HSSA–STC Model Solution Notes

1 Introduction

We study a streaming trace commitment (STC) scheme, instantiated by HSSA, that commits to a long sequence $v \in \mathbb{F}^n$ in one pass and produces a compact commitment $C = (n, \mathsf{root}, \vec{r}, \vec{s})$. The scheme is transparent (hash-based), prioritizes binding and public verifiability over hiding, and supports light-client data-availability checks. Security is based on collision resistance of H and random-oracle (Fiat–Shamir) challenges for sketches; we also outline a ROM argument-of-knowledge view showing that any accepting commitment evidences possession of a consistent trace.

Informal TL;DR

- Streaming, transparent commitment to long traces with one-pass updates and small state. - Hash chain (Merkle-style) + m random polynomial sketches enforce global consistency. - Binding and ROM AoK (not hiding): $C = (n, \mathsf{root}, \vec{r}, \vec{s})$ leaks linear info via $\vec{s}$. - Soundness error $\approx ((n_{\max} - 1)/(p - 1))^m$ plus hash-collision term. - Data-availability friendly: random openings + GlobalCheck give probabilistic integrity. - Composes with IVC/PCD: use C_t as the evolving state; separate SNARK enforces transition correctness.

2 Basic Cryptographic Preliminaries

Definition 2.1 (Negligible function)**.** *A function $\mu \colon \mathbb{N} \to \mathbb{R}_{\geq 0}$ is* negligible *if for every polynomial $p(\cdot)$ there exists $N \in \mathbb{N}$ such that for all $\lambda \geq N$ we have $\mu(\lambda) \leq 1/p(\lambda)$. Intuitively, $\mu(\lambda)$ vanishes faster than the inverse of every polynomial.*

Definition 2.2 (Probabilistic polynomial time)**.** *An algorithm A is* probabilistic polynomial time (PPT) *if there exists a polynomial $p(\cdot)$ such that for every input x the running time of $A(x)$ is at most $p(|x|)$, even when A is allowed to use internal randomness. Adversaries appearing in our security notions are always assumed to be PPT unless stated otherwise.*

Definition 2.3 (Security experiment and advantage)**.** *Let $\mathrm{Exp}^{\mathcal{P}}_{\Pi}(A, 1^\lambda)$ be a security experiment for a primitive Π and property $\mathcal{P}$ (for example the binding of a commitment scheme). The* advantage *of A in this experiment is*

$$\mathrm{Adv}^{\mathcal{P}}_{\Pi}(A, \lambda) := \Pr[\mathrm{Exp}^{\mathcal{P}}_{\Pi}(A, 1^\lambda) = 1].$$

We say that Π satisfies property $\mathcal{P}$ if $\mathrm{Adv}^{\mathcal{P}}_{\Pi}(A, \lambda)$ is negligible for all PPT adversaries A.

3 Hash Functions and Collision Resistance

Definition 3.1 (Hash function family)**.** *A hash function family is a function $H \colon \{0,1\}^* \to \{0,1\}^n$ for some fixed output length n. Concrete instantiations include SHA-256 and BLAKE3.*

Definition 3.2 (Collision-resistant hash function)**.** *A hash function H is* collision-resistant *if for every PPT adversary A the quantity*

$$\mathrm{Adv}_H^{\mathsf{crh}}(A, \lambda) := \Pr\left[(x, x') \leftarrow A(1^\lambda) : x \neq x' \text{ and } H(x) = H(x')\right]$$

is negligible in λ. Collision resistance allows us to build a Merkle or hash-chain commitment to the chunk structure of a trace.

4 Commitment Schemes

Definition 4.1 (Non-interactive commitment scheme)**.** *A (non-interactive) commitment scheme $\Pi = (\mathsf{Com}, \mathsf{Open})$ consists of two PPT algorithms:*

- $\mathsf{Com}(1^\lambda, x)$ *outputs a pair (c, d), where c is the commitment and d is the decommitment string.*
- $\mathsf{Open}(1^\lambda, x, d)$ *outputs* 1 *iff d is a valid opening of c to value x.*

The scheme must satisfy correctness: if $(c, d) \leftarrow \mathsf{Com}(1^\lambda, x)$ then $\mathsf{Open}(1^\lambda, x, d) = 1$. We focus solely on binding and explicitly drop hiding for HSSA.

Experiment 4.2 (Binding experiment for commitments)**.**

1. *The adversary A outputs (x, x', d, d').*
2. *The challenger checks whether $x \neq x'$ and whether both $\mathsf{Open}(1^\lambda, x, d)$ and $\mathsf{Open}(1^\lambda, x', d')$ accept for the same commitment c.*
3. *The experiment outputs* 1 *if both openings are valid with $x \neq x'$, and* 0 *otherwise.*

The binding advantage of A is $\mathrm{Adv}_\Pi^{\mathsf{bind}}(A, \lambda) := \Pr[\mathrm{Exp}_\Pi^{\mathsf{bind}}(A, 1^\lambda) = 1]$.

5 Universal Hashing and Polynomial Hash

Definition 5.1 (Almost-universal hash family)**.** *Let $\mathcal{H} = \{H_k \colon \mathcal{X} \to \mathcal{Y}\}_{k \in \mathcal{K}}$. The family is* universal *if for all distinct $x, x' \in \mathcal{X}$,*

$$\Pr_{k \leftarrow \mathcal{K}}[H_k(x) = H_k(x')] \leq \frac{1}{|\mathcal{Y}|}.$$

It is ε-almost-universal if the probability is bounded by ε. We instantiate $\mathcal{H}$ via polynomial hash over a finite field.

Construction 5.2 (Polynomial hash over $\mathbb{F}_p$)**.** *Fix a prime p and maximum trace length N. Define the domain $\mathcal{X} = \mathbb{F}_p^N$, range $\mathcal{Y} = \mathbb{F}_p$, and keys $k = r \in \mathbb{F}_p \setminus \{0\}$. For $v \in \mathbb{F}_p^N$ set*

$$H_r(v) := \sum_{i=0}^{N-1} v[i] r^i \in \mathbb{F}_p.$$

This is the streaming sketch $s = \sum_i v[i] r^i$ used throughout STC.

6 STC/HSSA Scheme Definition

Terminology. We use *Finalize* as the algorithm name that outputs the public commitment (often called *Commit*); both terms are synonymous here.

Remark 6.1 (Streaming model). Streaming is an *efficiency* requirement for honest committers, not a restriction on adversaries. An honest committer processes a trace $v = (v_0, \dots, v_{n-1})$ in a single pass via

$$(\mathsf{pp}, \mathsf{st}_0) \leftarrow \mathsf{Init}(1^\lambda), \quad \mathsf{st}_{i+1} \leftarrow \mathsf{Update}(\mathsf{pp}, \mathsf{st}_i, v_i), \quad C \leftarrow \mathsf{Finalize}(\mathsf{pp}, \mathsf{st}_n),$$

with state size polylogarithmic in n (e.g., n, root, and a constant-size sketch accumulator). Security is defined against *arbitrary* PPT adversaries without streaming/memory limits.

We detail the public algorithms of the STC scheme with explicit inputs, outputs, and procedures. Randomness is fixed either during setup or via Fiat–Shamir from public context; updates are deterministic.

Algorithm 6.2 (Setup)**.** ***Input:*** *Security parameter* 1^λ *and context string.*

Output: *Public parameters* $\mathsf{pp} = (\mathbb{F}, H, L, m, ctx)$*, where* $\mathbb{F} = \mathbb{F}_p$ *is a prime field,* $H : \{0,1\}^* \to \{0,1\}^b$ *is a CRH, L is the maximum chunk length,* $m \geq 1$ *is the number of sketch challenges, and ctx binds domain separation.*

Procedure:

1. *Choose a prime p of size* $\geq 2^{\kappa(\lambda)}$ *for some growth function* κ*; set* $\mathbb{F} \leftarrow \mathbb{F}_p$*.*
2. *Fix a collision-resistant hash H (e.g., SHA-256 truncated to b bits) and domain separators.*
3. *Fix chunk bound L and challenge count m (both polynomial in* λ *and context-specific).*
4. *Return* $\mathsf{pp} = (\mathbb{F}, H, L, m, ctx)$*.*

Algorithm 6.3 (Init)**.** ***Input:*** *Public parameters* pp*.*

Output: *Initial state* st_0*.*

Procedure:

1. *Set* $n \leftarrow 0$*.*
2. *Set* $\mathsf{root} \leftarrow H(\text{``bef-init''} \| ctx)$*.*
3. *Derive challenges* $\vec{r} = (r_0, \dots, r_{m-1})$ *via Fiat–Shamir from* root*:*

$$r_j \leftarrow \mathsf{FE}\big(H(\mathsf{root} \| \text{``bef-challenge''} \| j)\big), \quad r_j \neq 0.$$

4. *Set sketches* $\vec{s} \leftarrow (0, \dots, 0)$ *and cached powers* $\vec{\mathsf{pow}} \leftarrow (1, \dots, 1)$*.*
5. *Output* $\mathsf{st}_0 = (n, \mathsf{root}, \vec{r}, \vec{s}, \vec{\mathsf{pow}})$*.*

Algorithm 6.4 (Update)**.** ***Input:*** pp*, state* $\mathsf{st} = (n, \mathsf{root}, \vec{r}, \vec{s}, \vec{\mathsf{pow}})$*,*
and a chunk chunk $= (x_0, \dots, x_{\ell-1}) \in \mathbb{F}^\ell$ *with* $\ell \leq L$*.*

Output: *Updated state* st'*.*

Determinism: *No fresh randomness; this step is fully deterministic given inputs.*

Procedure:

1. *Let* $\mathsf{offset} \leftarrow n$*.*

2. *Compute chunk root* $\mathsf{root}_{chunk} \leftarrow H(\mathsf{offset}\|\textit{chunk-bytes})$ *(or Merkle root over the chunk leaves).*

3. *Update running root:*

$$\mathsf{root}' \leftarrow H(\mathsf{root}\|\texttt{``bef-chunk''}\|\mathsf{encode}(\mathsf{offset})\|\mathsf{root}_{chunk}).$$

4. *For each* $j \in \{0, \ldots, m-1\}$ *with* $(r, s, \mathsf{pow}) \leftarrow (r_j, s_j, \mathsf{pow}_j)$*, iterate elements in order:*

$$s \leftarrow s + x \cdot \mathsf{pow}, \qquad \mathsf{pow} \leftarrow \mathsf{pow} \cdot r.$$

   Set $(s_j, \mathsf{pow}_j) \leftarrow (s, \mathsf{pow})$.

5. *Set* $n \leftarrow n + \ell$ *and* $\mathsf{root} \leftarrow \mathsf{root}'$*; output the new state.*

Algorithm 6.5 (Finalize)**.** ***Input:*** pp *and a terminal state* $\mathsf{st} = (n, \mathsf{root}, \vec{r}, \vec{s}, \vec{\mathsf{pow}})$.
Output: *Commitment* $C = (n, \mathsf{root}, \vec{r}, \vec{s})$ *and optional metadata* meta.
Procedure: *Return* $C = (n, \mathsf{root}, \vec{r}, \vec{s})$*; keep* $\vec{\mathsf{pow}}$ *internal. Metadata may include per-chunk offsets, roots, and pre-shifted chunk sketches for fast verification.*

Algorithm 6.6 (Open)**.** ***Input:*** pp*, a state* st *associated with trace* v*, and an index* $i \in \{0, \ldots, n-1\}$.
Output: *Opening proof* π_i *for the claim "*$v[i] = a$*" where* $a \in \mathbb{F}$.
Procedure:

1. *Identify the chunk* t *and intra-chunk position for* i*; set* $a \leftarrow v[i]$.

2. *Provide the Merkle path from the leaf for* a *to* $\mathsf{root}_{chunk,t}$*, and the path from* $\mathsf{root}_{chunk,t}$ *through the hash chain to* root.

3. *Output* $\pi_i = (a, t, \textit{Merkle/chain paths})$.

Algorithm 6.7 (VerifyOpen)**.** ***Input:*** pp*, commitment* $C = (n, \mathsf{root}, \vec{r}, \vec{s})$*, index* i*, value* $a \in \mathbb{F}$*, and proof* π_i.
Output: 1 *if valid, else* 0.
Procedure: *Deterministically recompute the leaf and internal hashes from* π_i *and accept iff the resulting root equals* root *in* C.

Algorithm 6.8 (GlobalCheck)**.** ***Input:*** pp*, commitment* $C = (n, \mathsf{root}, \vec{r}, \vec{s})$*, and metadata* $\mathsf{meta} = \{(\mathsf{offset}_t, \mathsf{length}_t, \mathsf{root}_{chunk,t}, \mathsf{sketch\_vec}_t)\}_{t=0}^{K-1}$.
Output: 1 *if consistent, else* 0.
Checks:

1. *Coverage: chunks start at* 0*, have no gaps/overlaps, and total length equals* n.

2. *Root recomputation: starting with* $\mathsf{root}' \leftarrow H(\texttt{``bef-init''}\|ctx)$*, iteratively update*

$$\mathsf{root}' \leftarrow H(\mathsf{root}'\|\mathsf{encode}(\mathsf{offset}_t)\|\mathsf{root}_{chunk,t})$$

   and require $\mathsf{root}' = \mathsf{root}$.

3. *Sketch recomputation: for each* j*, set* $\hat{s}_j \leftarrow \sum_t \mathsf{sketch\_vec}_t[j]$ *and require* $\hat{s}_j = s_j$.

Accept iff all checks pass. This ensures C *plausibly arises from a single trace consistent with* meta.

Remark 6.9 (Construction of $\mathsf{meta}(v)$). The committer builds $\mathsf{meta}(v)$ with per-chunk sketches pre-shifted by $r_j^{\mathsf{offset}_t}$ so that global sketches add up directly: $\hat{s}_j = \sum_t \mathsf{sketch\_vec}_t[j]$. This matches the recomputation in GlobalCheck.

Lemma 6.10 (Collision bound via Schwartz–Zippel). *Let $v, v' \in \mathbb{F}_p^N$ with $v \neq v'$ and define $e = v - v'$. Consider the polynomial $E(X) = \sum_{i=0}^{N-1} e[i]X^i$. If $e \neq 0$ then $\deg(E) \leq N - 1$ and for uniformly random $r \in \mathbb{F}_p \setminus \{0\}$ we have*

$$\Pr_r[H_r(v) = H_r(v')] = \Pr_r[E(r) = 0] \leq \frac{N-1}{p-1}.$$

With m independent challenges $r_0, \ldots, r_{m-1}$ this probability is at most $\big((N-1)/(p-1)\big)^m$.

7 Streaming Trace Commitment (STC)

Definition 7.1 (Streaming Trace Commitment). *Let $\mathbb{F}$ be a prime field of size p. An STC scheme over traces $v \in \mathbb{F}^n$ consists of PPT algorithms*

$$(\mathsf{Setup}, \mathsf{Init}, \mathsf{Update}, \mathsf{Finalize}, \mathsf{Open}, \mathsf{VerifyOpen}, \mathsf{GlobalCheck})$$

with the following behavior:

- $\mathsf{Setup}(1^\lambda)$ *outputs public parameters*

  $$\mathsf{pp} = (\mathbb{F}, H, L, m, \ldots)$$

  containing the field, a collision-resistant hash H, a chunk size bound L, and the number m of sketch challenges.
- $\mathsf{Init}(\mathsf{pp})$ *outputs an initial state* st_0.
- $\mathsf{Update}(\mathsf{pp}, \mathsf{st}, \mathit{chunk})$ *ingests a chunk $\mathit{chunk} \in \mathbb{F}^\ell$ with $\ell \leq L$ and returns an updated state* st'.
- $\mathsf{Finalize}(\mathsf{pp}, \mathsf{st})$ *on the final state outputs a public commitment C and optional metadata.*
- $\mathsf{Open}(\mathsf{pp}, \mathsf{st}, i)$ *derives proof material π_i for position i.*
- $\mathsf{VerifyOpen}(\mathsf{pp}, C, i, v[i], \pi_i)$ *deterministically verifies that $(i, v[i])$ is consistent with C.*
- $\mathsf{GlobalCheck}(\mathsf{pp}, C, \mathsf{meta})$ *uses per-chunk metadata to run a probabilistic check that all chunks bind to a unique trace.*

Correctness demands that honest traces produce valid openings and that GlobalCheck *accepts with probability* 1. *The security goals drop hiding; we focus on binding / trace integrity and fast global sketch soundness.*

8 HSSA Construction inside STC

Construction 8.1 (HSSA–STC). *Let $\mathbb{F} = \mathbb{F}_p$ with $p = 2^{61} - 1$ in the prototype. Traces $v \in \mathbb{F}^N$ are chunked into lengths at most L. Let $H\colon \{0,1\}^* \to \{0,1\}^b$ be a collision-resistant hash and let m challenges $r_0, \ldots, r_{m-1} \in \mathbb{F} \setminus \{0\}$ be derived via Fiat–Shamir.*

The internal state is

$$\mathsf{st} = \big(n, \mathsf{root}, (r_j)_{j=0}^{m-1}, (s_j)_{j=0}^{m-1}, (\mathsf{pow}_j)_{j=0}^{m-1}\big),$$

where n counts elements processed, root *is the hash-chain commitment to chunk roots, $s_j = \sum_{i=0}^{n-1} v[i] r_j^i$ are sketches, and $\mathsf{pow}_j = r_j^n$ cache streaming powers.*

Initialization. *Set $n \leftarrow 0$ and $\mathsf{root} \leftarrow H(\texttt{``bef-init''} \| context)$. For each j sample r_j via Fiat–Shamir with rejection to avoid 0 and set $s_j \leftarrow 0$, $\mathsf{pow}_j \leftarrow 1$.*

Update. *Given a chunk $chunk = (x_0, \dots, x_{\ell-1})$ with $\ell \le L$:*

a. *Let $\mathsf{offset} = n$ and compute the chunk root $\mathsf{root}_{chunk} = H(\mathsf{offset} \| chunk\ bytes)$ (or Merkle root).*

b. *Update $\mathsf{root}' = H(\mathsf{root} \| \texttt{``bef-chunk''} \| \mathsf{encode}(\mathsf{offset}) \| \mathsf{root}_{chunk})$.*

c. *For each j iterate through chunk, updating $s_j \leftarrow s_j + x \cdot \mathsf{pow}_j$ and $\mathsf{pow}_j \leftarrow \mathsf{pow}_j \cdot r_j$ for each element.*

d. *Set $n \leftarrow n + \ell$ and $\mathsf{root} \leftarrow \mathsf{root}'$.*

Finalize. *Output $C = (n, \mathsf{root}, (r_j)_j, (s_j)_j)$; the powers remain internal.*

Open and Verify. *To open position i, provide $v[i]$, the chunk index, the Merkle path from the leaf to the chunk root, and the path from the chunk root through the hash chain to* root*. Verification recomputes the hashes and checks equality with the committed root. Sketch data are not involved in local verification.*

Global check. *Given C and metadata $\mathsf{meta} = \{(\mathsf{offset}_t, \mathsf{length}_t, \mathsf{root}_{chunk,t}, \mathsf{sketch\_vec}_t)\}_{t=0}^{K-1}$:*

a. *Perform a coverage check by sorting chunks by offset, ensuring they start at 0, have no gaps or overlaps, and sum to n.*

b. *Recompute root' by iterating through the ordered chunks, hashing*

$$\mathsf{root}' \leftarrow H(\mathsf{root}' \| \mathsf{encode}(\mathsf{offset}_t) \| \mathsf{root}_{chunk,t})$$

at each step, and require $\mathsf{root}' = \mathsf{root}$ at the end.

c. *Recompute sketches by setting $\hat{s}_j := \sum_t \mathsf{sketch\_vec}_t[j]$ (chunk sketches are assumed to include the appropriate $r_j^{\mathsf{offset}_t}$ shift) and require $\hat{s}_j = s_j$ for every j.*

Accept iff all checks pass.

9 Security Analysis: Binding and Soundness

Experiment 9.1 (HSSA–STC binding)**.**

1. *Run $\mathsf{Setup}(1^\lambda)$ to obtain public parameters containing $\mathbb{F}_p$, the hash function H, and other values.*

2. *The adversary outputs a claimed trace length n, two distinct traces $v, v' \in \mathbb{F}_p^n$, and metadata sets $\mathsf{meta}, \mathsf{meta}'$ describing chunk structure, chunk roots, and per-chunk sketches.*

3. *The challenger derives the commitment*

$$C = (n, \mathsf{root}, \vec{r}, \vec{s})$$

by honestly running $(\mathsf{Init}, \mathsf{Update}, \mathsf{Finalize})$ on (v, meta) while treating H as a random oracle for challenge derivation.

4. Run $\mathsf{GlobalCheck}(\mathsf{pp}, C, \mathsf{meta})$ *and* $\mathsf{GlobalCheck}(\mathsf{pp}, C, \mathsf{meta}')$.

The experiment outputs 1 *iff* $v \neq v'$ *and both global checks accept under the same commitment* C*. The binding advantage is* $\mathrm{Adv}^{\mathsf{bind}}_{HSSA}(A, \lambda) := \Pr[\mathrm{Exp}^{\mathsf{bind}}_{HSSA}(A, 1^{\lambda}) = 1]$.

Theorem 9.2 (HSSA binding bound)**.** *Assume* H *is a collision-resistant hash with output length* b*, the challenges* r_j *are derived from the root via Fiat–Shamir and modeled as independent uniform samples in* $\mathbb{F}_p \setminus \{0\}$ *(random-oracle model), and the system enforces a maximum trace length* $n_{\max}(\lambda)$*. Then for every PPT adversary* A *there exists a PPT adversary* A' *against the collision resistance of* H *such that*

$$\mathrm{Adv}^{\mathsf{bind}}_{HSSA}(A, \lambda) \leq \mathrm{Adv}^{\mathsf{crh}}_{H}(A', \lambda) + \left(\frac{n_{\max}(\lambda) - 1}{p - 1}\right)^{m}.$$

Proof sketch. Suppose A wins the binding experiment. Then it outputs distinct traces $v \neq v'$ together with metadata $\mathsf{meta}, \mathsf{meta}'$ so that both pass $\mathsf{GlobalCheck}$ under the same commitment C.

If the chunk root sequence or structure inferred from meta and meta' differs, but the final root inside C matches, then A yields a collision in the hash chain built from H. This case contributes the $\mathrm{Adv}^{\mathsf{crh}}_{H}(A', \lambda)$ term.

Otherwise the chunk structure and chunk roots must coincide, so the difference lies solely inside the field elements of the leaves. Define the error vector $e = v - v' \in \mathbb{F}_p^n$ and the polynomial $E(X) = \sum_{i=0}^{n-1} e[i]X^i$, which is non-zero and has degree at most $n - 1 \leq n_{\max} - 1$. Because the sketches inside C agree for both traces, $E(r_j) = 0$ for every challenge j. By Lemma 6.10 and independence of the r_j, the probability that this holds without already causing a hash collision is at most $\big((n_{\max} - 1)/(p - 1)\big)^m$. Combining the two cases yields the claimed bound. □

Corollary 9.3 (Single- vs multi-challenge soundness)**.** *If* $m = 1$*, the sketch collision term simplifies to* $\Pr[\textit{sketch-collision}] \leq (n_{\max} - 1)/(p - 1) = \Theta(n_{\max}/|\mathbb{F}|)$*. For general* m*, the probability scales as* $\big(\Theta(n_{\max}/|\mathbb{F}|)\big)^m$ *assuming independent challenges.*

Intuition. Distinct traces induce a nonzero error polynomial $E(X)$ of degree at most $n_{\max} - 1$. Independent random challenges r_j act as identity tests for E: each r_j is a random field point where a nonzero degree-($\leq n_{\max} - 1$) polynomial vanishes with probability at most $(n_{\max} - 1)/(p - 1)$. Hash collisions account for the remaining failure mode.

10 Application: Data Availability via HSSA–STC

The binding theorem enables a straightforward sampling-based data-availability (DA) protocol that mirrors KZG-style DA but swaps in HSSA–STC as the commitment layer.

10.1 Protocol overview

Participants are as follows.

- **Sequencer** $\mathcal{S}$: produces a blob B, encodes it as $v \in \mathbb{F}_p^n$, and runs the STC algorithms to create a commitment.
- **Full nodes**: store the entire blob, chunk metadata, and Merkle trees, answering opening queries via Open.

- **Light clients**: observe on-chain data (the commitment and metadata) and request random openings.

The protocol for a single blob proceeds:

1. *Finalize.* Split v into chunks $(\text{chunk}_t)_{t=0}^{T-1}$ of length at most L, run $(\mathsf{Init}, \mathsf{Update})$ sequentially, then run $\mathsf{Finalize}$ to publish $C = (n, \mathsf{root}, \vec{r}, \vec{s})$ with metadata (chunk offsets, chunk roots, per-chunk sketches).

2. *Sampling queries.* Each light client samples k chunk indices uniformly at random, fetches those chunks and accompanying Merkle paths from some full node, and runs $\mathsf{VerifyOpen}$ on each.

3. *Global check.* Clients optionally run $\mathsf{GlobalCheck}(\mathsf{pp}, C, \mathsf{meta})$ on the posted metadata to ensure coverage, root consistency, and agreement with $\vec{s}$.

4. *Decision.* If every sampled opening verifies and the global check (if run) accepts, the client accepts the blob; otherwise it rejects.

10.2 Security sketch

Let δ denote the fraction of chunks that are invalid or withheld, and let each client sample k chunks independently and uniformly. An adversary causes an honest client to accept only if all of the following occur simultaneously:

a. **Sampling miss.** The client avoids every bad chunk, which happens with probability at most $(1-\delta)^k$.

b. **Sketch collision.** Two distinct traces $v \neq v'$ of length at most $n_{\max}$ yield the same sketches. Lemma 6.10 and the argument of Theorem 9.2 bound this by $\big((n_{\max}-1)/(p-1)\big)^m$.

c. **Hash collision.** The metadata correspond to different chunk roots yet produce the same root, violating collision resistance of H; this occurs with probability at most $\mathrm{Adv}_H^{\mathsf{crh}}(A', \lambda)$ for some PPT adversary A'.

Union bounding gives the overall cheating probability

$$\mathrm{Adv}_{\mathrm{HSSA}}^{\mathsf{DA}}(\lambda) \le \mathrm{Adv}_H^{\mathsf{crh}}(A', \lambda) + (1-\delta)^k + \left(\frac{n_{\max}(\lambda)-1}{p-1}\right)^m .$$

Choosing parameters so that each term is negligible yields standard DA guarantees for light clients.

11 Hiding (or Lack Thereof)

The commitment $C = (n, \mathsf{root}, \vec{r}, \vec{s})$ is *not* hiding. The sketch vector $\vec{s}$ consists of random linear combinations of the trace entries, which leak global linear information (e.g., sums and other relations). While these do not directly reveal all coordinates, structured traces can leak nontrivial information. Achieving hiding would require augmentations such as hashing/salting elements prior to sketching or using a hiding commitment layer per entry/chunk, potentially sacrificing linearity and streaming simplicity. In this work we prioritize binding, streaming updates, and public verifiability over hiding.

Zero-knowledge (or lack thereof). In the niZK sense of Katz–Lindell, HSSA–STC is not zero-knowledge: (C, π) distributions for different traces are information-theoretically distinguishable due to $\vec{s}$. Thus no simulator can produce proofs indistinguishable from real ones (Attack Game 20.3) while answering RO queries consistently. This is intentional: we aim for existence/consistency (AoK) and public auditability, not privacy.

12 Argument of Knowledge in the ROM

We outline a ROM-based extractor showing that a prover producing a valid commitment C that passes GlobalCheck must "know" a consistent trace v (or else cause a hash collision).

Scope. This AoK guarantees *existence and global consistency of a trace* underlying C; it does not prove that the trace encodes a correct computation. For correctness, STC should be combined with a SNARK/IVC layer that enforces the transition logic.

Definition 12.1 (AoK experiment for HSSA–STC)**.** *Let R_{stream} be as in Definition 12.4. Consider PPT adversaries P^H (a prover with access to the random oracle H) and PPT extractors $E^{P^H,H}$. The experiment* $\mathrm{Exp}^{\mathsf{AoK}}_{\mathsf{HSSA}}(P, E, 1^\lambda)$ *proceeds:*

1. $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$.
2. $(C, \mathsf{meta}) \leftarrow P^H(1^\lambda, \mathsf{pp})$.
3. *If* $\mathsf{GlobalCheck}^H(\mathsf{pp}, C, \mathsf{meta}) = 0$, *output* 0.
4. $\mathsf{out} \leftarrow E^{P^H,H}(1^\lambda, \mathsf{pp}, C, \mathsf{meta})$ *where* $\mathsf{out} \in \{\mathsf{fail},\ (\mathsf{witness}, v),\ (\mathsf{coll}, (x, x'))\}$.
5. *Output* 1 *iff* $\mathsf{out} = \mathsf{fail}$ *or* $\mathsf{out} = (\mathsf{witness}, v)$ *but* $(C, v) \notin R_{stream}$, *and* $\mathsf{out} \neq (\mathsf{coll}, (x, x'))$; *otherwise output* 0.

Define the advantage

$$\mathrm{Adv}^{\mathsf{AoK}}_{\mathsf{HSSA}}(P, E, \lambda) := \Pr\left[\mathrm{Exp}^{\mathsf{AoK}}_{\mathsf{HSSA}}(P, E, 1^\lambda) = 1\right].$$

We say HSSA–STC is a non-interactive argument of knowledge for R_{stream} in the ROM if for every PPT P^H there exists a PPT $E^{P^H,H}$ such that

$$\mathrm{Adv}^{\mathsf{AoK}}_{\mathsf{HSSA}}(P, E, \lambda) \leq \mathrm{Adv}^{\mathsf{crh}}_{H}(A', \lambda) + \mathrm{negl}(\lambda)$$

for some PPT collision finder A' against H.

Lemma 12.2 (Coverage under random challenges)**.** *Fix $n \leq n_{\max}(\lambda)$. Let each transcript reveal at least one uniformly random index in $\{0, \ldots, n-1\}$. Then there exists a polynomial $T(\lambda, \log n_{\max})$ such that after T accepting forks with independent challenges,*

$$\Pr\Big[\bigcup_{t=1}^{T} Q^{(t)} \neq \{0, \ldots, n-1\}\Big] \leq \mathrm{negl}(\lambda).$$

Sketch. *Coupon-collector: $O(n \log n)$ random hits cover all indices with probability $1 - n^{-c}$ for any constant c; take $n_{\max} = \mathsf{poly}(\lambda)$.*

Remark 12.3 (Hash as RO and CRH)*.* In the ROM analysis we use the same H both as a collision-resistant hash in the Merkle chain and as the Fiat–Shamir oracle for deriving challenges. Following standard FS proofs, we program H when extracting; any successful adversary that makes $\mathsf{GlobalCheck}$ accept without a witness yields either an RO-fork that enables extraction or a collision for H.

Definition 12.4 (Stream relation R_{stream})**.** *Over parameters* pp*, define*

$$R_{stream} = \big\{(C, v) : C = \mathsf{Finalize}(\mathsf{pp}, \mathsf{st}) \text{ for the unique state from streaming } v \text{ via } \textsc{Init}/\textsc{Update}, \quad \text{and } \textsc{GlobalCheck}(\mathsf{pp}, C, \mathsf{meta}(v)) = 1\big\}.$$

Theorem 12.5 (AoK for streaming trace commitments)**.** *Under collision resistance of H and in the ROM for Fiat–Shamir, for every PPT P^H there exists a PPT extractor $E^{P^H,H}$ such that*

$$\mathrm{Adv}^{\mathsf{AoK}}_{\mathsf{HSSA}}(P, E, \lambda) \leq \mathrm{Adv}^{\mathsf{crh}}_{H}(A', \lambda) + \mathrm{negl}(\lambda).$$

In particular, HSSA–STC forms a non-interactive argument of knowledge for R_{stream} in the ROM.

Extractor outline.

1. Invoke $\mathcal{P}$ to get (n, root); program RO to challenge $c_1 = (\vec{r}^{(1)}, Q^{(1)})$; obtain accepting $(\vec{s}^{(1)}, \{\pi_i^{(1)}\})$.
2. Rewind to just after (n, root); program RO to $c_2 = (\vec{r}^{(2)}, Q^{(2)})$; obtain accepting $(\vec{s}^{(2)}, \{\pi_i^{(2)}\})$.
3. Aggregate opened positions $\bigcup_k Q^{(k)}$ and their values/proofs. If coverage is complete, reconstruct v and verify consistency; output (v, meta).
4. If conflicting openings occur or sketch systems disagree, derive a hash collision or a polynomial-identity violation; output a break.
5. Otherwise repeat forks until coverage suffices (polynomially many with high probability) and extract v.

12.1 Interactive protocol and Fiat–Shamir

Consider a public-coin three-move protocol for R_{stream}:

1. Prover sends C and metadata.
2. Verifier samples a random challenge (e.g., a seed defining positions and/or sketch coefficients).
3. Prover responds with the requested openings and/or linear checks; verifier runs $\textsc{VerifyOpen}/\textsc{GlobalCheck}$.

Applying Fiat–Shamir (hashing the prover's message to derive the challenge) yields a non-interactive proof π attached to C. In practice, π consists of the metadata meta and the responses to FS-derived challenges (the sketch vector $\vec{s}$ and the requested openings). The verifier recomputes the challenge from H, runs $\textsc{VerifyOpen}$ on the sampled positions, and then runs $\textsc{GlobalCheck}$ on $(\mathsf{pp}, C, \mathsf{meta})$. In the ROM, standard rewinding then gives the extractor above.

Definition 12.6 (STC–AoK)**.** *Let Π_{STC} be the three-move protocol above and let challenges be derived as $c \leftarrow H(\mathsf{pp}, C, \mathrm{tag})$. Define the non-interactive system* $\mathsf{STC\text{-}AoK} = (\mathsf{Setup}, \mathsf{GenPrf}, \mathsf{VrfyPrf})$*, where* GenPrf *runs the prover with FS and* $\mathsf{VrfyPrf}$ *runs* $\textsc{GlobalCheck}$ *on the induced transcript. By Theorem 12.5 and Definition 12.1,* $\mathsf{STC\text{-}AoK}$ *is a non-interactive* argument of knowledge *for R_{stream} in the ROM.*

Remark 12.7 (Argument vs proof). Security relies on computational assumptions (CRH and polynomial soundness), hence we obtain a computational *argument* of knowledge, not a statistical/perfect proof of knowledge.

12.2 Composition with IVC/PCD

In recursive systems, maintain a commitment C_t as the state at step t. Each step proves (inside a SNARK) that (i) the transition from state $t-1$ to t is correct, and (ii) C_t is a valid FINALIZE/UPDATE of C_{t-1} with the step's outputs. Thus STC serves as the integrity/availability backbone, while the SNARK enforces computation correctness. The propagated object is the succinct C_t (not the raw trace), and the final verifier learns a compact proof plus C_T binding the full execution.

13 Positioning and Comparison

Merkle commitments. Transparent, collision-resistance based, succinct root, hiding by default; less streaming-friendly without additional mechanisms; proofs scale with $\log n$.

KZG polynomial commitments. Very short commitments and proofs; can be hiding; require trusted setup and pairings; algebraic assumptions; strong batch properties.

HSSA–STC. Transparent and hash-based like Merkle; uses algebraic random sketches akin to polynomial checks; supports one-pass streaming; not hiding; commitment and metadata include a sketch vector of length m and per-chunk structure.

STC sits between Merkle and KZG: transparent and simple, with algebraic integrity checks, trading succinctness and hiding for streaming and minimal assumptions.

14 Summary

HSSA–STC is a streaming, transparent commitment scheme for long traces. It provides computational binding with sketch soundness $\big(\Theta(n_{\max}/|\mathbb{F}|)\big)^m$, integrates cleanly with DA sampling, and (under ROM) yields a non-interactive argument of knowledge for the existence of a consistent trace. It is not hiding by design; achieving hiding would require masking that may forfeit linearity and streaming simplicity. These properties make STC a good integrity backbone to compose with SNARK/IVC systems.