Reachability obstruction)**.** *Let* Trans *be any accepted transition relation for a correct weighted threshold-signature scheme. If* Trans *admits an F-unsafe path, then no active-security definition that promises unforgeability for the initially sub-threshold coalition F over arbitrary accepted transition sequences can hold.*

Proof. Let $S_0, \ldots, S_k$ be an F-unsafe path. The adversary corrupts F in S_0. Initially $W_{S_0}(F) < \tau_{S_0}$, so F is below threshold. The adversary submits the accepted transcripts realizing the path to S_k. Since every transition is accepted, the system reaches S_k. In the final state, $W_{S_k}(F) \geq \tau_{S_k}$, so $F \in \Gamma(S_k)$. By correctness, F can run the signing protocol and produce a valid signature on any fresh message. This success follows from the final state's authorization policy. Therefore sequence-level active unforgeability is impossible over transition relations that admit such paths. □

Corollary 1 (Reachable-state invariant)**.** *Any sequence-secure active weighted reconfiguration interface must preserve a global safety invariant for every adversarial coalition covered by the security claim.*

3 Rank-Exposure Ledgers

3.1 Reconfiguration States

Definition 3 (Reconfiguration state)**.** *A reconfiguration state is a tuple*

$$S = (e, \mathcal{U}, w, \tau, \mathsf{pk}, \mathcal{L}, \mathsf{aux}),$$

where e is an epoch, $\mathcal{U}$ is the participant set, $w : \mathcal{U} \to \mathbb{N}$ is the current signing weight function, τ is the current threshold, pk is the stable verification key, $\mathcal{L}$ is the public exposure ledger, and aux contains public commitments, policy metadata, and transcript digests.

Definition 4 (Authorized sets)**.** *For $A \subseteq \mathcal{U}$, define*

$$W_S(A) := \sum_{i \in A} w_S(i).$$

The authorized sets in state S are

$$\Gamma(S) := \{A \subseteq \mathcal{U}_S : W_S(A) \geq \tau_S\}.$$

3.2 Linear Exposure Model

The ledger is a public syntactic object. Its security meaning is supplied by a row interpretation map that interprets live entries in the linear space protecting the current signing secret.

Definition 5 (State row model)**.** *Each active state S has a finite-dimensional vector space $\mathcal{V}_S$ over $\mathbb{F}$ and a target row $\sigma_S \in \mathcal{V}_S^*$. If $x_S \in \mathcal{V}_S$ is the current secret-sharing coordinate vector, then $\sigma_S(x_S)$ is the signing secret represented by the public key. This is the usual linear-secret-sharing view in which reconstruction is a span condition on a target vector [7]. For a Shamir polynomial $f(x) = \sum_{\ell=0}^{\tau-1} a_\ell x^\ell$, one may take $\mathcal{V}_S = \mathbb{F}^\tau$, $x_S = (a_0, \ldots, a_{\tau-1})$, and $\sigma_S(x_S) = a_0 = f(0)$.*

Definition 6 (Ledger entry and row interpretation)**.** *A ledger entry is a tuple*

$$(i, \eta, \mathrm{src}, \mathrm{scope}, \mathrm{exp}),$$

where $i \in \mathcal{U} \cup \{\perp\}$ is the owner, η is a syntactic row descriptor, src *records its source,* scope $\in \{\mathrm{private}, \mathrm{public}, \mathrm{transient}\}$*, and* exp *is an expiry condition. The owner $\perp$ denotes public scalar information. The state row provider maps a live descriptor to a linear functional*

$$\eta_S \in \mathcal{V}_S^*.$$

Commitments, verification equations, and proofs are public verification data used to establish transcript soundness; they become ledger rows only when the underlying scalar is opened or otherwise learned as a linear functional.

Definition 7 (Interpreted ledger rows)**.** *For a coalition $F \subseteq \mathcal{U}_S$, define*

$$\mathsf{Rows}_S(F) := \{\eta_S : (i, \eta, \mathrm{src}, \mathrm{scope}, \mathrm{exp}) \in \mathcal{L}_S,\ i \in F \cup \{\perp\},\ \mathrm{exp} \text{ has not expired in } S\}.$$

Definition 8 (Reconstruction and safety)**.** *The coalition F reconstructs the signing secret in state S if*

$$\mathsf{Rec}_S(F) = 1 \quad \Longleftrightarrow \quad \sigma_S \in \mathrm{span}(\mathsf{Rows}_S(F)).$$

The coalition is reconstruction-safe in S, written $\mathsf{Safe}_S(F)$, if $\mathsf{Rec}_S(F) = 0$.

Definition 9 (Exposure rank)**.** *The exposure rank of coalition F in state S is*

$$\rho_S(F) := \mathrm{rank}\, \mathsf{Rows}_S(F).$$

Rank is a deployment-friendly sufficient measure for reconstruction safety. It is not the definition of secrecy: the definition of secrecy is whether the target row σ_S lies in the adversary span.

Lemma 1 (Rank safety for Shamir/GLI row systems)**.** *Consider a Shamir/GLI state whose interpreted row system has reconstruction distance τ_S: every set $E \subseteq \mathcal{V}_S^*$ satisfying $\sigma_S \in \mathrm{span}(E)$ has rank at least τ_S. Then $\rho_S(F) < \tau_S$ implies $\mathsf{Safe}_S(F)$.*

Proof. For Shamir rows and GLI derivative rows at distinct participant points with multiplicities, the interpreted row matrix is confluent Vandermonde. Any row set of rank below the reconstruction distance leaves a nonzero degree-$(\tau_S - 1)$ perturbation that preserves all exposed rows while changing the target row value, so it cannot span σ_S. Suppose $\mathsf{Safe}_S(F)$ is false. Then $\sigma_S \in \mathrm{span}(\mathsf{Rows}_S(F))$. By reconstruction distance, the interpreted row set $\mathsf{Rows}_S(F)$ has rank at least τ_S. This contradicts $\rho_S(F) < \tau_S$. □

3.3 Exposure Envelopes

Definition 10 (Environmental corruption envelope)**.** *An environment specifies a family $\mathcal{C}_\beta(S)$ of coalitions covered by the security claim. For PoS, a typical example is*

$$\mathcal{C}_\beta(S) := \{F \subseteq \mathcal{U}_S : \mathrm{stake}_S(F) \le \beta \cdot \mathrm{stake}_S(\mathcal{U}_S)\}.$$

The exact rank envelope is

$$B^\star_\beta(S) := \max_{F \in \mathcal{C}_\beta(S)} \rho_S(F).$$

The protocol may use any efficiently computable public upper bound

$$\overline{B}_\beta(S) \ge B^\star_\beta(S).$$

Definition 11 (Envelope safety)**.** *A state S is envelope-safe if every covered coalition is reconstruction-safe:*

$$\forall F \in \mathcal{C}_\beta(S) : \quad \mathsf{Safe}_S(F).$$

For Shamir/GLI states, the public rank condition

$$\overline{B}_\beta(S) < \tau_S$$

implies envelope safety by Lemma 1.

3.4 Transient Exposure Model

Reconfiguration protocols often create intermediate information before the next state is active. Old rows, new rows, and update transcript rows may live in different coordinate spaces. A transition therefore has its own row interpretation space.

Definition 12 (Transient row interpretation)**.** *For a transition* $(S_e, \mathsf{tr}, S_{e+1})$*, let*

$$\mathcal{V}_{e,\mathsf{tr},e+1}$$

be the common ambient space used to interpret old-state rows, new-state rows, and scalar rows opened by the update transcript. Let

$$\Sigma_{e,\mathsf{tr},e+1} \subseteq \mathcal{V}^*_{e,\mathsf{tr},e+1}$$

be the protected transition rows. These include the old signing-secret row, the proposed new signing-secret row, and any temporary handoff row whose reconstruction would give signing capability during the transition. The transient row provider maps the joined ledger

$$\mathcal{L}_e \parallel \mathsf{tr} \parallel \mathcal{L}_{e+1}$$

to interpreted rows

$$\mathsf{Rows}_{e,\mathsf{tr},e+1}(F) \subseteq \mathcal{V}^*_{e,\mathsf{tr},e+1}.$$

Definition 13 (Transient safety)**.** *A transition transcript* tr *from* S_e *to* S_{e+1} *is transient-safe if every covered coalition is safe for every protected transition row:*

$$\forall F \in \mathcal{C}_\beta(S_e),\ \forall \sigma \in \Sigma_{e,\mathsf{tr},e+1} : \quad \sigma \notin \operatorname{span}(\mathsf{Rows}_{e,\mathsf{tr},e+1}(F))\,.$$

In the rank-envelope fast path, it suffices that

$$\overline{B}_\beta(S_e \parallel \mathsf{tr} \parallel S_{e+1}) < \tau^{\mathrm{tr}}_{e\to e+1},$$

where $\tau^{\mathrm{tr}}_{e\to e+1}$ *is the transient reconstruction threshold supplied by the update engine.*

3.5 Sequence-Level Active Security

Definition 14 (ARWTS experiment)**.** *The experiment* $\mathsf{Exp}^{\mathrm{arwts}}_{\mathcal{A}}$ *maintains an active state* S_e*, a corrupted set* F_e*, and a table* $\mathcal{Q}$ *of honest signing help. The adversary has access to:*

- $\mathsf{Corrupt}(i)$*, which adds* i *to* F_e *and reveals all unexpired private ledger entries owned by* i*;*
- $\mathsf{SignOracle}(A, m)$*, which returns honest signing shares from* A *in the current state and records the queried honest weight for* $(e, \mathsf{Digest}(S_e), m)$*;*
- $\mathsf{Reconfig}(S', \mathsf{tr}, \mathsf{Cert})$*, which runs the compiled activation predicate and, if it accepts, makes* S' *the active state.*

If after any adversarial action the corrupted set leaves $\mathcal{C}_\beta(S_e)$*, the experiment raises the environment-failure event* $\mathsf{Bad}_{\mathrm{env}}$*. The adversary wins if it outputs* (S_j, m, σ) *such that* S_j *is a reachable active state,* $\mathsf{Vrf}_{S_j}(\mathsf{pk}, m, \sigma) = 1$*, and the output would be fresh in the fixed-state unforgeability game for* S_j *under the same corruption and signing-query transcript. Thus the compiler treats freshness as part of the fixed-state signature interface it lifts.*

4 The Rank-Exposure Guard Compiler

4.1 Compiler Inputs

The compiler REG takes four inputs.

1. A fixed-state weighted threshold signature scheme Π_{sig}.
2. A one-step update engine Upd, which transforms signing material from S_e to S_{e+1} and outputs a public transcript tr.
3. A ledger-update algorithm that maps $(S_e, S_{e+1}, \mathsf{tr})$ to final and transient ledgers.
4. A public envelope algorithm computing $\overline{B}_\beta$.

4.2 Compiler Algorithms

$\mathsf{Propose}(S_e, \Delta)$
Given a policy update Δ, compute a canonical candidate state

$$S_{e+1} = (e+1, \mathcal{U}_{e+1}, w_{e+1}, \tau_{e+1}, \mathsf{pk}, \mathcal{L}_{e+1}, \mathsf{aux}_{e+1}).$$

The proposal advances to preparation exactly when the public final rank check $\overline{B}_\beta(S_{e+1}) < \tau_{e+1}$ holds.

$\mathsf{Prepare}(S_e, S_{e+1})$
An authorized old-epoch set runs Upd to create new shares and public verification material. The output is an update transcript tr. Local operations such as weight increases and threshold changes occur internally; the atomic activation boundary is the reconfiguration state.

$\mathsf{Validate}(S_e, S_{e+1}, \mathsf{tr})$
The public predicate checks:

1. syntax, epoch monotonicity, domain separation, and preservation of the intended public key;
2. transcript soundness of commitments, share-verification equations, and complaint/abort rules;
3. final safety, $\overline{B}_\beta(S_{e+1}) < \tau_{e+1}$;
4. transient safety,
$$\overline{B}_\beta(S_e \parallel \mathsf{tr} \parallel S_{e+1}) < \tau^{\mathrm{tr}}_{e \to e+1}.$$

$\mathsf{Certify}(S_e, S_{e+1}, \mathsf{tr})$
An honest old-epoch signer first runs $\mathsf{Validate}(S_e, S_{e+1}, \mathsf{tr})$. If validation fails, it returns $\perp$. If validation succeeds, it produces a reconfiguration signature share on the transition digest. Corrupted signers may sign arbitrary digests. The certificate is a threshold signature under S_e:

$$\mathsf{Cert}_e := \mathsf{Sign}_{S_e}(\mathsf{reconfigure}, \mathsf{Digest}(S_e, S_{e+1}, \mathsf{tr})).$$

$\mathsf{Activate}(S_e, S_{e+1}, \mathsf{tr}, \mathsf{Cert}_e)$
The next state activates when $\mathsf{Validate} = 1$ and Cert_e verifies under the old public key. All application signatures are domain-separated by epoch and state digest.

$\mathsf{GarbageCollect}(S_e)$
Ledger entries are removed when their expiry condition is publicly satisfied. In a PoS system, expiry can depend on unbonding, slashing, and finality windows. In a custody system, expiry can depend on policy-delay or key-rotation windows. The default rule keeps an entry live until the surrounding protocol proves it has zero future reconstruction rank.

4.3 Component Games

Definition 15 (Transcript-sound update)**.** *An update engine is transcript-sound if no efficient adversary can produce an accepted transcript* $(S_e, S_{e+1}, \mathsf{tr})$ *whose public verification equations hold but whose scalar deliveries, public openings, or state digest do not implement the claimed transition relation. Let* $\mathsf{Adv}_e^{\mathrm{upd}}$ *denote the maximum probability of this event at epoch* e*.*

Definition 16 (Ledger-sound update)**.** *An accepted update transcript is ledger-sound if every scalar exposure row learned by the adversary during the update is contained in*

$$\mathrm{span}(\mathsf{Rows}_{e,\mathsf{tr},e+1}(F))\,.$$

Scalar exposure rows include shares, derivative values, opened coefficients, and public correction terms. Here "learned" means explicitly delivered or opened to a corrupted party, broadcast publicly, or reconstructible from accepted transcript messages together with corrupted local state. Commitments, verification equations, and proofs are public verification data for transcript soundness; they are not scalar exposure rows unless their underlying scalar is opened. Let $\mathsf{Adv}_e^{\mathrm{ledger}}$ *denote the maximum probability that an accepted update leaks a scalar exposure row outside the recorded transient span.*

Definition 17 (Fixed-state signature game)**.** *For an active state* S_e*,* $\mathsf{Adv}_e^{\mathrm{sig}}$ *is the maximum probability that an efficient adversary whose view is contained in* $\mathrm{span}(\mathsf{Rows}_{S_e}(F))$*, with* $\mathsf{Safe}_{S_e}(F)$*, outputs a fresh valid application signature in state* S_e*. This game is the abstraction of the fixed-state signing layer: the row span captures secret-sharing exposure, while nonce handling, commitments, abort behavior, and ordinary signing transcripts are covered by the fixed-state unforgeability experiment.*

Definition 18 (Transition-certificate game)**.** *For an active state* S_e*,* $\mathsf{Adv}_e^{\mathrm{cert}}$ *is the maximum probability that an efficient adversary whose corrupted old-epoch view is reconstruction-safe in* S_e *produces a valid old-epoch certificate on a transition digest that honest signers would reject under* $\mathsf{Validate}$*.*

Definition 19 (Envelope failure)**.** $\mathsf{Adv}^{\mathrm{env}}$ *is the probability that during the* ARWTS *experiment one of the following events occurs:*

$$\mathsf{Bad}_{\mathrm{corr}} \vee \mathsf{Bad}_{\mathrm{bound}} \vee \mathsf{Bad}_{\mathrm{rank}}.$$

Here $\mathsf{Bad}_{\mathrm{corr}}$ *is the event that the corrupted set leaves* $\mathcal{C}_\beta(S)$*;* $\mathsf{Bad}_{\mathrm{bound}}$ *is the event that the public bound* $\overline{B}_\beta$ *fails to upper-bound the exposure rank of a covered coalition; and* $\mathsf{Bad}_{\mathrm{rank}}$ *is the event that a state satisfying the public rank check* $\overline{B}_\beta(S) < \tau_S$ *fails reconstruction safety because the deployed row system lacks the claimed reconstruction distance.*

4.4 Lifting Theorem

Theorem 2 (Rank-exposure guard lifting theorem)**.** *Let* $\mathsf{REG}[\Pi_{\mathrm{sig}}, \mathsf{Upd}]$ *be the compiled reconfiguration scheme. Suppose:*

1. *the initial state is envelope-safe;*
2. Π_{sig} *has fixed-state advantage* $\mathsf{Adv}_e^{\mathrm{sig}}$ *in every reachable state;*
3. Upd *has transcript-soundness advantage* $\mathsf{Adv}_e^{\mathrm{upd}}$*;*
4. Upd *has ledger-soundness advantage* $\mathsf{Adv}_e^{\mathrm{ledger}}$*;*

5. old-epoch transition certificates have advantage $\mathsf{Adv}_e^{\mathrm{cert}}$*; and*

6. the envelope failure probability is $\mathsf{Adv}^{\mathrm{env}}$*.*

Then $\mathsf{REG}[\Pi_{\mathrm{sig}}, \mathsf{Upd}]$ *satisfies sequence-level active unforgeability in the* ARWTS *experiment. More precisely,*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{arwts}} \le \sum_e \left(\mathsf{Adv}_e^{\mathrm{sig}} + \mathsf{Adv}_e^{\mathrm{cert}} + \mathsf{Adv}_e^{\mathrm{upd}} + \mathsf{Adv}_e^{\mathrm{ledger}} \right) + \mathsf{Adv}^{\mathrm{env}},$$

where the terms correspond to fixed-state signature forgery, invalid-transition certification, update-transcript soundness, ledger under-accounting, and environmental-envelope failure.

Proof. Let the bad events at epoch e be:

$$\mathsf{Bad}_e^{\mathrm{cert}}, \quad \mathsf{Bad}_e^{\mathrm{upd}}, \quad \mathsf{Bad}_e^{\mathrm{ledger}}, \quad \mathsf{Bad}_e^{\mathrm{sig}},$$

corresponding to the component games above, and let $\mathsf{Bad}^{\mathrm{env}}$ be the envelope failure event. We show that if none of these events occurs, the adversary cannot win the ARWTS experiment.

First, every activated transition is valid. Activation requires $\mathsf{Validate}(S_e, S_{e+1}, \mathsf{tr}) = 1$ and a digest-bound old-epoch certificate. If the transcript is accepted but does not implement the claimed state and ledger transition, then $\mathsf{Bad}_e^{\mathrm{upd}}$ occurs. If the transition digest is invalid for honest signers and still has a valid old-epoch certificate, then $\mathsf{Bad}_e^{\mathrm{cert}}$ occurs because honest signers return $\perp$ on invalid digests and the corrupted old-epoch view is reconstruction-safe. Thus, outside these events, activation applies exactly the transition validated by the compiler.

Second, activated states preserve reconstruction safety for all covered coalitions. The base state is envelope-safe by assumption. For the induction step, validation checks the public final rank bound $\overline{B}_\beta(S_{e+1}) < \tau_{e+1}$ and the public transient rank bound. Outside $\mathsf{Bad}_{\mathrm{corr}}$, the actual corrupted set is in $\mathcal{C}_\beta$. Outside $\mathsf{Bad}_{\mathrm{bound}}$, the public bounds upper-bound the exposure ranks of all covered coalitions. Outside $\mathsf{Bad}_{\mathrm{rank}}$, the Shamir/GLI rank-safety condition converts the strict rank inequalities into final and transient reconstruction safety. Outside $\mathsf{Bad}_e^{\mathrm{ledger}}$, the scalar rows learned by the adversary during the update are in the recorded transient span, so the row interpretation continues to dominate the adversary's actual linear view. Hence every activated successor remains reconstruction-safe for every coalition in $\mathcal{C}_\beta$, and no unrecorded transient exposure is carried into the next epoch.

Finally, suppose the adversary wins the ARWTS experiment without any bad event. The winning signature is in a reachable active state S_j, and the message is fresh relative to the signing oracle. By the previous paragraph, the corrupted coalition is reconstruction-safe in S_j, and the adversary's view is contained in the span of the interpreted ledger rows for that coalition. Therefore the winning signature is exactly a fixed-state application-signature forgery in state S_j, giving $\mathsf{Bad}_j^{\mathrm{sig}}$, a contradiction. Taking a union bound over epochs gives the stated advantage. □

5 REG-ADAPT: Guarded GLI Reconfiguration

REG-ADAPT uses ADAPT/GLI as the one-step weighted update engine and uses the guard compiler as the external reconfiguration interface. The public state contains the ADAPT verification state, the current weight vector and threshold, the exposure ledger, the stale-row expiry policy, and the current epoch digest. A transition activates only after the ADAPT update transcript verifies, the final and transient envelopes are safe, and the old epoch certifies the transition digest

$$d_e = H(S_e, S_{e+1}, \mathsf{tr}).$$

The certificate binds the old state digest, the proposed next state digest, the update transcript, the operation descriptor, and the chosen certifier set. The certifier set is part of the activation object: its old-epoch weight must meet the old threshold, and its members produce the aggregate transition certificate under the old ADAPT/FROST signing key.

5.1 GLI Shares as Linear Functionals

Let $f(x) = \sum_{\ell=0}^{d} a_\ell x^\ell$ be the current secret-sharing polynomial. The derivative share

$$f^{(k)}(i) = \sum_{\ell=k}^{d} \ell(\ell-1)\cdots(\ell-k+1)\, a_\ell\, i^{\ell-k}$$

is a linear functional of the coefficient vector $(a_0, \ldots, a_d)$. Thus every GLI share has a canonical ledger row. The rank of a coalition's ledger slice is exactly the rank of the confluent Vandermonde-style matrix induced by its unexpired derivative rows, plus any public rows from update transcripts.

5.2 Ledger Rules for ADAPT Updates

Operation	ADAPT scalar or public data	REG ledger treatment
WIncrease	New derivative $f^{(w_j+k)}(j) = \sum_{i\in P} \mu_{i,k}$.	Private row for owner j; visible to the adversary when j is corrupted.
WDecrease	Removed derivative $f^{(k)}(j)$.	Stale private row for owner j until the expiry rule proves it irrelevant.
WDecrease	Correction $h_i^{(k)}(\ell)$ delivered to recipient ℓ.	Private or transient row for ℓ; $h_i(0) = 0$ preserves the signing secret row.
TIncrease	Transformed value $\mu_{i,\ell} = (\delta_i F_i)(\ell) h_i(\ell) + d_i + e_i \rho_i$.	Private or transient row for ℓ; $\sum_i \mu_{i,\ell}$ contributes to ℓ's new share.
TDecrease	Broadcast coefficient f_{t-k}.	Owner-$\perp$ public row, counted for every coalition.
All updates	Commitments, equations, hashes, complaints.	Public verification data; not scalar exposure unless the underlying scalar is opened.

Table 1: ADAPT-to-REG ledger interpretation. Opened scalar values become ledger rows. Public verification data establishes transcript soundness but is not counted as linear exposure unless it reveals the scalar itself.

Weight increase
If signer j's weight increases from w_j to $w_j + \alpha$, the update adds rows

$$f^{(w_j)}(j), \ldots, f^{(w_j+\alpha-1)}(j)$$

to j's live ledger. In ADAPT notation, each authorized participant $i \in P$ computes

$$\mu_{i,k} = (\delta_i F_i)^{(w_j+k)}(j), \qquad 0 \le k < \alpha,$$

and the target updates

$$sk_j^{(w_j+k)} = \sum_{i\in P} \mu_{i,k}.$$

Ledger soundness requires the delivery protocol for these derivatives to reveal no independent scalar rows beyond the rows assigned to j. Public commitments, $\hat{s}_i$-style proofs, and verification equations are checked by transcript soundness and become ledger rows only if the corresponding scalar is opened.

Weight decrease
If signer j's current signing weight decreases, the removed derivative rows become stale entries with expiry tied to the corruption model, refresh window, or unbonding policy. This is the key difference between current weight and rank exposure: a row removed from the current signing policy can still contribute to reconstruction until its expiry condition holds. ADAPT's weight-decrease masking polynomial $h_i(x)$ is treated through its scalar outputs. For each remaining recipient ℓ and derivative order k, the delivered correction $h_i^{(k)}(\ell)$ contributes to

$$sk_\ell^{(k)} \leftarrow sk_\ell^{(k)} + \sum_{i \in P} h_i^{(k)}(\ell).$$

If ℓ is corrupted these correction values are private or transient rows for ℓ; if ℓ is honest they are hidden by the delivery channel. The zero-secret-term condition $h_i(0) = 0$ prevents the mask itself from changing the protected signing secret.

Threshold increase
If the update engine transforms the sharing polynomial or raises the reconstruction degree, the ledger records the induced linear map on existing rows and all new rows created by the update transcript. In ADAPT notation, participants compute transformed values

$$\mu_{i,\ell} = (\delta_i F_i)(\ell) \cdot h_i(\ell) + d_i + e_i \rho_i$$

and distribute them to users ℓ. These $\mu_{i,\ell}$ values are recorded as private or transient rows for the recipients that learn them. The aggregate $\sum_i \mu_{i,\ell}$ is the scalar contribution to recipient ℓ's new share component, while higher-order components are obtained from the corresponding product-rule derivative expressions. Masking values whose constant term is fixed to zero, or whose randomness is independent of the protected secret, do not add reconstruction-relevant rows for nonrecipients. The transition is valid exactly when the transient ledger is safe during the handoff.

Threshold decrease
Threshold decrease can expose high-degree coefficients or other public correction terms. Such public information is recorded with owner $\perp$. In ADAPT's threshold-decrease routine, the broadcast coefficients

$$f_{t-k} = \sum_{i \in P} \frac{\mu_{i,k}}{(t-k)!}$$

computed from high-order derivative/correction values $\mu_{i,k}$ are exactly such public scalar rows. They contribute to every coalition's exposure rank until garbage collection proves zero future reconstruction rank. The owner-$\perp$ rule makes public update information visible to every coalition in the envelope computation.

Activation
The public API activates the final atomic

$$\mathsf{Reconfig}(S_e, S_{e+1}, \mathsf{tr})$$

state with an old-epoch certificate.

Transcript soundness for REG-ADAPT
For REG-ADAPT, transcript soundness instantiates the generic update-soundness game operation by operation. For $WIncrease$, the delivered derivatives must match the committed GLI relation for

the target point and derivative orders, and the proof values $\hat{s}_i, \hat{c}_i$ and commitments must bind the sender to those delivered $\mu_{i,k}$ values. For $WDecrease$, the updated shares must match the derivative corrections $h_i^{(k)}(\ell)$, the constraint $h_i(0) = 0$, and the removed derivative rows recorded as stale rows. For $TIncrease$, the new shares must match the transformed polynomial relation, including that each recipient ℓ combines the delivered $\mu_{i,\ell}$ values into its new share component while the $d_i + e_i\rho_i$ terms serve only as recipient-owned masks or transcript authentication randomness. For $TDecrease$, the broadcast coefficients f_{t-k} must match the aggregated high-order terms $\sum_{i\in P} \mu_{i,k}/(t-k)!$. In every case, the state digest must bind the claimed next weight vector, threshold, stale rows, owner-$\perp$ public rows, transient ledger entries, and activation certificate metadata.

5.3 Computing the Envelope

The exact envelope $B^\star_\beta(S)$ is a maximum-rank problem over a coalition family. A deployable system uses conservative upper bounds. For example:

$$\overline{B}_\beta(S) = r_{\text{pub}}(S) + \max_{F\in\mathcal{C}_\beta(S)} \sum_{i\in F} r_i(S),$$

where $r_{\text{pub}}(S)$ is the rank of public ledger rows and $r_i(S)$ upper-bounds the number of independent unexpired private rows owned by i. For PoS, the maximization can be upper-bounded by exposed stake:

$$\overline{B}_\beta(S) \le r_{\text{pub}}(S) + \beta \cdot W_{\text{exposed}}(S),$$

where W_{exposed} includes active, pending, unbonding, and stale-share weights. Loose bounds reduce liveness while preserving the security invariant.

5.4 Operation-Level Ledger Soundness

The REG-ADAPT security proof uses operation-level ledger-soundness lemmas for the ADAPT/GLI update routines. The lemmas enumerate the scalar families that can enter the adversary's linear view. Values delivered or opened as field elements are ledger rows; commitments, hashes, Schnorr responses, verification equations, and complaint records are public verification data for transcript soundness and become exposure rows only when the underlying scalar is opened.

Lemma 2 (Weight-increase ledger soundness)**.** *For an accepted, transcript-sound GLI weight-increase transcript, every reconstruction-relevant row learned by the adversary is in the span of the old rows available under the corruption and erasure model, the newly assigned derivative rows, and the public scalar openings recorded in the ledger.*

Proof. A GLI weight increase assigns additional derivative evaluations $f^{(w_j)}(j), \ldots, f^{(w_j+\alpha-1)}(j)$ to the target owner and publishes only the verification material needed to check delivery. Equivalently, the only new scalar GLI rows for owner j are the row descriptors for

$$\sum_{i\in P} \mu_{i,k} = \sum_{i\in P} (\delta_i F_i)^{(w_j+k)}(j) = f^{(w_j+k)}(j), \qquad 0 \le k < \alpha.$$

Let F be the corrupted set. The adversary's old secret-dependent information is, by the induction invariant, contained in $\text{span}(\mathsf{Rows}_{S_e}(F))$. During the update, honest messages are either deliveries of the $\mu_{i,k}$ values contributing to the new derivative rows, commitments or verification equations for those rows, or randomness independent of the sharing polynomial. Commitments, $\hat{s}_i$-style proofs, and verification equations are public verification data for transcript soundness and do not expose

scalar rows unless the scalar is opened. Deliveries to honest owners are hidden from the adversary by the delivery channel. Deliveries to corrupted owners are recorded at the granularity exposed by the delivery protocol: if the recipient learns only the aggregate derivative, the aggregate row $f^{(w_j+k)}(j)$ is recorded; if the recipient learns individual sender contributions $\mu_{i,k}$, those contribution rows are recorded as recipient-owned transient rows and their aggregate is the assigned derivative row. Hence every secret-dependent scalar row extractable from the adversary's post-update view lies in the span of the available old rows, the newly assigned corrupted derivative rows, and the recorded delivery rows. □

Lemma 3 (Weight-decrease stale-row soundness)**.** *For an accepted, transcript-sound GLI weight-decrease transcript, every reconstruction-relevant row learned by the adversary is in the span of the old rows available under the corruption and erasure model, the removed derivative rows recorded as stale rows, the masking-polynomial corrections delivered to corrupted recipients, and the public scalar openings recorded with owner $\perp$.*

Proof. The current authorization policy may stop counting a derivative row immediately, but a previously corrupted owner may still remember the row, and an adaptively corrupted owner may reveal old local state unless the deployed erasure rule has certified deletion. The ledger therefore moves removed rows that remain available under the corruption/erasure model from live status to stale status instead of deleting them. ADAPT's weight-decrease routine also uses masking-polynomial material to update the remaining sharing state. In the ADAPT notation, each participant i samples a mask $h_i(x)$ with $h_i(0) = 0$, publishes commitments to the coefficients of h_i, and privately delivers derivative evaluations

$$h_i^{(k)}(\ell)$$

to the recipients whose shares must be corrected. For each recipient ℓ, the accepted transcript updates the remaining derivative rows by

$$sk_\ell^{(k)} \leftarrow sk_\ell^{(k)} + \sum_{i \in P} h_i^{(k)}(\ell)$$

for the operation-specific range of k. Transcript soundness enforces that the delivered field elements match the committed mask polynomial and that the committed mask has zero constant term. The public commitments, Schnorr-style delivery proofs, complaint records, and verification equations bind the delivery transcript, but they reveal no scalar coefficient of h_i unless an opening is made public. Hence the scalar families available to the adversary after a weight decrease are exactly:

$$\{f^{(k)}(j) : \text{removed row still available for owner } j\}, \qquad \{h_i^{(k)}(\ell) : \ell \in F \text{ and the value is delivered to } \ell\},$$

together with scalar openings visible to every party. The first family is recorded as stale private rows, the second as recipient-owned private or transient rows, and the third as owner $\perp$ rows. Since $h_i(0) = 0$, adding the correction polynomial preserves the signing secret row; any reconstruction-relevant information about the corrected sharing state is therefore a linear combination of the pre-update rows and the recorded correction rows. Which removed rows remain stale is determined by the deployed erasure model: without certified erasure, a removed row owned by a party that can later be corrupted remains in the ledger until expiry. Thus the exposure rank used by the guard dominates the adversary's post-update reconstruction rank. □

Lemma 4 (Threshold-decrease public-row soundness)**.** *For an accepted, transcript-sound threshold-decrease transcript, public correction terms released during threshold decrease are owner-$\perp$ rows and count for every coalition until they have zero future reconstruction rank.*

Proof. In ADAPT threshold decrease, parties compute high-order derivative/correction values

$$\mu_{i,k} = (\delta_i F_i)^{(t-k)}(0), \qquad f_{t-k} = \sum_{i \in P} \frac{\mu_{i,k}}{(t-k)!}.$$

The aggregation step broadcasts the coefficient terms f_{t-k}. These broadcast coefficients are scalar linear functionals of the sharing state and are observed by every coalition. They therefore cannot be charged to a particular private owner. In the coefficient-coordinate row model, f_{t-k} is the row extracting the a_{t-k} coefficient of the sharing polynomial. Recording each such row with owner $\perp$ makes it part of every interpreted ledger slice $\mathsf{Rows}_S(F)$. The public-row rank is added to every envelope computation, so no coalition's exposure is understated by treating public threshold-decrease information as private or local update state. □

Lemma 5 (Threshold-increase transient soundness)**.** *For an accepted, transcript-sound threshold-increase transcript, every reconstruction-relevant row revealed by the handoff transcript is recorded in the transient ledger or is linearly dependent on rows already recorded there.*

Proof. A threshold increase may temporarily relate old and new sharing spaces through a handoff transcript. The transient ledger joins the old rows, proposed new rows, and transcript rows before activation. ADAPT-style threshold increase sends recipient-owned transformed scalars

$$\mu_{i,\ell} = (\delta_i F_i)(\ell) \cdot h_i(\ell) + d_i + e_i \rho_i$$

from participants i to recipients ℓ. The public ADAPT artifact instantiates this transformation with the $xf(x)$-style map, so the delivered scalar has the form

$$\mu_{i,\ell} = (\lambda_i F_i)(\ell) \cdot \ell + d_i + e_i \rho_i.$$

These $\mu_{i,\ell}$ values are field elements learned by their recipients, so the row provider records them as recipient-owned transient rows whenever the recipient is corrupted or the value is otherwise opened. For recipient ℓ, the aggregate

$$\sum_{i \in P} \mu_{i,\ell}$$

is the new base-share component in the higher-threshold sharing space. Higher derivative rows are deterministic linear functions of old derivative rows and delivered transformed rows. In the artifact's $xf(x)$ instance, the product rule gives

$$(xf(x))^{(k)}(\ell) = kf^{(k-1)}(\ell) + \ell f^{(k)}(\ell),$$

and the general ADAPT-style transformation uses the corresponding Leibniz-rule expressions with the derivatives of h_i. Every derivative $h_i^{(r)}(\ell)$ used in this expansion is handled by the same rule: if it is delivered or opened as a field element, it is a recipient-owned or owner-$\perp$ ledger row; if it remains hidden, transcript soundness only certifies the delivery relation and contributes no row. These derived rows therefore add no independent scalar information outside the span of the old rows and the recorded delivered rows.

The mask term $d_i + e_i \rho_i$ is accounted for according to what is exposed. If it appears only inside the delivered transformed scalar, then $\mu_{i,\ell}$ is the recorded recipient-owned row. If the mask term is separately opened, the opening is public verification data or an owner-$\perp$ scalar as appropriate; because the mask is independent of the protected signing secret, it does not increase reconstruction rank except through the secret-dependent transformed row it authenticates. Public

commitments, transcript hashes, Schnorr responses, and verification equations authenticate the transformed-share relation but do not expose scalar rows unless the scalar itself is opened. Therefore every reconstruction-relevant scalar exposed while old and new material coexist is recorded in the transient ledger or is linearly dependent on rows recorded there. Transient safety checks this joined span before the new state can activate. □

The lemmas above prove the scalar-exposure accounting needed by the compiler; transcript soundness is the separate condition that public verification data really enforces the claimed update relations. Expiry rules, identifiable aborts, transcript availability, adaptive erasures, and the commit boundary are protocol parameters: they determine when rows leave $\mathsf{Rows}_S(F)$, how disputed openings become owner-$\perp$ rows, and when the ADAPT key update and REG metadata update are activated atomically.

Theorem 3 (REG-ADAPTconstruction theorem)**.** *Assume:*

1. *the ADAPT/GLI signing layer satisfies fixed-state unforgeability for reconstruction-safe coalitions;*
2. *the ADAPT/GLI one-step update routines satisfy operation-level transcript soundness for the public verification data in Table 1;*
3. *the deployed envelope $\overline{B}_\beta$ soundly upper-bounds live, stale, public, and transient exposure and satisfies the Shamir/GLI rank-safety condition; and*
4. *old-epoch transition certificates are unforgeable and digest-bound.*

Then REG-ADAPT *satisfies sequence-level active weighted threshold-signature security. More precisely, for every efficient adversary $\mathcal{A}$,*

$$\mathsf{Adv}^{\text{REG-ADAPT}}_{\mathcal{A}} \le \sum_e \left(\mathsf{Adv}^{\text{ADAPT-sig}}_e + \mathsf{Adv}^{\text{ADAPT-upd}}_e + \mathsf{Adv}^{\text{ADAPT-ledger}}_e + \mathsf{Adv}^{\text{cert}}_e\right) + \mathsf{Adv}^{\text{env}}.$$

Here $\mathsf{Adv}^{\text{ADAPT-sig}}_e$ is the fixed-state ADAPT/GLI signing advantage, $\mathsf{Adv}^{\text{ADAPT-upd}}_e$ is the ADAPT transcript-soundness advantage, $\mathsf{Adv}^{\text{ADAPT-ledger}}_e$ is the maximum of the operation-specific ledger under-accounting events ruled out by Lemmas 2–5, $\mathsf{Adv}^{\text{cert}}_e$ is the old-epoch certificate-forgery advantage, and $\mathsf{Adv}^{\text{env}}$ is the envelope-failure probability.

Proof. Instantiate Theorem 2 with ADAPT/GLI as the fixed-state weighted threshold signature scheme and with the four ADAPT local update routines as the one-step update engine. Transcript soundness for these routines gives the $\mathsf{Adv}^{\text{ADAPT-upd}}_e$ term. Lemmas 2–5 show that, outside the corresponding operation-specific ledger events, every reconstruction-relevant scalar exposed by an accepted ADAPT update is recorded in the final or transient ledger, giving $\mathsf{Adv}^{\text{ADAPT-ledger}}_e$. The deployed public envelope and the Shamir/GLI rank-safety condition instantiate $\mathsf{Adv}^{\text{env}}$, and the digest-bound old-epoch transition certificate instantiates $\mathsf{Adv}^{\text{cert}}_e$. Substituting these terms into the lifting theorem gives the stated bound. □

5.5 Artifact and Evaluation

The compiler adds one old-epoch threshold signature per reconfiguration, transcript validation, and rank-envelope accounting. For systems that already reconfigure by epochs, the certificate is a single extra threshold signature over a state digest. Ledger accounting can be implemented with coarse

conservative weights for the fast path and exact linear-rank checks for audit or dispute paths. The local cryptographic work of ADAPT/GLI is unchanged.

We implemented the REG-ADAPT wrapper directly on top of the public Go ADAPT artifact [23]. The wrapper adds an explicit epoch state, weight vector, stale-row ledger, owner-$\perp$ public rows, final exposure checks, transient exposure checks, exact-rank audit mode, and atomic activation around the existing ADAPT local update routines. The transition certificate is digest-bound: the message includes the pre-state digest, post-state digest, operation descriptor, target, update magnitude, certifier-set digest, and certifier weight.

Code availability. The open-source implementation and benchmark results are available at `https://github.com/ainta/ADAPT_imple` [24]. The repository is a fork of the public ADAPT Go artifact with the REG-ADAPT guard, benchmark harness, exact-rank audit mode, digest-bound certification path, and recorded benchmark outputs added on the main branch.

For certifier-set activation, the artifact treats the certifier set as a signed activation input. During certification, REG installs the chosen old-epoch certifiers as ADAPT's signing set, signs the transition digest, verifies the aggregate certificate, and restores the previous signing set. The transition-level verifier checks that the certificate message is the expected digest, the certificate key is the old public key, the listed certifiers are the old-epoch certifiers chosen for the transition, their old-epoch weight is at least the old threshold, and the aggregate Schnorr/FROST certificate verifies. This is the implementation object corresponding to Cert_e in Theorem 2.

The benchmark driver reports three deployment modes. *Metadata* runs conservative envelope accounting and the local ADAPT update. *Online certificate* additionally signs and verifies the transition digest $H(S_e, S_{e+1}, \mathsf{tr})$ with the old-epoch ADAPT signing path. *Full certificate* includes nonce preprocessing before certificate signing. Exact-rank audit is measured separately from local ADAPT update cost.

N	t	guard	audit	local	online	full
20	10	18.31	2.38	8.36	15.94	12.56
40	20	15.44	3.79	23.64	34.95	35.52
80	40	18.46	7.13	137.25	176.02	190.64

Table 2: REG-ADAPT artifact timings. Values are medians on an Apple M4 host with Go 1.26.2. Guard and audit are in microseconds; local, online, and full are in milliseconds. Guard and audit columns measure metadata/rank logic; the remaining columns include the public ADAPT local update path and, where indicated, digest-bound old-epoch certification.

Each row reports medians. Wrapper activation timings use 120 independent activations from a fresh initial state: thirty repetitions of $WIncrease$, $TIncrease$, $WDecrease$, and $TDecrease$. Exact-rank timings use matching ledger states and exclude local ADAPT update cost. Mean and standard-deviation summaries are included with the artifact. The measurements support the compiler cost model. Conservative metadata checks and exact-rank audit are microsecond-scale at these sizes. The local ADAPT update dominates no-certificate activation, and the old-epoch transition certificate is the dominant cryptographic wrapper cost. Online and full certificate modes measure digest-bound certification without and with nonce preprocessing in the benchmark harness. These modes are not intended as directly comparable implementations of the same preprocessing state; their relative ordering is artifact- and cache-dependent, so the stable interpretation is the cost scale rather than a deterministic ordering.

The same implementation also reproduces the motivating reachable-state sequence. Starting from adversarial exposure $99 < 100$, the guard accepts $TIncrease : 100 \to 150$ and $WIncrease$:

$99 \to 149$, then rejects $TDecrease : 150 \to 100$ because the final rank-exposure envelope reaches the new threshold. This is the executable counterpart of the invariant in Theorem 1 and Theorem 2.

The artifact implements two envelope backends. The deployment fast path is the conservative row count

$$r_\perp + r_{\text{live}} + r_{\text{stale}} + r_{\text{tr}},$$

which is the public bound used for activation. The audit backend constructs GLI derivative rows over a finite field and computes their matrix rank. The row provider maps derivative shares to the same participant-coordinate convention as the public ADAPT artifact, which uses participant indices directly as GLI coordinates. The construction-level interface also supports nonzero Shamir point offsets. The compiler and ledger definitions are independent of this convention; only the concrete row provider changes. Public threshold-decrease information and transient information are represented as coefficient rows, matching the conservative ledger rule used by the activation guard. Conservative envelopes preserve safety; exact rank checks quantify tightness and can reduce unnecessary rejections in audit or dispute paths. The artifact includes a tightness example where a duplicated stale derivative row gives conservative rank 2 at threshold 2, while exact rank is 1; the conservative fast path rejects and the audit path identifies that the transition is rank-safe.

6 Discussion and Further Directions

The results above show that the transfer from locally valid weighted updates to sequence-level active security is controlled by interpreted row spans rather than by current-weight checks or operation-by-operation correctness alone. In ADAPT/GLI, the relevant rows are live derivative shares, stale derivatives removed from the current policy, owner-$\perp$ coefficients opened by threshold decrease, and transient rows created while old and new signing states coexist. The guard compiler does not change the local GLI algebra; it makes these reconstruction-relevant rows part of the public reconfiguration state machine and activates a transition only when the final and transient spans remain reconstruction-safe.

This row-span view also gives the deployment interpretation. A PoS, bridge, oracle, or custody system may want a stable verification key while stake, committee membership, or signing policy changes over time [8, 20, 19]. REG turns activation into a public check: old, pending, exiting, stale, public, and transient signing material must remain below the reconstruction threshold, and an old-epoch certificate must bind the accepted next-state digest. Expiry rules are therefore part of the security model, not metadata: in PoS they should track unbonding, slashing, erasure, and refresh-finality assumptions; in custody and governance they should track policy-delay and key-rotation evidence.

The REG-ADAPT construction realizes ledger soundness by mapping each scalar in the ADAPT update transcript to a private, stale, public, or transient ledger row while keeping commitments, hashes, complaints, and verification equations in the public-verification layer. This separation is what lets the same theorem cover the delicate ADAPT cases: masking-polynomial corrections in $WDecrease$ preserve the signing secret row because the masks have zero constant term, and transformed-share values in $TIncrease$ are recipient-owned transient rows whose derived derivatives remain in the span checked before activation. The system interface exposed by REG is therefore concrete: private delivery assumptions, complaint handling, identifiable aborts, transcript availability, adaptive erasures, concrete expiry rules, and atomic commit boundaries all determine how rows enter and leave $\mathsf{Rows}_S(F)$. A natural extension is rank-sensitive liveness: the artifact already includes exact-rank audit, and a production system could use dynamic rank data structures or

dispute certificates to accept transitions that conservative row-count envelopes reject only because of dependent stale or public rows.

References

[1] A. Shamir. How to Share a Secret. *Communications of the ACM*, 22(11):612–613, 1979.

[2] C.-P. Schnorr. Efficient Signature Generation by Smart Cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[3] P. Feldman. A Practical Scheme for Non-interactive Verifiable Secret Sharing. In *FOCS*, pages 427–438, 1987.

[4] T. P. Pedersen. A Threshold Cryptosystem without a Trusted Party. In *EUROCRYPT*, pages 522–526, 1991.

[5] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure Distributed Key Generation for Discrete-Log Based Cryptosystems. In *EUROCRYPT*, pages 295–310, 1999.

[6] A. Kate, G. M. Zaverucha, and I. Goldberg. Constant-Size Commitments to Polynomials and Their Applications. In *ASIACRYPT*, pages 177–194, 2010.

[7] R. Cramer, I. Damgård, and U. Maurer. General Secure Multi-Party Computation from any Linear Secret Sharing Scheme. In *EUROCRYPT*, pages 316–334, 2000. `https://eprint.iacr.org/2000/037`

[8] A. Kiayias, A. Russell, B. David, and R. Oliynykov. Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol. In *CRYPTO*, pages 357–388, 2017.

[9] D. Connolly, C. Komlo, I. Goldberg, and C. A. Wood. The Flexible Round-Optimized Schnorr Threshold (FROST) Protocol for Two-Round Schnorr Signatures. RFC 9591, 2024. `https://www.rfc-editor.org/rfc/rfc9591`

[10] C. Komlo and I. Goldberg. FROST: Flexible Round-Optimized Schnorr Threshold Signatures. In *Selected Areas in Cryptography*, 2021.

[11] T. Ruffing, V. Ronge, E. Jin, J. Schneider-Bensch, and D. Schroder. ROAST: Robust Asynchronous Schnorr Threshold Signatures. Cryptology ePrint Archive, Report 2022/550, 2022. `https://eprint.iacr.org/2022/550`

[12] S. K. D. Maram, F. Zhang, L. Wang, A. Low, Y. Zhang, A. Juels, and D. Song. CHURP: Dynamic-Committee Proactive Secret Sharing. In *ACM CCS*, 2019. `https://eprint.iacr.org/2019/017`

[13] K. M. Martin, J. Pieprzyk, R. Safavi-Naini, and H. Wang. Changing Thresholds in the Absence of Secure Channels. In *ACISP*, pages 177–191, 1999.

[14] C. Tartary and H. Wang. Dynamic Threshold and Cheater Resistance for Shamir Secret Sharing Scheme. In *Inscrypt*, pages 103–117, 2006.

[15] R. Steinfeld, J. Pieprzyk, and H. Wang. Lattice-Based Threshold-Changeability for Standard Shamir Secret-Sharing Schemes. *IEEE Transactions on Information Theory*, 53(7):2542–2559, 2007.

[16] A. Noack and S. Spitz. Dynamic Threshold Cryptosystem without Group Manager. Cryptology ePrint Archive, Report 2008/380, 2008. https://eprint.iacr.org/2008/380

[17] M. Nojoumian and D. R. Stinson. On Dealer-Free Dynamic Threshold Schemes. *Advances in Mathematics of Communications*, 7(1):39–56, 2013.

[18] J. Yuan and L. Li. A Fully Dynamic Secret Sharing Scheme. *Information Sciences*, 496:42–52, 2019.

[19] H. Wang, W. Ma, F. Deng, H. Zheng, and Q. Wu. Dynamic Threshold ECDSA Signature and Application to Asset Custody in Blockchain. *Journal of Information Security and Applications*, 61:102805, 2021.

[20] M. Y. Kubilay, M. S. Kiraz, and H. A. Mantar. KORGAN: An Efficient PKI Architecture Based on PBFT through Dynamic Threshold Signatures. *The Computer Journal*, 64(4):564–574, 2021.

[21] A. Cimatti, F. De Sclavis, G. Galano, S. Giammusso, M. Iezzi, A. Muci, M. Nardelli, and M. Pedicini. Dynamic-FROST: Schnorr Threshold Signatures with a Flexible Committee. Cryptology ePrint Archive, Report 2024/896, 2024. https://eprint.iacr.org/2024/896

[22] J. H. Choi, H. Jang, I. R. Jeong, and C. Lee. Enhancing Blockchain Proof of Stake with Active Weighted Signatures: The ADAPT Framework. Cryptology ePrint Archive, Report 2026/958, 2026. https://eprint.iacr.org/2026/958

[23] J. H. Choi, H. Jang, I. R. Jeong, and C. Lee. ADAPT: Active Weighted Signature Implementation. Public Go artifact, 2025. https://github.com/maskingidentity/ADAPT_imple

[24] S. Jo. REG-ADAPT Guard and Benchmark Artifact. Open-source Go artifact, 2026. https://github.com/ainta/ADAPT_imple

[25] P. Dikshit and K. Singh. Efficient Weighted Threshold ECDSA for Securing Bitcoin Wallet. In *ISEA Asia Security and Privacy*, pages 1–9. IEEE, 2017.

[26] L. Baird, S. Garg, A. Jain, P. Mukherjee, R. Sinha, M. Wang, and Y. Zhang. Threshold Signatures in the Multiverse. In *IEEE Symposium on Security and Privacy*, pages 1454–1470, 2023. https://eprint.iacr.org/2023/063

[27] S. Garg, A. Jain, P. Mukherjee, R. Sinha, M. Wang, and Y. Zhang. Cryptography with Weights: MPC, Encryption and Signatures. In *CRYPTO*, pages 295–327, 2023. https://eprint.iacr.org/2022/1632

[28] S. Garg, A. Jain, P. Mukherjee, R. Sinha, M. Wang, and Y. Zhang. hinTS: Threshold Signatures with Silent Setup. Cryptology ePrint Archive, Report 2023/567, 2023. https://eprint.iacr.org/2023/567

[29] A. Kate, P. Mukherjee, S. Samanta, and P. Sarkar. Dyna-hinTS: Silent Threshold Signatures for Dynamic Committees. Cryptology ePrint Archive, Report 2025/631, 2025. https://eprint.iacr.org/2025/631

[30] S. Das, P. Camacho, Z. Xiang, J. Nieto, B. Bünz, and L. Ren. Threshold Signatures from Inner Product Argument: Succinct, Weighted, and Multi-threshold. Cryptology ePrint Archive, Report 2023/598, 2023. https://eprint.iacr.org/2023/598

[31] A. Goel, S. Sasmal, and M. Wang. Weighted Cryptography with Weight-Independent Complexity. Cryptology ePrint Archive, Report 2026/273, 2026. https://eprint.iacr.org/2026/273