ZK-DABE

Zero-Knowledge and Ledger-Anchored Multi-Authority ABE with Negations and Immediate Revocation for Subsequent Issuances

Johan B

2025-02-02 – v2.7

Abstract

I present *ZK-DABE*, a concrete way to combine multi-authority attribute-based encryption (MA-ABE) with zero-knowledge state proofs that are tied to a tamper-evident ledger. Classical ABE gives expressive access control [1], [2], MA-ABE distributes trust [3], [4], and non-monotonic ABE supports negations [5], but deployed systems still lack a practical way to enforce `NOT` conditions across organizations, make revocation take effect immediately for *subsequent issuances of helper material* without re-encrypting old ciphertexts, and keep audit trails without exposing attribute metadata (cf. BDABE [6]). ZK-DABE addresses this by binding proofs to per-authority state roots on a permissioned BFT ledger [7], [8], [9], by gating decryption with short, per-ciphertext and header-bound tokens and authority shares that simply stop *issuing* after revocation, and by logging commitments instead of plaintexts. Two core ideas make this workable in practice: a global satisfiability transcript τ that ties cross-authority OR and threshold policies through per-authority projection commitments, and a short on-ledger session identifier `sid` that binds a single global transcript to all authorities so tokens from different branches or sessions cannot be mixed; the session identifier is also bound inside the algebra of released authority shares. Token derivation uses a PRF/VRF under authority keys. I define the model, protocols, and security properties in a backend-agnostic way for both the CP-ABE and ZK components (e.g., STARKs [10]) and argue confidentiality, immediate revocation for subsequent *issuances*, and privacy under standard assumptions. A *zk-VRF commit-and-prove* path (*VRF-inside-ZK*) lets public verifiers check that a well-formed issuance consistent with a posted commitment occurred, without revealing the VRF inputs.

1 Introduction

Modern zero-trust architectures assume breach, continuously verify claims, and refuse to treat network location or a single identity provider as sufficient evidence of authorization [11], [12]. Attribute-Based Encryption (ABE) [1], [2] naturally supports predicates over attributes, while Multi-Authority ABE (MA-ABE) [3], [4] reflects the fact that attributes live in different administrative domains. Non-monotonic ABE [5] extends expressiveness by adding negation.

Three practical gaps remain. First, enforcing `NOT` conditions across organizations without revealing unrelated attributes is difficult: embedding `NOT` directly in CP-ABE ciphertexts inflates headers and leaks structure, and asking one authority to publish negative facts about another authority's subjects is rarely acceptable. Second, revocation usually relies on periodic key updates, attribute expiry, or re-encryption; these are either coarse or operationally heavy and do not give the simple guarantee that attempts to obtain *helper material* for existing ciphertexts should fail immediately after a revocation event. Third, auditors need tamper-evident trails, but naive logging of access decisions or attribute assertions exposes metadata and undermines the privacy that motivated ABE in the first place.

The design here separates concerns. Positive conditions remain inside a standard CP-ABE scheme. Encryptors publish ordinary BSW07 headers and are unaware of any token material. Negations and revocation are enforced at the point where a reader tries to use a ciphertext: the reader proves membership or non-membership in zero knowledge against per-authority state roots anchored to a Byzantine fault tolerant ledger with finality, and each authority issues a short, per-ciphertext and header-bound token plus an encrypted authority contribution that is needed to complete decryption. Authorities stop *issuing* this helper

material once a revocation is finalized, so later attempts fail even for old ciphertexts. A single global proof of satisfiability produces a transcript identifier τ and per-authority projection commitments. A pre-commit session sid written to the ledger binds that transcript to all authorities for a given $(\mathsf{cid}, \mathsf{H}(C_0), \mathsf{EncGT}(B_u))$, which prevents mixing tokens from different branches when the policy contains OR or threshold structure. The session id is also bound inside the algebra of the released shares so that shares issued under different sessions cannot be combined offline.

The contributions of this work can be summarized informally as follows. ZK-DABE uses zero-knowledge (non-)membership proofs at public roots to enforce `NOT` conditions across authorities without changing ciphertexts. Revocation is expressed as "no more helper issuance after this root," which gives immediate effect for subsequent attempts without re-encryption. A global transcript τ and a short on-ledger session sid tie all authority decisions for a given attempt to one witness and one ciphertext header, and sid is embedded algebraically in the released shares so that cross-session mixing fails. A concrete BSW07-based instantiation shows how users hold only an α_0 share while authorities provide per-ciphertext group elements that reconstruct the remaining mask via threshold combine. The security argument relies only on standard assumptions about CP-ABE, PRFs/VRFs, hash-based authenticated structures, and zero-knowledge proof systems such as STARKs [10]; an explicit Immediate-Revocation-for-Subsequent-Issuances (IRSI) game captures the operational guarantee, including a small cache window Δ that is controlled by policy.

Contributions We introduce no new ABE or ZK primitives. Our contribution is a deployable composition that:

1. **IRSI semantics:** We define and realize *Immediate-Revocation-for-Subsequent-Issuances* for systems with multi-authority control over helper issuance in CP-ABE *without* re-encrypting ciphertexts: after a new root finalizes, authorities refuse *new* helper issuance tied to that root, so later attempts fail.
2. **Coherence across authorities:** A *single-use, on-ledger session* sid binds one global ZK transcript τ (and per-authority projections) to a single attempt, preventing mix-and-match of tokens from different branches; sid also appears *inside the algebra of releases* so cross-session mixing fails.
3. **Private negation at use-time:** Negations are enforced by ZK (non-)membership proofs against public *Sparse Merkle Tree* roots; positives remain in standard CP-ABE (BSW07). Ciphertexts stay unchanged.
4. **Audit with minimal metadata:** A standard VRF derives a key used only to AEAD-encrypt an authority's contribution; an on-chain ZK proof attests *VRF verification on committed inputs* (no inputs revealed).

We explicitly *do not* claim new primitives, ciphertext clawback, or cryptographic non-transferability.

2 Background, Motivation, and Problem Statement

The systems that hold attributes in large organizations are heterogeneous and independently administered. Human resources systems assert roles; project registries record membership; security and legal entities maintain blocklists and legal holds. Access to sensitive data often depends on a combination of positive and negative conditions that cross these boundaries. It is common to require that a reader's project membership is current, that the reader is not on a sanctions or insider-risk list, and that the device posture indicates no active compromise. Positive conditions map naturally to CP-ABE. Negative conditions are more delicate because saying that an attribute does *not* hold often reveals too much about surrounding attributes when encoded directly into a ciphertext policy, especially if the policy structure is exposed. Worse, cross-organizational negatives require one administrator to "speak about" another administrator's subjects.

Revocation raises a different but equally practical concern. A user's attribute can be removed for many reasons—a contract ends, a laptop is compromised, an investigation starts—and each reason calls for an immediate effect from that point forward. Administrators almost never have the option to re-encrypt historical ciphertexts on short notice, and epoch-style key updates give at best a coarse guarantee that still leaves a window in which issuance can succeed. What operators want is a guarantee that any attempt to obtain helper material after a particular revocation event must fail, even if the ciphertext was produced long before.

Finally, organizations must account for access decisions without revealing cryptographic secrets or sensitive metadata. Ledger anchoring is helpful because it provides a global notion of time and finality, and it makes tampering with histories difficult. However, logging the content of attribute assertions or the precise inputs to authorization decisions reveals information that ABE was chosen to conceal.

These pressures motivate ZK-DABE's division of labor. Positives stay in CP-ABE, and encryption remains unchanged. Negatives and revocations are enforced through zero-knowledge proofs against succinct, publicly committed state at each authority. Tokens become the focal object for enforcement: a token is a short value derived under an authority key that is bound to the current root, to the intended ciphertext header, to the global transcript τ, to a *ledger session* sid, and to an identity-tagging value; its sole purpose is to encrypt and authenticate the authority's per-ciphertext contribution to the ABE decryption equation. Because tokens must be obtained afresh on each attempt and because authorities can refuse issuance after a revocation is finalized, obtaining new helper material fails for subsequent attempts even for old ciphertexts.

3 Background and Related Work

ABE, particularly CP-ABE, places the policy in the ciphertext and distributes attribute-key material to users based on attributes [2]. MA-ABE spreads issuance authority across multiple organizations so that no single party can unilaterally create all keys [3], [4]. Non-monotonic ABE is powerful in theory [5], but embedding negation in ciphertexts complicates implementations, increases ciphertext size, and in practice can reveal the shape of conditions that organizations prefer to keep private. Revocation in the ABE literature has several forms—time-based expiration, key updates, and proxy-assisted revocation—but these tend to be either coarse or operationally heavy.

Anchoring state to a ledger with practical byzantine fault tolerance [7] and succinct commitments is an appealing alternative. Sparse Merkle Trees [8], [9] provide logarithmic proofs and a simple non-membership story when the universe is fixed and default values are well-defined. Dynamic accumulators [13], [14] bring constant-size witnesses with different update trade-offs. At the proof layer, transparent systems such as STARKs [10] let a reader show that a fact holds at a committed root without revealing the witness itself. Ideas from outsourced decryption [15] suggest a way to interpose a lightweight gate on the last step of decryption without giving an authority the plaintext; ZK-DABE borrows this intuition but retools it for revocation, cross-authority policy coherence, and identity tagging. BDABE [6] similarly aims for ledgerized audit and decentralized control, but it does not provide a private, cross-authority non-membership mechanism or immediate per-ciphertext issuance semantics.

4 System Model and Assumptions

The system involves a root authority for bootstrapping (optional), a fixed set of attribute authorities (AAs), encryptors, decryptors, and a ledger $\mathcal{L}$ providing finalized blocks. Each AA posts its current domain root $\mathsf{R}_{\mathsf{AA\_id}}$ and signs commitments to attribute lifecycle events, token issuances, and revocations. The per-authority state is represented either as a sparse Merkle tree or as a dynamic accumulator. In the SMT case, keys are $\mathsf{H}(\mathsf{pid}_{u,\mathsf{AA\_id}} \parallel \mathsf{A\_id})$ and values are single bits; default zero leaves are interpreted as absence and yield succinct non-membership proofs. In the accumulator case, the design targets constructions that support both deletions and universal non-membership with efficient witness updates. The adversary may corrupt readers and some authorities and observe ledger metadata; it cannot forge signatures of honest authorities, break collision resistance of the hash/commitment scheme, or defeat the soundness and zero-knowledge properties of the proof system. The ledger provides finality, so a finalized block is never rolled back. The goal is to support policies with conjunctions, disjunctions, thresholds, and negations across multiple authorities, to achieve immediate refusal of *issuance* after revocation without re-encryption, and to preserve the privacy of unrelated attributes while providing auditability.

Although we motivate from MA-ABE, our concrete instantiation uses a single-authority CP-ABE backend (BSW07) together with multi-authority control over attribute state and helper issuance. The logical effect is that no single AA can unilaterally satisfy all conditions required to reconstruct the missing share of α; generalizing the construction to a "true" MA-ABE key structure such as [3], [4] is natural future work.

Control-plane on chain. We assume the control-plane is on the ledger: AA roots, revocation events, and short *pre-commit sessions* (Section 7.5) are written on-chain as commitments. The data plane (ciphertexts, U_i, and ZK witnesses) remains off-chain; authorities verify ZK proofs locally and only commit compact audit entries on chain.

Witness acquisition. Clients obtain Merkle/accumulator membership and non-membership witnesses from one or more authenticated witness services that mirror the latest finalized root; freshness is enforced by binding the root digest into the local proof and by authorities checking equality to the on-chain root at issuance time.

5 Design Overview

The central idea is to leave positive requirements in CP-ABE and to push all negative requirements and revocation checks into zero-knowledge proofs against public state roots, with a lightweight cryptographic gate at decryption time. A decryptor first creates a global proof that there exists a witness assignment under which the policy evaluates to true at the currently finalized roots for all authorities. This proof, denoted π_{glob}, is constructed so that it commits to the subset of leaves relevant to each authority in the form of a projection commitment ϕ_i, and so that it binds to the ciphertext through its identifier and header. The transcript identifier τ is defined as a collision-resistant hash of π_{glob} concatenated with all ϕ_i. *Before any authority issues a token, the decryptor writes a short pre-commit to the ledger that fixes* $(\mathsf{cid}, \mathsf{H}(C_0), \{\phi_i\}, \tau, \mathsf{EncGT}(B_u), \exp)$ *and receives a session id* sid*; all authorities must bind their issuance to this* (sid, τ)*, ensuring cross-authority coherence for OR/threshold policies.*

Session-bound, identity-bound, threshold-friendly combine. Each decryptor has a long-term secret sk_u. Let $b_u = \mathsf{HashToScalar}(sk_u)$ and define a ciphertext-bound user value $B_u = e(C_0, g_2)^{\zeta\, b_u} \in \mathbb{G}_T$ for public constant ζ. For a given session sid, define the *session-bound* tag

$$B_{u,\mathsf{sid}} \;:=\; B_u^{\mathsf{HashToScalar}(\mathsf{sid})} \in \mathbb{G}_T.$$

Authorities incorporate a canceling factor of $B_{u,\mathsf{sid}}^{-1}$ into their releases. This prevents unsolicited replay by third parties that do not possess B_u and, crucially, *prevents cross-session mixing*: shares issued under different sid embed different exponents and cannot be combined. We adopt a threshold-friendly split of the master α: users hold a static α_0, while authorities hold Shamir shares of α_1 so that any quorum Q of size t reconstructs α_1 via Lagrange interpolation. Specifically, fix distinct public indices $x_i \in \mathbb{Z}_p$ for $i \in S$ (e.g., $x_i = i$); sample a random degree-$(t-1)$ polynomial f with $f(0) = \alpha_1$; authority i holds $\alpha_{1,i} = f(x_i)$. The decryptor, who holds only the α_0 share in its ABE key, can compute $Z_0 = e(g_1, g_2)^{\alpha_0 s}$ but cannot recover M without a quorum. Given $Q \subseteq S$ with $|Q| = t$, let $\lambda_{i,Q}(0)$ be the Lagrange coefficient of x_i at point 0 over set Q. Authorities release $U_i = e(C_0, g_2)^{\alpha_{1,i}} \cdot B_{u,\mathsf{sid}}^{-1}$; the decryptor raises U_i to $\lambda_{i,Q}(0)$ and multiplies across $i \in Q$, then multiplies once by $B_{u,\mathsf{sid}}$ to cancel the session-tag, recovering the full mask. Since $\sum_{i \in Q} \lambda_{i,Q}(0) = 1$, the $B_{u,\mathsf{sid}}$ factors cancel exactly once and shares from different sessions do not mix.

The decryptor then approaches each authority with a local projection and a short local proof that the statements relevant to that authority are valid under its root *and* a PoK linking B_u to the same secret used to derive the per-authority pseudonym. If the local check succeeds, the authority derives a token using a PRF or VRF under its secret key; the input to that derivation includes the attribute id, a pseudonym for the reader, the ciphertext id, a hash of the CP-ABE header $C_0 = g_1^s$, the current finalized root, the transcript id τ, *the session id* sid *obtained from the ledger pre-commit*, a short expiry, and an encoding of B_u. This token is not a decryption share by itself. Instead, the token key is used to encrypt and authenticate an authority contribution U_i defined above that is necessary to complete the pairing equation in CP-ABE. Because issuance is denied unless it references the unique on-ledger sid for $(\mathsf{cid}, \mathsf{H}(C_0), \mathsf{EncGT}(B_u))$ and because $B_{u,\mathsf{sid}}$ is embedded in U_i, obtaining new helper material fails for subsequent attempts even for old ciphertexts, and cross-session combination is impossible.

Worked example (two authorities, one negation). Let the policy be $(\mathsf{Role@AA}_1) \wedge \neg(\mathsf{Block@AA}_2)$. Encrypt as in BSW07 to get header $C_0 = g_1^s$. The reader proves in ZK: membership of (Role) under $\mathsf{R}_{\mathrm{AA}_1}$ and *non-membership* of the key for (Block) under $\mathsf{R}_{\mathrm{AA}_2}$, yielding τ and $\{\phi_1, \phi_2\}$. They open a ledger session to get

sid bound to $(\mathsf{cid}, \mathsf{H}(C_0), \{\phi_i\}, \tau, \mathsf{EncGT}(B_u))$. Each authority verifies its local ZK, checks the on-chain pre-commit for (τ, sid), computes $B_{u,\mathsf{sid}} = B_u^{\mathsf{HashToScalar}(\mathsf{sid})}$, and returns an AEAD of $U_i = e(C_0, g_2)^{\alpha_{1,i}} \cdot B_{u,\mathsf{sid}}^{-1}$ keyed by a VRF token on the committed tuple. The reader chooses any quorum of size t, combines $Z_0 = e(g_1, g_2)^{\alpha_0 s}$ with $\prod_{i \in Q} U_i^{\lambda_{i,Q}(0)}$ and then multiplies by $B_{u,\mathsf{sid}}$ to recover $e(g_1, g_2)^{\alpha s}$ and decrypt. If Block is set later and a new $\mathsf{R}_{\mathrm{AA}_2}$ finalizes, future issuance attempts fail under IRSI.

6 Data Structures and Identifiers

An attribute is identified as $\mathsf{A\_id} = \mathsf{H}(PK_A \parallel \mathsf{AA\_id})$, where PK_A is the attribute's public key and $\mathsf{AA\_id}$ is the authority identifier. A user's stable lifecycle pseudonym with respect to an authority is $\mathsf{pid}_{u,\mathsf{AA\_id}} = \mathsf{PRF}_{sk_u}(\mathsf{AA\_id})$; optional per-event pseudonyms can be derived from this stable value when stronger unlinkability is desired, while revocation remains tied to the stable pid. Implementations should derive distinct subkeys from a master user secret for pseudonym generation and binding tags (e.g., sk_u^{pid} and sk_u^{bind} obtained via a KDF with domain separation); for notational simplicity we write sk_u throughout. The SMT key for a subject–attribute pair is $k = \mathsf{H}(\mathsf{pid}_{u,\mathsf{AA\_id}} \parallel \mathsf{A\_id})$ and the value is a single bit. The ciphertext identifier is $\mathsf{cid} = \mathsf{H}(CT \parallel \mathsf{H}(\mathsf{Canon}(G') \parallel \mathsf{salt}))$, which binds the ciphertext to a canonical commitment of the policy and a fresh salt to reduce linkability. A ciphertext-bound user value is $B_u = e(C_0, g_2)^{\zeta\, \mathsf{HashToScalar}(sk_u)}$ as above; the *local* proof includes a PoK that links B_u to the same sk_u used in pid. Because $C_0 = g_1^s$ is fresh per ciphertext, the resulting $B_u = e(g_1, g_2)^{\zeta\, \mathsf{HashToScalar}(sk_u)\, s}$ varies pseudorandomly across ciphertexts; under the discrete-log assumption in $\mathbb{G}_T$ this does not create an obvious cross-ciphertext linkage. *A pre-commit session is identified by* sid *returned by the ledger after recording* $C_\tau = \mathsf{H}(\mathsf{cid} \parallel \mathsf{H}(C_0) \parallel \phi_1 \parallel \cdots \parallel \phi_m \parallel \tau \parallel \mathsf{EncGT}(B_u) \parallel \exp \parallel \mathit{nonce} \parallel \mathit{tag} = \mathit{precommit})$. For threshold combine we fix distinct public indices x_i for $i \in S$ (e.g., enumeration order) and use standard Lagrange coefficients $\lambda_{i,Q}(0)$ at point 0 for quorums Q.

7 Protocols

The setup and onboarding phase publishes global parameters for ABE and ZK and initializes authority roots on chain. At setup time, the ABE master secret is split into $\alpha = \alpha_0 + \alpha_1$, where α_1 is Shamir-shared among a fixed authority set S with threshold t using a random degree-$(t$–$1)$ polynomial f such that $f(0) = \alpha_1$; authority i holds $\alpha_{1,i} = f(x_i)$. $E_0 = e(g_1, g_2)^{\alpha_0}$ is published, and each authority $i \in S$ publishes $Y_i = e(g_1, g_2)^{\alpha_{1,i}}$ so public parameters are stable. (If S or t changes, a new epoch and re-encryption are required.) During attribute creation and assignment, an authority registers attribute keys and sets the SMT bit at the key k to one for an assignment; it posts the recomputed root and signs a commitment to the assignment event. The user's long-lived CP-ABE keys are issued as in BSW07 except that the component tied to α carries only α_0.

7.1 Setup and Onboarding

Algorithm 1 Setup$(\lambda) \to (\mathrm{params}, \{SK_{\mathsf{AA\_id}}\})$

Require: Security parameter λ.

1: Root Authority (optional) publishes global parameters for ABE and ZK on-chain.
2: **ABE split (fixed authority set S, threshold t):** choose $\alpha_0, \alpha_1, \beta \leftarrow \mathbb{Z}_p$; let $\alpha \leftarrow \alpha_0 + \alpha_1 \pmod p$; set generators $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$, $h = g_1^\beta$, and publish standard BSW07 params including $E = e(g_1, g_2)^\alpha$ and $E_0 = e(g_1, g_2)^{\alpha_0}$.
3: Fix distinct public indices x_i for $i \in S$; sample degree-$(t-1)$ polynomial f with $f(0) = \alpha_1$; each authority $i \in S$ receives $\alpha_{1,i} = f(x_i)$ and registers $Y_i = e(g_1, g_2)^{\alpha_{1,i}}$ (and a VRF pk).
4: Each authority initializes empty domain state and posts initial root with finality.
5: **return** $(\mathrm{params}, \{SK_{\mathsf{AA\_id}}\})$.

7.2 Attribute Lifecycle

Algorithm 2 CreateAttribute$(SK_{\mathsf{AA\_id}}, A) \to (PK_A, SK_A)$

Require: Authority secret $SK_{\mathsf{AA\_id}}$, optional plaintext label A.

1: Generate (PK_A, SK_A); set $\mathsf{A\_id} = \mathsf{H}(PK_A \parallel \mathsf{AA\_id})$; optionally encrypt label.
2: Post $(\mathsf{A\_id}, PK_A, \text{enc-label})$ and a signature over a commitment to $(\mathsf{A\_id}, PK_A)$.
3: Store SK_A securely.
4: **return** (PK_A, SK_A).

Algorithm 3 AssignAttribute$(SK_{\mathsf{AA\_id}}, SK_A, \mathsf{pid}_{u,\mathsf{AA\_id}}, PK_u)$

1: Set SMT bit to 1 at $k = \mathsf{H}(\mathsf{pid}_{u,\mathsf{AA\_id}} \parallel \mathsf{A\_id})$; recompute root; post new $\mathsf{R}_{\mathsf{AA\_id}}$ with finality.
2: Derive a long-lived partial key for (A, u) and encrypt to PK_u (CP-ABE dependent; users receive keys with α_0 share only).
3: Log a signed commitment to $(\mathsf{A\_id}, \mathsf{Com}(\mathsf{pid}_{u,\mathsf{AA\_id}}), \mathsf{R}_{\mathsf{AA\_id}}, \mathsf{tag} = \mathsf{assign})$.

7.3 Encryption

Algorithm 4 EncryptData$(M, G') \to (CT, \mathsf{cid})$

Require: Plaintext M, acyclic policy DAG G' with positives/negations.

1: Verify acyclicity; compute canonical form and salted commitment (Section 12). *(Acyclicity via DFS/SCC [16].)*
2: Collect PK_A for positive leaves; run standard CP-ABE (BSW07) to produce CT with header $C_0 = g_1^s$, $C = h^s$, and $\tilde{C} = M \cdot E^s = M \cdot e(g_1, g_2)^{\alpha s}$.
3: Set $\mathsf{cid} = \mathsf{H}(CT \parallel \mathsf{H}(\mathsf{Canon}(G') \parallel \mathsf{salt}))$; post commitments; store CT off-chain.
4: **return** (CT, cid).

7.4 Global Satisfiability with Projection Commitments

Algorithm 5 ProveSatisfy$(\mathsf{cid}, G', \vec{\mathsf{R}}) \to (\pi_{\text{glob}}, \{\phi_i\}, \tau)$

Require: Ciphertext id cid, policy G', current finalized roots $\vec{\mathsf{R}}$.

1: Construct a ZK proof π_{glob} that there exists a witness assigning truth to leaves such that G' evaluates to true under $\vec{\mathsf{R}}$, using membership/non-membership statements for only the leaves in the witness, all bound to (cid, PK_u).
2: For each authority i, compute a projection commitment $\phi_i = \mathsf{H}(\mathsf{AA\_leaves}_i \parallel \mathsf{truth\text{-}bits}_i \parallel \mathsf{R}_i)$ inside the circuit; output $\{\phi_i\}$.
3: Let $\tau \leftarrow \mathsf{H}(\pi_{\text{glob}} \parallel \phi_1 \parallel \cdots \parallel \phi_m)$.
4: **return** $(\pi_{\text{glob}}, \{\phi_i\}, \tau)$.

7.5 On-ledger Pre-Commit / Session Open

Algorithm 6 OpenSession$(\mathsf{cid}, C_0, \mathsf{EncGT}(B_u), \{\phi_i\}, \pi_{\text{glob}}, \tau, \text{exp}) \to (\mathsf{sid}, C_\tau)$

Require: Ciphertext id cid, header $C_0 = g_1^s$, encoding $\mathsf{EncGT}(B_u)$, projection commitments $\{\phi_i\}$, global proof π_{glob}, transcript id τ, expiry exp.

1: Compute a pre-commitment $C_\tau \leftarrow \mathsf{H}(\mathsf{cid} \parallel \mathsf{H}(C_0) \parallel \phi_1 \parallel \cdots \parallel \phi_m \parallel \tau \parallel \mathsf{EncGT}(B_u) \parallel \text{exp} \parallel \mathsf{nonce} \parallel \mathsf{tag} = \mathsf{precommit})$.
2: Submit C_τ to $\mathcal{L}$ and receive a unique session id sid; the ledger enforces that, for a given $(\mathsf{cid}, \mathsf{H}(C_0), \mathsf{EncGT}(B_u))$, only the latest unexpired sid is considered active for issuance.
3: **return** (sid, C_τ).

7.6 Per-Authority Token Issuance (Header-, Identity-, and Session-Bound)

Algorithm 7 IssueToken$(SK_{\mathsf{AA\_id}}, \mathsf{pid}_{u,\mathsf{AA\_id}}, PK_u, \mathsf{cid}, C_0, B_u, G'_{\mathsf{AA\_id}}, \pi_{\text{loc}}, (\pi_{\text{glob}}, \phi_i, \tau), (\mathsf{sid}, C_\tau), \text{exp}) \to (Tok_i, \pi_{\text{vrf},i}, \mathsf{Enc}_{Tok_i}[U_i])$

Require: Authority secret $SK_{\mathsf{AA\_id}}$, reader pseudonym and public key, ciphertext id, the CP-ABE header $C_0 = g_1^s$, user binding value B_u, the authority-local slice $G'_{\mathsf{AA\_id}}$, a local proof π_{loc} (membership/non-membership under $\mathsf{R}_{\mathsf{AA\_id}}$ and a PoK linking B_u to the same secret used in $\mathsf{pid}_{u,\mathsf{AA\_id}}$) and the global proof π_{glob} with projection commitment ϕ_i and transcript id τ, current finalized root $\mathsf{R}_{\mathsf{AA\_id}}$ and expiry exp (epoch or height), and a session (sid, C_τ) fetched from the ledger.

1: Verify π_{loc} against $G'_{\mathsf{AA\_id}}$ and $\mathsf{R}_{\mathsf{AA\_id}}$ and that it binds $(\mathsf{pid}_{u,\mathsf{AA\_id}}, B_u, PK_u, \mathsf{cid}, \mathsf{H}(C_0))$.
2: Check *projection and session consistency*: (i) ϕ_i opens (via provided opening data inside π_{loc}) to the same leaf indices and truth-bits the authority validated; (ii) recompute τ and verify that C_τ on-chain matches $(\mathsf{cid}, \mathsf{H}(C_0), \{\phi_j\}, \tau, \mathsf{EncGT}(B_u), \text{exp})$; (iii) ensure the ledger marks sid as the latest unexpired session for $(\mathsf{cid}, \mathsf{H}(C_0), \mathsf{EncGT}(B_u))$.
3: **if** any check fails **then**
4: **return** `NULL`.
5: **end if**
6: Compute $(Tok_i, \pi_{\text{vrf},i}) \leftarrow \mathsf{VRF.Eval}_{sk^{\text{vrf}}_{\mathsf{AA\_id}}}(\mathsf{A\_id} \parallel \mathsf{pid}_{u,\mathsf{AA\_id}} \parallel \mathsf{cid} \parallel \mathsf{H}(C_0) \parallel \mathsf{R}_{\mathsf{AA\_id}} \parallel \tau \parallel \mathsf{sid} \parallel \text{exp} \parallel \mathsf{EncGT}(B_u))$.
7: Compute the session-bound tag $B_{u,\mathsf{sid}} \leftarrow B_u^{\mathsf{HashToScalar(sid)}}$ in $\mathbb{G}_T$.
8: Compute $U_i \leftarrow e(C_0, g_2^{\alpha_{1,i}}) \cdot B_{u,\mathsf{sid}}^{-1} \in \mathbb{G}_T$; symmetrically encrypt its encoding under $K_i = \mathsf{KDF}(Tok_i)$ to get $\mathsf{Enc}_{Tok_i}[U_i]$ (AAD: $(\mathsf{cid}, \mathsf{H}(C_0), \mathsf{R}_{\mathsf{AA\_id}}, \tau, \mathsf{sid}, \text{exp}, \mathsf{EncGT}(B_u), \mathsf{EncGT}(B_{u,\mathsf{sid}}))$).
9: Create commitment $C = \mathsf{H}(\mathsf{A\_id} \parallel \mathsf{Com}(\mathsf{pid}_{u,\mathsf{AA\_id}}) \parallel \mathsf{R}_{\mathsf{AA\_id}} \parallel \mathsf{cid} \parallel \tau \parallel \mathsf{sid} \parallel \text{exp} \parallel \mathsf{H}(C_0) \parallel \mathsf{EncGT}(B_u) \parallel \mathsf{tag} = \mathsf{token})$; sign with $SK_{\mathsf{AA\_id}}$.
10: Produce a *zk-VRF commit-and-prove* (VRF-inside-ZK) proof π_{zkvrf} attesting, in zero knowledge, that there exists an input tuple equal to the one committed in C for which the VRF output verifies under the authority's public VRF key. Record $(C, \text{sig}, \pi_{\mathsf{zkvrf}})$ on-chain. Return $(Tok_i, \pi_{\text{vrf},i}, \mathsf{Enc}_{Tok_i}[U_i])$ encrypted to PK_u.

7.7 Revocation

Algorithm 8 RevokeAttribute$(SK_{\mathsf{AA\_id}}, \mathsf{pid}_{u,\mathsf{AA\_id}}, \mathsf{A\_id})$

Require: Authority secret, reader pseudonym, attribute id.

1: Set SMT bit to 0 at $k = \mathsf{H}(\mathsf{pid}_{u,\mathsf{AA\_id}} \parallel \mathsf{A\_id})$; recompute and post new finalized $\mathsf{R}_{\mathsf{AA\_id}}$; sign a commitment with tag revoke.
2: From the finalizing block forward, treat this new $\mathsf{R}_{\mathsf{AA\_id}}$ as the only valid root for issuance and refuse to issue tokens for $(\mathsf{pid}_{u,\mathsf{AA\_id}}, \mathsf{A\_id})$ whenever the local ZK statement would have to assert membership of this pair at $\mathsf{R}_{\mathsf{AA\_id}}$; this makes post-revocation issuance fail (up to the explicit cache window Δ defined in Section 14).

7.8 Decryption (Header-, Identity-, and Session-Bound Threshold Combine)

Algorithm 9 DecryptData$(CT, \mathsf{cid}, G') \to M$ or $\perp$

Require: Ciphertext CT with header $C_0 = g_1^s$, id cid, policy G'.

1: Compute $B_u = e(C_0, g_2)^{\zeta\, \mathsf{HashToScalar}(sk_u)}$.
2: Obtain $(\pi_{\text{glob}}, \{\phi_i\}, \tau)$ and verify against finalized roots and policy commitment.
3: Open a session (sid, C_τ) via Algorithm 6.
4: Compute the session-bound tag $B_{u,\mathsf{sid}} = B_u^{\mathsf{HashToScalar}(\mathsf{sid})}$.
5: Ensure long-lived partial keys for positives are present (carrying α_0 share).
6: For each authority $i \in S$, present the local projection, B_u, and receive $(Tok_i, \pi_{\text{vrf},i}, \mathsf{Enc}_{Tok_i}[U_i])$; verify the *standard* VRF proof $\pi_{\text{vrf},i}$ locally on the known input tuple and decrypt U_i with $K_i = \mathsf{KDF}(Tok_i)$, checking that AAD contains the same $(\mathsf{cid}, \mathsf{H}(C_0), \mathsf{R}_{\mathsf{AA\_id}}, \tau, \mathsf{sid}, \text{exp}, \mathsf{EncGT}(B_u), \mathsf{EncGT}(B_{u,\mathsf{sid}}))$.
7: **if** any required material is missing/invalid or τ or sid mismatches or exp is stale **then return** $\perp$.
8: **end if**
9: Choose a quorum $Q \subseteq S$ with $|Q| = t$; compute Lagrange coefficients $\lambda_{i,Q}(0)$ for $i \in Q$.
10: Run BSW07 recursion to obtain $Z_0 = e(g_1, g_2)^{\alpha_0 s}$; set $Z \leftarrow Z_0 \cdot \left(\prod_{i \in Q} U_i^{\lambda_{i,Q}(0)}\right) \cdot B_{u,\mathsf{sid}}$; recover M from $\tilde{C} = M \cdot Z$.
11: **return** M.

8 Concrete Instantiation on BSW07, Session- and Identity-Bound Threshold Gating

Let $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ be bilinear groups of prime order p with generators $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$, set $h = g_1^\beta$, and let $E = e(g_1, g_2)^\alpha$ be the public BSW07 parameter published at setup.

The BSW07 instantiation is minimal and leaves encryption unchanged. The master exponent α is split so that users receive keys reflecting only α_0 while the authorities collectively hold a Shamir sharing of α_1: authority $i \in S$ holds $\alpha_{1,i} = f(x_i)$ for a public index x_i and a degree-$(t-1)$ polynomial f with $f(0) = \alpha_1$. Standard BSW07 encryption publishes $C_0 = g_1^s$ and $\tilde{C} = M \cdot E^s = M \cdot e(g_1, g_2)^{\alpha s}$. A decryptor with a valid positive-attribute key can compute $Z_0 = e(g_1, g_2)^{\alpha_0 s}$, which is insufficient to recover M. Authority i computes

$$U_i \;=\; e(C_0, g_2)^{\alpha_{1,i}} \cdot B_{u,\mathsf{sid}}^{-1} \;=\; e(g_1, g_2)^{\alpha_{1,i} s} \cdot B_{u,\mathsf{sid}}^{-1},$$

but releases it only if the local and global zero-knowledge checks pass; the value is delivered under an AEAD keyed by a VRF-derived token that is itself bound to $(\mathsf{A\_id}, \mathsf{pid}, \mathsf{cid}, \mathsf{H}(C_0), \mathsf{R}_i, \tau, \mathsf{sid}, \text{exp}, \mathsf{EncGT}(B_u), \mathsf{EncGT}(B_{u,\mathsf{sid}}))$. Combining Z_0 with any quorum Q of such U_i and Lagrange exponents, then multiplying by $B_{u,\mathsf{sid}}$, yields the correct mask

$$Z \;=\; Z_0 \cdot \Big(\prod_{i \in Q} U_i^{\lambda_{i,Q}(0)}\Big) \cdot B_{u,\mathsf{sid}} \;=\; e(g_1, g_2)^{(\alpha_0 + \sum_{i \in Q} \lambda_{i,Q}(0)\alpha_{1,i}) s} \cdot B_{u,\mathsf{sid}}^{\,1 - \sum_{i \in Q} \lambda_{i,Q}(0)} \;=\; e(g_1, g_2)^{\alpha s},$$

since $\sum_{i \in Q} \lambda_{i,Q}(0) = 1$. Because each U_i is computed from the actual header, tagged to the user via $B_{u,\mathsf{sid}}$, and authorities will not issue tokens for revoked pairs once a new root finalizes, obtaining new helper material after revocation fails (IRSI). *The session* sid *is cryptographically embedded in the releases, so cross-session mix-and-match fails.*

9 Projection Consistency and Global τ

The global proof π_{glob} returns, for each authority, a commitment to the exact subset of leaves and their truth values that were used in the satisfying witness, together with the root at which those statements were evaluated. Concretely, the π_{glob} circuit takes as private input the full policy witness (all leaf values and their Merkle/accumulator proofs), evaluates the Boolean DAG G' under the vector of roots $\vec{\mathsf{R}}$, and for each

authority i computes $\phi_i = \mathsf{H}(\mathsf{AA\_leaves}_i, \mathsf{truth\text{-}bits}_i, \mathsf{R}_i)$; it outputs π_{glob} and the list $\{\phi_i\}$. The transcript id is $\tau = \mathsf{H}(\pi_{\text{glob}}, \phi_1, \ldots, \phi_m)$.

For each authority i, the local proof π_{loc} takes as private input exactly the slice of the witness corresponding to $G'_{\mathsf{AA\_id}}$, proves the Merkle/accumulator verification equations at the posted $\mathsf{R}_{\mathsf{AA\_id}}$, proves a PoK link between $\mathsf{pid}_{u,\mathsf{AA\_id}}$ and B_u, and enforces inside the circuit that recomputing $H(\mathsf{AA\_leaves}_i, \mathsf{truth\text{-}bits}_i, \mathsf{R}_i)$ yields the public ϕ_i. The authority requests an opening to its ϕ_i when checking the local proof and recomputes τ from $(\pi_{\text{glob}}, \phi_1, \ldots, \phi_m)$.

Before issuance, the decryptor records a pre-commit C_τ and receives sid *from the ledger (Section 7.5); each authority verifies that the C_τ it reads on chain matches the* $(\mathsf{cid}, \mathsf{H}(C_0), \{\phi_i\}, \tau, \mathsf{EncGT}(B_u))$ *it sees locally and that* sid *is the latest unexpired session for that tuple.* Any attempt to assemble tokens from different policy branches would require conflicting $\{\phi_i\}$ and hence a different τ, which would not match the single pre-committed C_τ; authorities *refuse* issuance for sessions that do not match the on-chain pre-commit. Together with the session-bound algebraic tag $B_{u,\mathsf{sid}}$, this enforces cross-authority and cross-session coherence; mixing tokens across witnesses would require either breaking collision resistance of H or the soundness of π_{glob} or π_{loc}.

10 SMT / Accumulator Details and ZK Constraints

Default choice. We use *Sparse Merkle Trees* by default. Accumulators are optional and only when a *dynamic, universal* non-membership construction with deletions is available; otherwise we remain with SMTs.

The SMT instantiation uses a binary tree over a fixed 256-bit space. Leaves are domain-separated as $L(k, v) = \mathsf{H}(\texttt{0x01} \parallel k \parallel v)$ with $v \in \{0, 1\}$, and internal nodes are $I(l, r) = \mathsf{H}(\texttt{0x00} \parallel l \parallel r)$. Default leaves represent the value $v = 0$, and default internal nodes are computed bottom-up from the default leaves. A membership proof consists of the sibling hashes along the path and the claimed leaf; the zero-knowledge constraint system recomputes the root from $L(k, 1)$ using the index bits of k to choose left or right at each level. A non-membership proof shows the path to the default leaf at index k and convinces the verifier that the chain of default nodes is consistent; the circuit checks that the path yields the posted root and that no conflicting occupied sibling is claimed, which is sufficient for sparse trees with fixed defaults. When accumulators are used, the construction restricts to dynamic, universal schemes that support deletions and provide efficient universal non-membership witnesses; the zero-knowledge interface exposes the group and pairing checks needed to verify these witnesses.

11 Auditability with Privacy

To enable public accountability without revealing the VRF input tuple, we post a zero-knowledge proof that a *standard VRF verification equation holds on a committed input.* This is a conventional ZK statement ("VRF verification inside ZK"), not a new primitive. Issuance events are logged as signed commitments $C = \mathsf{H}(\mathsf{A\_id}, \mathsf{Com}(\mathsf{pid}), \mathsf{R}, \mathsf{cid}, \tau, \mathsf{sid}, \exp, \mathsf{H}(C_0), \mathsf{EncGT}(B_u), \mathsf{tag} = \mathsf{token})$. The on-chain verifier checks π_{zkvrf} against C and the public VRF key but learns neither the input tuple nor the VRF output. Off-chain, the decryptor still receives and verifies the *standard* VRF proof π_{vrf} for its known input tuple before accepting Tok_i and decrypting U_i. If public verifiability is not required, the system can replace the VRF with a PRF or OPRF; in the OPRF case the authority does not learn the input tuple at issuance time, which improves privacy at the cost of public audit.

12 Policy Canonicalization

To avoid ambiguity in policy commitments and to reduce linkability, the policy DAG is canonicalized before hashing. The process starts by topologically sorting the DAG using depth and a stable node identifier to obtain a deterministic ordering. Each node is encoded as a tuple containing the node type (one of AND, OR, threshold, positive leaf, or negative leaf), a label, the ordered list of child identifiers if any, and the threshold parameter for k-of-n gates. Leaves are hashed with domain separation as $\mathsf{H}(\texttt{leaf} \parallel \mathsf{AA\_id} \parallel \mathsf{A\_id})$,

while internal nodes are hashed as $\mathsf{H}(\texttt{node} \parallel \mathsf{type} \parallel \mathsf{child\_hashes} \parallel \mathsf{k})$. The policy commitment is the root hash of this canonical representation combined with a fresh salt.

13 Security Properties and Why They Hold

Correctness follows from the BSW07 equations once the missing contributions are supplied. A decryptor computes $Z_0 = e(g_1, g_2)^{\alpha_0 s}$ using its long-lived key and then multiplies in $\prod_{i \in Q} U_i^{\lambda_{i,Q}(0)}$ for any quorum Q of size t and finally $B_{u,\mathsf{sid}}$; because $\sum_{i \in Q} \lambda_{i,Q}(0) = 1$, the product is $e(g_1, g_2)^{\alpha s}$, which inverts the mask on $\tilde{C}$. Confidentiality reduces to the IND-CPA security of CP-ABE and the security of the PRF/VRF. Replacing the tokens with random values makes the AEAD under which U_i is delivered unlinkable and semantically secure; without valid tokens and the corresponding U_i, the adversary learns nothing about Z. The privacy of attribute claims rests on the zero-knowledge properties of the global and local proofs, which reveal only acceptance and not the unused parts of the witness. **Immediate revocation for subsequent issuances (IRSI)** is captured by the definition in Section 14. After a new root finalizes, the local non-membership statement for a revoked pair becomes false; by soundness, it cannot be proven, and by PRF/VRF security, tokens bound to the new root cannot be forged. Previously released helper material for a ciphertext may still be usable; the system guarantees refusal of *new issuance* after revocation. *Session coherence:* because every issuance must reference the unique on-ledger sid tied to the pre-committed $(\mathsf{cid}, \mathsf{H}(C_0), \{\phi_i\}, \tau, \mathsf{EncGT}(B_u))$ and because sid is embedded in the algebra via $B_{u,\mathsf{sid}}$, mixing tokens across different global witnesses or sessions is rejected by honest authorities and fails algebraically at combine time.

14 Immediate-Revocation-for-Subsequent-Issuances (IRSI)

IRSI formalizes the operational guarantee that, after a revocation event finalizes, attempts to obtain *new* helper material for a revoked subject–attribute pair should fail, even for ciphertexts created before the revocation. We give a brief game sketch.

Consider the following experiment $\mathsf{Exp}^{\mathsf{IRSI}}_{\mathcal{A}}(\lambda)$.

1. The challenger runs $\mathsf{Setup}(\lambda)$, obtaining public parameters and secret authority keys, and gives the public parameters to the adversary $\mathcal{A}$.
2. Before revocation, $\mathcal{A}$ may adaptively:
   - request attribute assignments and revocations (subject to well-formedness),
   - obtain user keys for chosen identities and attribute sets,
   - issue *token queries* for any tuple $(u, \mathsf{A\_id}, \mathsf{cid}, \mathsf{H}(C_0), \mathsf{EncGT}(B_u))$ consistent with the current roots, receiving $(Tok_i, \pi_{\mathrm{vrf},i}, \mathsf{Enc}_{Tok_i}[U_i])$ from honest authorities as specified by the protocol.
3. At some time of its choice, encoded as a ledger height $h^\star$, $\mathcal{A}$ designates a target subject–attribute pair $(u^\star, \mathsf{A\_id}^\star)$ and requests that the challenger post a revocation event for this pair. The challenger updates the corresponding SMT/accumulator, posts the new root with finality at height $h^\star$, and from then on treats this root as current.
4. The challenger now generates a challenge ciphertext $CT^\star$ under a policy that can be satisfied using $(u^\star, \mathsf{A\_id}^\star)$ and some set of other attributes, with encryption time strictly after $h^\star$, and returns $CT^\star$ to $\mathcal{A}$.
5. After the challenge, $\mathcal{A}$ continues to interact with a *post-revocation* issuance oracle that:
   - enforces root freshness (all local proofs must be evaluated at the latest finalized root),
   - enforces the session rules (only the latest unexpired sid per $(\mathsf{cid}, \mathsf{H}(C_0), \mathsf{EncGT}(B_u))$ is honored),
   - refuses to issue new tokens (and thus U_i) for the revoked pair $(u^\star, \mathsf{A\_id}^\star)$ at or after height $h^\star$, except that it may re-issue previously produced helper material within a fixed cache window Δ (modeling clock skew and implementation caches).

6. Eventually $\mathcal{A}$ outputs a bit b' and a candidate plaintext $M^\star$ for $CT^\star$.

We say $\mathcal{A}$ wins IRSI if it can decrypt $CT^\star$ *after* revocation by using helper material that would be refused by the post-revocation oracle (for example, by forging new tokens at the new root), or by using helper material that was legitimately issued before $h^\star$ but only applied after the cache window Δ has expired. The IRSI advantage of $\mathcal{A}$ is its success probability in this game.

Any winning strategy would either violate zero-knowledge soundness by proving a false local statement at the new root, or forge a PRF/VRF output on a fresh input tuple, or recover the ABE plaintext without the correct mask, which contradicts CP-ABE's IND-CPA security. *Additionally, the oracle enforces that post-challenge issuance must reference the unique pre-committed* sid *for* $(\mathsf{cid}, \mathsf{H}(C_0), \mathsf{EncGT}(B_u))$ *and the algebraic combine uses* $B_{u,\mathsf{sid}}$*, preventing cross-branch or cross-session token mixing.*

15 Complexity and Performance

Sparse Merkle proofs cost $O(\log N)$ hashes to create and verify, and their zero-knowledge verification amounts to recomputing the root along a constrained path with domain-separated compression at each level. The dominant cost in the global proof is the size of the projection and the Boolean and arithmetic constraints used to evaluate the policy DAG and to check the (non-)membership witnesses. Transparent proof systems such as STARKs [10] remove the need for a trusted setup and perform well at the scales typical for access control. On-chain data consists of roots, short commitments, and compact zk-VRF proofs (or PRF/OPRF commitments); ciphertexts, long-lived keys, and authority contributions remain off chain. Finality and expiry are expressed either in heights or in epoch timestamps, and verifiers require that tokens refer to the latest finalized root and that the current height or time is within the allowed window.

16 Implementation Guide

Implementers should treat identifiers and commitments as typed values with explicit domain separation to prevent cross-protocol collisions. Pseudonyms should be stable at the lifecycle layer to make revocation unambiguous; per-issuance pseudonyms can be derived where unlinkability is a priority. Commitments to events should include nonces and explicit tags and should be paired with signatures under long-lived authority keys. Tokens should be used only as AEAD keys to deliver authority shares; shares then combine multiplicatively in G_T with Lagrange exponents before the standard BSW07 unmasking step. Finality should be measured against a local view of the ledger with a simple rule: accept issuance only when the referenced root is the latest finalized one and the expiry is within a small safety margin. *When placing the control-plane on chain, keep only succinct commitments and the pre-commit sessions (Section 7.5) on-chain; verify ZK proofs off-chain; never put* U_i *or witnesses on-chain.* For high-value negative conditions, it is prudent to replicate authorities and require a threshold of approvals or denials so that a single compromised authority cannot subvert enforcement. Authorities should query a finalized view of the ledger and ignore sessions that are not yet final or that have been superseded as "not latest" for a given $(\mathsf{cid}, \mathsf{H}(C_0), \mathsf{EncGT}(B_u))$.

17 Limitations, Risks, and Ethics

Scope and non-claims. I present a systems composition of known primitives. I *do not* introduce new ABE/ZK primitives, nor prove fine-grained traceability or non-transferability beyond preventing unsolicited replay via identity tagging. My revocation guarantee (IRSI) affects *future issuance attempts*; previously obtained helper material for the same ciphertext may remain usable.

Once recovered, plaintext cannot be clawed back; token gating affects future *issuance* only. Liveness depends on authorities being available to issue tokens upon request; pre-issuance with short expiry mitigates but does not remove this dependency. If an authority is corrupt, the system provides accountability through zk-VRFs and commitments but cannot force a malicious authority to refuse issuance; thresholds and replication are the appropriate countermeasure. Ledger timing and write volumes leak limited metadata about issuance activity; pseudonyms and commitments reduce this exposure but do not eliminate it. The zero-knowledge layer can be post-quantum if implemented with STARKs; the CP-ABE backend remains pairing-based unless

replaced with a lattice-based alternative. **Identity tagging is not a cryptographic non-transferability mechanism:** a cooperating recipient can share B_u and received shares U_i; ZK-DABE relies on ledgered issuance and audit to deter and attribute such behavior.

18 Related Work

Foundational work on ABE [1], [2], MA-ABE [3], [4], and non-monotonic ABE [5] established the expressive and administrative baseline. Ledger anchoring [8], [17] and accumulators [13], [14], [18], [19] contribute succinct authenticity, while VRFs [20] and modern zero-knowledge systems [10] enable verifiable, private gatekeeping at issuance time. ZK-DABE composes these elements and introduces the global transcript and header-, session-, and identity-bound authority shares with threshold combine to realize private negations and immediate revocation for subsequent *issuances*.

19 Conclusion

ZK-DABE connects the expressiveness of MA-ABE with the operational demands of zero-trust deployments. Private non-membership proofs at public roots enforce negatives without revealing unrelated attributes. Header-, session-, and identity-bound tokens and authority shares make revocation take effect immediately for future *issuances*, and ledger-anchored commitments provide accountability while preserving privacy. The global transcript τ and the projection commitments ensure coherence across authorities and prevent cross-branch token mixing at issuance time. *A minimal on-ledger pre-commit session* sid*, also bound in the algebra of releases, ties the single global transcript to all authorities for a given attempt, closing the mix-and-match gap while keeping encryption unchanged.* The construction is modular in both its ABE and ZK backends and admits practical implementation without requiring changes to encryption. Session-bound identity tagging prevents unsolicited replay and cross-session mixing; audit deters cooperative transfer. A VRF-inside-ZK path enables public accountability without revealing issuance inputs.

Appendix A: Security Game Sketches

Confidentiality. Replacing tokens with random values using PRF/VRF security makes the AEAD under which U_i is delivered semantically secure; the adversary's advantage collapses to CP-ABE IND-CPA.

Revocation soundness / IRSI. After revocation, the local statement becomes false under the new root; forging tokens violates ZK soundness or PRF/VRF security, while using pre-revocation tokens outside Δ contradicts verifier checks. Previously issued helper material may still decrypt the same ciphertext; our guarantee is refusal of *new issuance*.

Coherence and identity tagging. Mixing tokens across witnesses changes some ϕ_i and hence τ; mismatch causes issuance and AEAD checks to fail with overwhelming probability. *The on-ledger pre-commit* sid*, and its algebraic embedding via* $B_{u,\mathsf{sid}}$*, make this enforcement binding across authorities and sessions.* Identity tagging prevents unsolicited replay by parties lacking B_u, but does not prevent cooperative transfer; accountability follows from ledgered issuance commitments and the VRF-inside-ZK audit path.

Copyright and Licensing

References

[1] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *ACM CCS*, 2006.

[2] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *IEEE Symposium on Security and Privacy (S&P)*, 2007.

[3] M. Chase, "Multi-authority attribute-based encryption," in *TCC*, 2007, pp. 515–534.

[4] A. Lewko and B. Waters, "Decentralizing attribute-based encryption," in *EUROCRYPT*, 2011.

[5] R. Ostrovsky, A. Sahai, and B. Waters, "Attribute-based encryption with non-monotonic access structures," in *ACM CCS*, 2006, pp. 195–203.

[6] H. Guo, Z. Zheng, B. Ou, and H. Yang, "Bdabe: Blockchain-based decentralized attribute-based encryption for data sharing in cloud computing," *Peer-to-Peer Networking and Applications*, vol. 14, 2021.

[7] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in *OSDI*, 1999, pp. 173–186.

[8] B. Laurie, A. Langley, and E. Kasper, *Certificate transparency (rfc 6962)*, RFC, 2013.

[9] R. C. Merkle, "A certified digital signature," in *CRYPTO*, 1989.

[10] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev, "Scalable, transparent, and post-quantum secure computational integrity," *IACR ePrint*, no. 2018/046, 2018.

[11] M. Shpantzer, *Assume breach: Red teaming for data center resilience*, SANS Whitepaper, 2015.

[12] S. Rose, O. Borchert, S. Connelly, and M. Souppaya, "Zero trust architecture," NIST, Tech. Rep. SP 800-207, 2020.

[13] J. Camenisch and A. Lysyanskaya, "Dynamic accumulators and application to efficient revocation of anonymous credentials," in *CRYPTO*, 2002.

[14] L. Nguyen, "Accumulators from bilinear pairings and applications," in *CT-RSA*, 2005.

[15] M. Green, S. Hohenberger, and B. Waters, "Outsourcing the decryption of abe ciphertexts," in *USENIX Security*, 2011.

[16] R. E. Tarjan, "Depth-first search and linear graph algorithms," *SIAM Journal on Computing*, vol. 1, no. 2, pp. 146–160, 1972.

[17] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," Ethereum Foundation, Tech. Rep., 2014.

[18] J. Benaloh and M. de Mare, "One-way accumulators: A decentralized alternative to digital signatures," in *EUROCRYPT Workshop*, 1993.

[19] N. Barić and B. Pfitzmann, "Collision-free accumulators and fail-stop signature schemes without trees," in *EUROCRYPT*, 1997.

[20] S. Micali, M. Rabin, and S. Vadhan, "Verifiable random functions," in *FOCS*, 1999.