Bearer Object Protocol

Pre-MVP Technical Specification v0.6

Singular digital authority without a global transaction ledger

Frozen pre-MVP specification - September 2026

Core thesis: local singular authority without a global transaction ledger

BOP v0.6 freezes the pre-MVP architecture around a small bearer core and explicit optional recovery profiles. The principal changes from v0.5 are: an atomic MOVE+RETIRE Transition Bundle; Closure-Recoverable formalized as an append-only recovery-nullifier set with blinded successor commitments; quorum-signed Closure Receipts; bounded closure timing; post-retirement operational safety; and use of Closure Receipts as temporal anchors for catastrophic hardware revocation.

Status: frozen pre-MVP technical specification. Quantum constructions are used only as conceptual motivation for the UAP abstraction. The MVP target is classical UAP-H secure hardware.

Abstract

Digital systems ordinarily prevent double spending by maintaining authoritative shared state. Centralized systems store the current owner or balance; blockchains replicate that state and establish a canonical history through consensus. BOP investigates a narrower primitive: whether one application-defined digital authority can have exactly one legitimate irreversible outcome from each live predecessor without maintaining a global transaction ledger.

BOP v0.6 defines an implementation-independent Unclonable Authority Primitive (UAP). The security game is output-centric: an adversary must not be able to derive two conflicting valid successor or terminal certificates from the same predecessor serial. This directly models the canonical rollback attack in which the same authority is restored and used twice. A separate Destination Integrity property covers misdirection to an ineligible or attacker-selected recipient.

The bearer core separates encrypted, freely replicable object data from the singular authority resource. MOVE creates one successor authority. USE and PRESENT exercise authority without transferring it. DELEGATE creates constrained subordinate capabilities. CONSUME irreversibly converts authority into a terminal certificate. Ordinary MOVE remains device-to-device and does not require witnesses, a transaction registry or consensus.

Recovery is explicitly outside the core. Pure Bearer has no recovery. Challenge-Recoverable permits delayed recovery if no valid descendant or terminal evidence appears. Closure-Recoverable is the preferred peer-resale profile: every committed MOVE emits a typed RETIRE output, and the recipient may asynchronously lodge an opaque recovery nullifier plus a blinded successor commitment with a closure quorum. The resulting Closure Receipt permanently eliminates recovery from the predecessor, provides a temporal anchor for that transition, and can reveal competing successor commitments if a compromised UAP equivocates. Registered-Recoverable is retained only as a non-normative deployment category.

A fresh verifier of a recoverable object must also establish recovery freshness. BOP therefore distinguishes serial-level UAP safety from object-level freshness safety: after a valid recovery finalizes, stale branches may remain physically operable against verifiers using outdated recovery information, but a verifier with sufficiently current recovery state accepts at most one live branch. This limitation is explicit rather than hidden.

The paper also defines bounded closure timing, quorum-signed Closure Receipts, historical hardware-revocation semantics, recovery-evaluator quorums, publication certificates, Recovery Root policies and an optional short-lived watchtower role. Quantum money appears only as motivation for defining unclonability as an abstract property rather than tying the protocol to one secure-element implementation.

1. Scope and Core Thesis

1.1 What BOP is

BOP is a protocol for singular, transferable cryptographic authority over application-defined objects. It is not a cryptocurrency protocol, a global ownership database, or a blockchain with consensus removed. Its core property is local: from one live predecessor state, there should be no way to produce two conflicting irreversible outputs that independently verify.

$$\text{Local output exclusivity} \;!=\; \text{global transaction consensus}$$

The architecture is intended for objects such as licences, tickets, access capabilities, credentials, device permissions, API rights and other application-bound entitlements where exclusive control matters but unrelated events do not need total ordering.

1.2 What BOP does not claim

- No global ordering of unrelated transitions.
- No globally shared smart-contract state or execution VM.
- No native currency, protocol balance system, price mechanism or exchange.
- No guarantee that plaintext already revealed to a user can later be made uncopyable.

- No claim that current secure hardware provides physics-level unclonability.
- No claim that recovery preserves pure bearer semantics without adding external trust, state or liveness.
- No guarantee that a verifier with stale recovery information will reject a branch superseded by recovery.

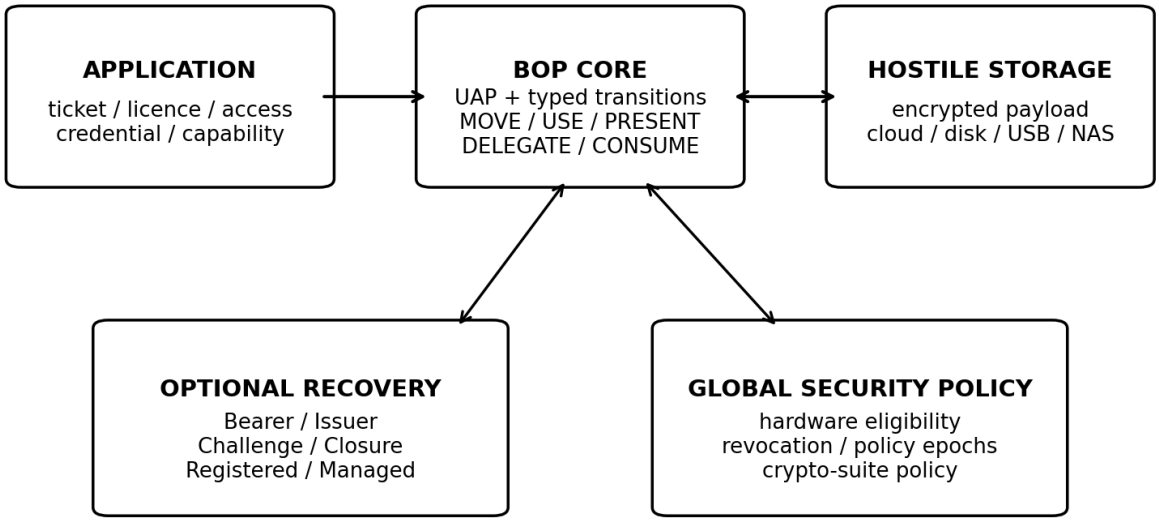


Figure 1. BOP Core is local. Recovery and global security policy are explicit auxiliary control planes.

2. Object and Authority Model

2.1 Replicable data, singular authority

BOP deliberately does not attempt to make the object payload unique. Encrypted payload and metadata may be copied across local storage, cloud storage, USB media or backups. The unique element is the authority state capable of producing valid protocol operations.

$$\text{Unique(Authority)} \;!=\; \text{Unique(Data)}$$

Encrypted payload: freely copyable
Authority state: one live UAP instance per predecessor state
Proof material: portable and independently verifiable

2.2 State structure

A representative root state is:

$$S_i = (\text{OID}, g, s, \text{PK\_move}, \text{PK\_op}, R_i, \text{PolicyRoot}, \text{PayloadRoot}, \text{PolicyEpoch})$$

OID is the stable object identifier. g is recovery generation. s is ordinary root succession sequence. PK_move authorizes exactly one root succession. PK_op authorizes permitted non-root operations. R_i identifies recovery authority where the selected profile supports recovery. PolicyEpoch records the hardware/security-policy epoch relied upon when the state was accepted.

2.3 State serial

The state serial used by the UAP security game is:

$$\sigma_i = H(\text{"BOP-state"} \,\|\, \text{OID} \,\|\, g \,\|\, s \,\|\, H(S_i))$$

The serial identifies a predecessor state for security purposes. It is not a public ownership record and need not be globally registered.

3. Unclonable Authority Primitive and Security Games

3.1 UAP interface

BOP is defined above an abstract Unclonable Authority Primitive rather than directly against a specific TPM, secure element or quantum construction. UAP-H is the implementable classical profile using trusted hardware. The UAP interface exposes typed transitions rather than a generic sign(hash) oracle.

```
CreateRoot()
PrepareReceive()
CommitMove() -> {MoveCertificate, RetireAuthorization}
Use()
Present()
Delegate()
Consume()
Revoke()
```

3.2 Conflicting-output security

The principal UAP security game is defined over outputs, not over whether the adversary constructed two different in-memory authority objects. This captures rollback of the same authority and other state-reuse attacks.

Let C be an irreversible certificate derived from predecessor serial sigma. Relevant classes include MOVE certificates and terminal certificates such as CONSUME or root retirement. The adversary wins if it produces C1 and C2 such that:

$$\text{Verify}(\text{sigma}, C1) = \text{Verify}(\text{sigma}, C2) = 1$$

$$\text{and } \text{Conflict}(C1, C2) = 1$$

Conflict includes two different successor commitments, two distinct terminal outcomes, or a successor and a terminal outcome from the same predecessor. The RETIRE output emitted atomically alongside its matching MOVE is part of the same Transition Bundle and is not a conflicting outcome. Two byte-identical retransmissions of the same certificate are not a conflict.

3.3 Post-retirement operational safety

Conflicting irreversible outputs are not the only rollback failure. A seller must also be unable to keep using a retired predecessor after MOVE. After CommitMove(sigma_i), the adversary may receive fresh verifier challenges and wins if it can produce a valid USE, PRESENT or DELEGATE proof under sigma_i. Old pre-commit proofs do not count because operational proofs bind a fresh verifier nonce or monotonic application challenge.

$$\text{CommitMove}(\text{sigma\_i}) \text{ before } \text{Challenge\_V} \Rightarrow \text{VerifyOp}(\text{sigma\_i}, \text{Challenge\_V}, \text{Proof}) = 0$$

3.4 Destination Integrity

Output exclusivity does not prevent a malicious host from directing the one permitted MOVE to the wrong recipient. Destination Integrity is therefore a separate property. A valid MOVE must bind the exact destination public key, approved hardware-eligibility evidence, object policy and any required trusted user confirmation. A compromised host must not be able to substitute an ineligible or attacker-controlled destination while still obtaining a valid MOVE certificate.

High-value deployments may require trusted-display or protected-confirmation support in addition to cryptographic recipient eligibility.

3.5 Object-level freshness safety

Recovery creates a different class of risk. A finalized recovery may supersede generation g even though a device holding generation g can still physically produce local UAP outputs. The serial-level UAP game therefore cannot by itself describe object-level acceptance after recovery.

Let F be a verifier freshness view containing the recovery-finalization information required by the selected profile. Define Accept_F(OID,S) as the verifier acceptance predicate. BOP requires:

$$\text{Fresh}(F) \Rightarrow |\{ S : \text{Accept\_F}(OID,S) = 1 \}| \leq 1$$

This is a verifier-relative property. A verifier with stale recovery information may accept a branch that has already been superseded. BOP makes this limitation explicit.

4. UAP-H: Classical Secure-Hardware Realization

4.1 Required hardware properties

- Non-exportable root authority keys or equivalent protected state.
- Rollback-resistant persistent state.
- Atomic or crash-safe irreversible transition semantics.
- Secure randomness.
- Verified boot and firmware anti-downgrade.
- Attestation of hardware class and security version.
- Typed command parsing inside the trusted boundary.
- Optional trusted time where a profile explicitly requires time-based offline decisions.

4.2 Split transfer and operational authority

A one-use transfer key alone is insufficient because USE, PRESENT and DELEGATE may require repeated authorization. UAP-H therefore maintains separate transfer and operational authority under one state machine. MOVE atomically retires both authorities for the predecessor root state.

$$\text{MOVE}(S_i) \Rightarrow \text{disable}(PK\_move\_i) \text{ AND } \text{disable}(PK\_op\_i)$$

The host never receives either private key and cannot ask the UAP to sign an arbitrary digest.

5. Hardware Eligibility and Policy Authority

5.1 Device-level HEC

A Hardware Eligibility Credential (HEC) is issued to a device or OSM root, not to every fresh object key. This avoids placing a remote policy server in the MOVE path. The OSM locally generates fresh object keys and certifies them as children of the eligible device root.

$$\text{HEC} = \text{Sign\_HPA}(PK\_OSM, \text{HardwareClass}, \text{SecurityVersion}, \text{Capabilities}, \text{PolicyEpoch}, \text{Expiry})$$

$$\text{ChildCert} = \text{Sign\_OSM}(PK\_object, \text{OID/context}, \text{nonce})$$

The sender verifies the long-lived HEC and the local child certificate. No HEC issuance request is required during ordinary MOVE.

5.2 Linkability

A long-lived OSM credential creates a potential device-linkability channel. Deployments should prefer anonymous or unlinkable hardware-membership proofs where available. The protocol does not assume that such privacy is free; it is a trade-off between revocability, auditability and linkability.

5.3 Hardware-policy authority trust

Compromise of the hardware-policy authority can break recipient eligibility by certifying non-compliant software as a valid UAP-H destination. High-assurance deployments should therefore support threshold or multi-source eligibility policy rather than one universal policy signer.

6. Crash-Safe MOVE

6.1 Five-stage protocol

MOVE uses a recipient-persisted pending successor to avoid consuming the sender before the intended recipient has durably stored the unique successor. The round trip solves that specific atomicity problem; it does not guarantee delivery after the sender disappears.

1. PREPARE: the sender fixes the exact successor key, state commitment, policy epoch and recipient eligibility evidence.
2. PENDING INSTALL: the recipient UAP durably stores the successor as non-usable pending state.
3. PERSISTED ACK: the recipient UAP proves durable storage of that exact pending successor.
4. COMMIT: the sender irreversibly retires the predecessor and atomically emits one Transition Bundle.
5. ACTIVATE: the recipient verifies the MOVE certificate and activates the pending successor.

6.2 Transition Bundle

A successful COMMIT emits exactly one atomic Transition Bundle:

$$TB_i = \{ TC_i, RETIRE_i \}$$

TC_i binds the predecessor to the unique successor. RETIRE_i is a typed recovery-closure authorization for the same predecessor and contains the predecessor serial, the accepted PolicyEpoch and a blinded commitment to TC_i. The UAP creates both as one protocol outcome. Producing MOVE plus its matching RETIRE therefore does not violate the conflicting-output game.

$$C_i = H(r_i \,\|\, H(TC_i))$$

The random salt r_i is delivered only to the transfer parties. The closure system need not learn the successor key or TC_i. A later holder can open C_i by presenting TC_i and r_i when temporal anchoring or equivocation evidence is required.

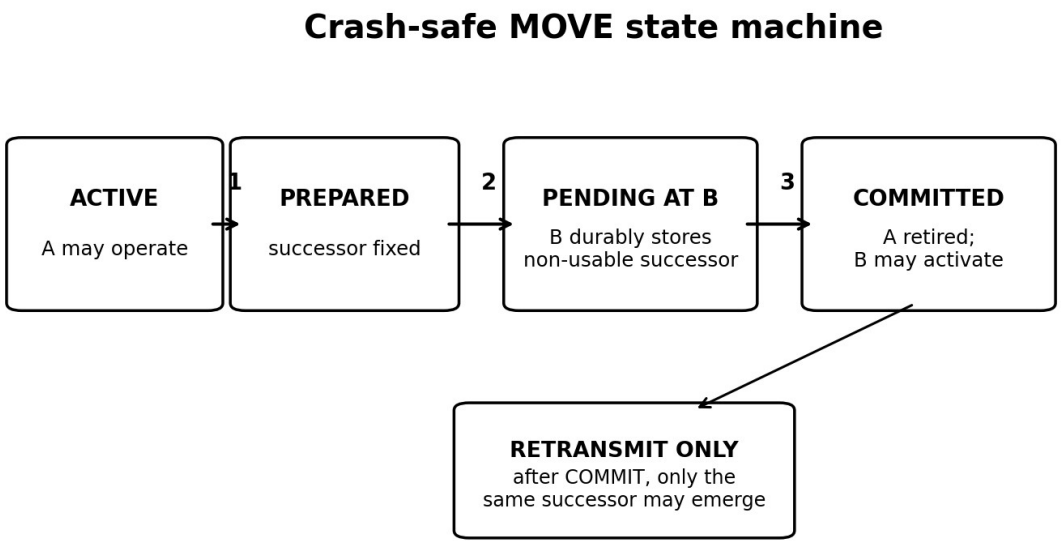


Figure 2. MOVE prevents a second successor; after commit, retransmission may only reproduce the same successor authorization.

6.3 Failure semantics

If the recipient disappears before PERSISTED ACK, the sender remains active. If COMMIT was issued but the commit message and sender device are then lost, availability depends on whether the recipient obtained the commit certificate. If it did, it has valid descendant evidence. If it did not, there is no valid successor evidence and a permitted recovery profile may eventually restore authority. This outcome is intentional.

6.4 No hosted transfer

No BOP service takes possession of live bearer authority while waiting for the recipient. If the recipient cannot participate in MOVE, MOVE does not complete.

7. USE, PRESENT, DELEGATE and CONSUME

7.1 USE and PRESENT

USE exercises a repeatable application capability without changing root authority. PRESENT proves a permitted claim or entitlement. Both use typed UAP commands and are bound to the current root state.

7.2 DELEGATE

Delegations are subordinate capabilities bound to the root state hash or recovery generation and to explicit constraints. A delegation therefore cannot become a new unrestricted root authority.

$$D = (OID, g, RootStateHash, DelegatePK, Actions, Limits, ExpiryOrUseCount)$$

Recovery to a later generation invalidates old-generation delegations for fresh verifiers. Offline delegates may still be accepted by stale verifiers; immediate revocation therefore requires freshness or an application-specific online check.

Delegation and secure time

Time-based BLOCK_ROOT_MOVE policies require trustworthy time. Where the UAP lacks secure time, offline-safe delegation should use use counts, explicit return, or other monotonic conditions rather than host wall-clock expiry. Applications may instead let the verifier enforce time using its own trusted clock.

7.3 CONSUME

CONSUME is an irreversible authority-to-certificate conversion used for tickets, one-time rights and terminal actions. It is not ordinary USE.

$$S_i \text{ --CONSUME--> } TerminalCertificate_i$$

A terminal certificate is conflicting-output evidence for the same predecessor serial and defeats later recovery from that predecessor under profiles that permit such evidence. A recoverable consumable object therefore requires terminal-evidence availability, issuer participation, or a rule that recovery is disabled after first consumption.

8. Global Security Policy and Historical Hardware Revocation

8.1 Global security state is not transaction state

BOP does not require global transaction state, but it cannot avoid shared security-policy state. Accepted hardware classes, compromised firmware versions, cryptographic-suite deprecations and policy epochs are globally relevant security information.

8.2 Policy epoch in every transition

Each root transition records the security PolicyEpoch under which the destination HEC and source hardware were accepted. This makes policy evolution explicit in the lineage.

8.3 Prospective revocation

A prospective revocation applies when a defect makes future use unacceptable but does not permit credible backdating or forgery of already-issued evidence. Transitions anchored under earlier accepted policy epochs remain valid; new transitions from the class are rejected after the effective revocation epoch.

8.4 Catastrophic revocation

A catastrophic revocation applies where key extraction, rollback, forged attestation or another failure destroys the historical evidentiary value of the hardware class. A forged transition can be backdated inside its own signed fields, so PolicyEpoch alone is not a temporal anchor.

For Closure-Recoverable objects, a quorum-signed Closure Receipt accepted before the compromise epoch serves as the normal independent temporal anchor. The holder can later open the receipt's blinded successor commitment with TC_i and r_i, proving that the specific transition commitment existed no later than the closure event. For Bearer or unclosed states, no such guarantee exists unless the application supplied another independently trusted anchor.

Normative rule: after catastrophic revocation, unanchored lineage through the affected class cannot be assumed historically trustworthy. A recoverable profile may re-authorize from the last trusted anchored state. A non-recoverable bearer object may become stranded. The protocol does not pretend this ambiguity can be solved locally.

9. Recovery Is an Optional Profile

Pure singular bearer authority and recoverability pull in opposite directions. BOP Core therefore contains no universal recovery mechanism. Applications select an explicit profile and inherit its external state, liveness and trust costs.

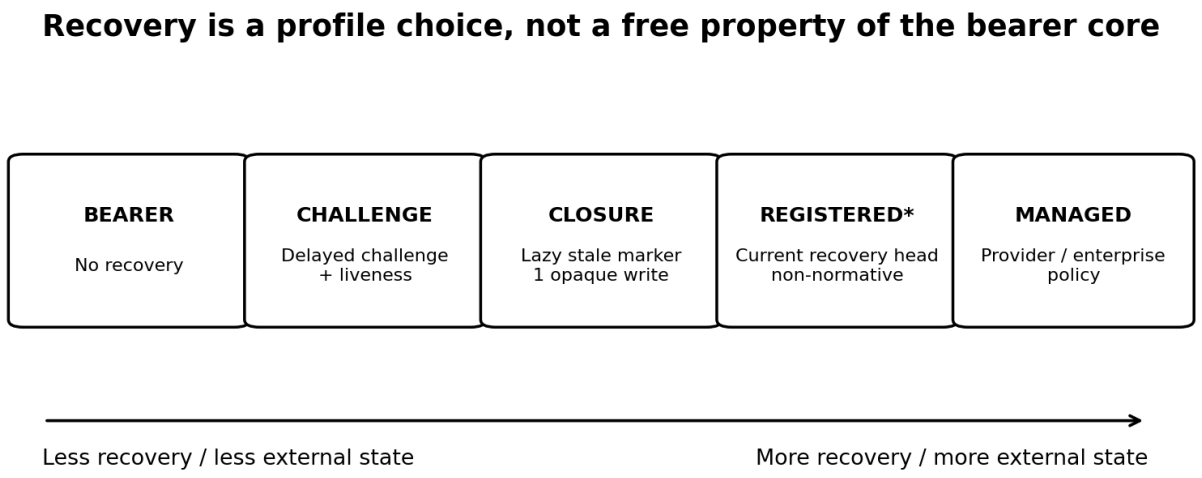


Figure 3. Recovery profiles form a continuum from pure bearer semantics to stronger externally managed recoverability.

Profile	Ordinary MOVE	Recovery mechanism	Primary cost
Bearer	Pure device-to-device	None	Loss may be final
Issuer-Recoverable	Pure device-to-device	Issuer/application adjudication	Issuer trust
Challenge-Recoverable	Pure device-to-device	Delayed public challenge	Holder/watchtower liveness
Closure-Recoverable	Pure device-to-device + lazy closure	Current unclosed state recoverable	One opaque write per transfer
Registered-Recoverable*	Device MOVE + recovery-head update	Application-specific; non-normative	Per-transfer external state
Managed	Application-defined	Provider policy / enterprise controls	Highest external trust

10. Challenge-Recoverable Profile

10.1 Recovery intent

A claimant presents recovery authorization tied to a claimed predecessor state S_c and a new eligible destination. A threshold recovery-evaluator set checks the private evidence. If preliminarily valid, a public recovery intent (PRI) is published for a challenge interval.

10.2 Defeating evidence

Any valid irreversible output from S_c defeats recovery. This includes a valid MOVE successor certificate or a terminal CONSUME certificate. A defeated state receives a permanent stale marker and can never be used to initiate recovery again.

$$\text{DescendantOrTerminalProof}(S_c) \Rightarrow \text{STALE}(S_c) \text{ forever}$$

10.3 Recovery finality

If no valid defeating evidence is received during the complete challenge interval and the required evaluator quorum authorizes finalization, recovery advances the generation. The new generation strictly dominates every branch in the earlier generation for fresh verifiers. Late evidence cannot reverse finalized recovery.

$$\text{Finalize}(g \to g+1) \Rightarrow \text{generation } g \text{ is permanently stale for fresh verifiers}$$

10.4 Recovery Availability Assumption

Challenge recovery is safe only if a legitimate current holder, or an agent acting for that holder, can observe and answer stale recovery attempts during the challenge interval. This is a lifetime liveness obligation for every unclosed historical predecessor capable of initiating recovery. That cost is the principal reason Challenge-Recoverable is not the preferred profile for long-lived resaleable objects.

11. Closure-Recoverable Profile

11.1 Structural model: a recovery nullifier set

Closure-Recoverable converts each retired predecessor into one append-only opaque record. Structurally this is a nullifier or spent-serial set: once a predecessor serial is closed, that historical state can never again serve as the basis of recovery. Unlike payment nullifier systems, ordinary MOVE validity does not depend on consulting the set; the set governs recovery and supplies auxiliary evidence.

At COMMIT, the UAP emits RETIRE_i as part of the Transition Bundle. The closure submission contains the predecessor serial, PolicyEpoch and blinded successor commitment C_i, together with RETIRE_i. A naked state hash is not sufficient to close recovery rights.

$$\text{ClosureRecord}_i = (\sigma_i, C_i, \text{PolicyEpoch}_i, \text{RETIRE}_i)$$

$$C_i = H(r_i \,\|\, H(TC_i))$$

The closure quorum verifies RETIRE_i but need not see the successor key or TC_i. Once accepted, recovery from S_i is permanently invalid.

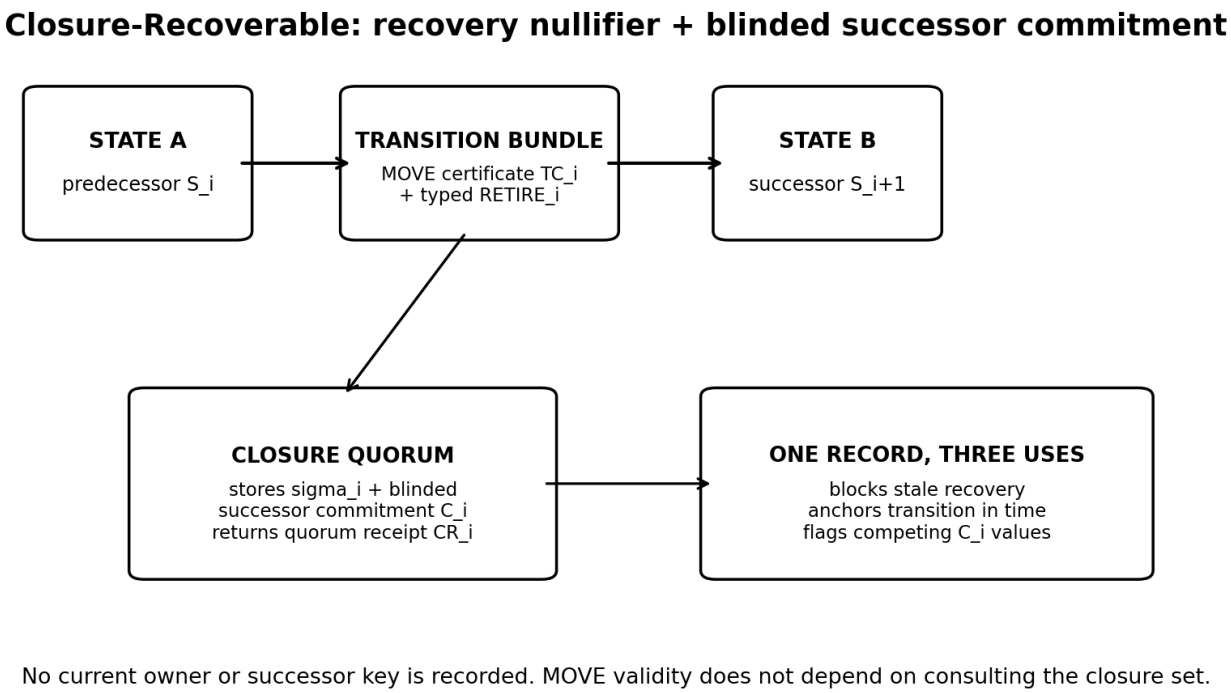


Figure 4. Closure-Recoverable is an append-only recovery-nullifier set with a blinded successor commitment.

11.2 Closure quorum and durable receipt

The closure service is replicated. A profile defines n_C closure replicas and threshold q_C. A valid closure receipt CR_i contains q_C signed acceptances over the same record and an authenticated closure time or closure epoch.

$$CR_i = q\_C\text{-of-}n\_C \text{ signatures over } (sigma\_i, C\_i, PolicyEpoch\_i, ClosureEpoch)$$

Closure nodes cannot fabricate a valid retirement without RETIRE_i, but they can censor or later deny a write. A quorum receipt makes a successfully closed predecessor self-proving: the holder, a later descendant, a watchtower or a recovery evaluator can present CR_i even if some closure replicas later deny the record.

11.3 Bounded closure obligation

Closure is asynchronous but not unbounded. Let Delta_C be the maximum permitted delay from successful receipt to submission, Delta_P the maximum propagation/confirmation allowance, Delta_M a safety margin, and W_R the recovery challenge interval. Closure-Recoverable requires:

$$W\_R > Delta\_C + Delta\_P + Delta\_M$$

Until CR_i is obtained, the new holder remains exposed to a stale recovery claim from the predecessor and must satisfy the Recovery Availability Assumption directly or through a watchtower. An offline receipt therefore has provisional recovery protection until the holder reconnects and obtains a Closure Receipt.

11.4 Equivocation detection

If a compromised UAP emits two different MOVE certificates from the same predecessor, two honest recipients derive different blinded successor commitments C_i^A and C_i^B. If both submit closure, the closure quorum observes two different commitments for the same sigma_i and can return cryptographic equivocation evidence. Closure therefore provides conditional automatic detection when competing recipients both seek closure; it does not guarantee detection if only one branch is ever reported.

11.5 Temporal anchoring

The Closure Receipt also timestamps or epochs the blinded successor commitment. If the relevant hardware class is catastrophically compromised later, a holder can reveal TC_i and r_i and show that H(r_i || H(TC_i)) equals the commitment recorded in a pre-compromise CR_i. This makes the ordinary closure record the historical temporal anchor for Closure-Recoverable objects and removes the need for a separate periodic anchor mechanism in the normal profile.

11.6 Recovery defeat without revealing the transition

For ordinary stale-recovery rejection, evaluators need only verify that a valid Closure Receipt exists for sigma_i. They do not need the successor state or TC_i. The successor commitment is opened only when needed for equivocation or catastrophic-revocation evidence.

11.7 Repairing gaps and recovery-root compromise

RETIRE authorizations and closure receipts travel with the proof package. If a holder fails to close a predecessor, a later legitimate descendant may close that missing state. Closure also bounds the blast radius of recovery-root compromise: historical states already closed cannot be used for future recovery claims even if an old recovery factor is later stolen.

11.8 Privacy

The closure set reveals that an opaque predecessor serial was retired and exposes timing/volume metadata. It does not need the current owner, successor key, application meaning, monetary value or payload. Delay and batching may reduce timing correlation. This is still external per-transfer state for the Closure-Recoverable profile and is acknowledged as such.

12. Other Recovery Profiles

12.1 Bearer

Bearer mode has no recovery infrastructure. Device loss may mean object loss. This is the cleanest approximation to physical bearer possession and imposes no recovery freshness requirement on recipients.

12.2 Issuer-Recoverable

An issuer or application authority adjudicates exceptional recovery according to domain rules. This fits tickets, licences, credentials and access rights where an issuer already participates in verification or lifecycle management.

12.3 Registered-Recoverable

Registered-Recoverable is a non-normative deployment category in v0.6. An application may maintain an opaque current recovery head after every root MOVE and thereby obtain faster recovery at the cost of per-transfer external state and privacy exposure. BOP Core does not specify a canonical registered-recovery protocol.

12.4 Managed

Managed recovery permits provider or enterprise policy to participate directly in recovery. This can offer the best UX but should be described as managed authority rather than pure bearer self-custody.

13. Recovery Freshness for Recipients and Verifiers

13.1 Receipt-time freshness

A recipient accepting a MOVE under any recovery-capable profile must determine whether the presented branch has been superseded by finalized recovery, unless the recipient explicitly accepts stale-recovery risk. This is mandatory for strong finality.

13.2 Recovery Finalization Set

Because finalized recoveries should be sparse relative to ordinary transfers, a deployment may publish a signed Recovery Finalization Set (RFS) or compact authenticated representation. Devices can download the complete current set and check locally rather than querying an OID-specific endpoint for every receipt. The RFS necessarily contains an OID-derived or otherwise object-matchable recovery handle, so a party already possessing the OID can determine that the object underwent recovery. This is an explicit privacy cost of recoverability.

Higher-privacy deployments may use private-information-retrieval mechanisms. An offline verifier may rely on its last signed RFS snapshot, but its security is bounded by that snapshot's age and should be surfaced as a freshness level F.

13.3 Freshness levels

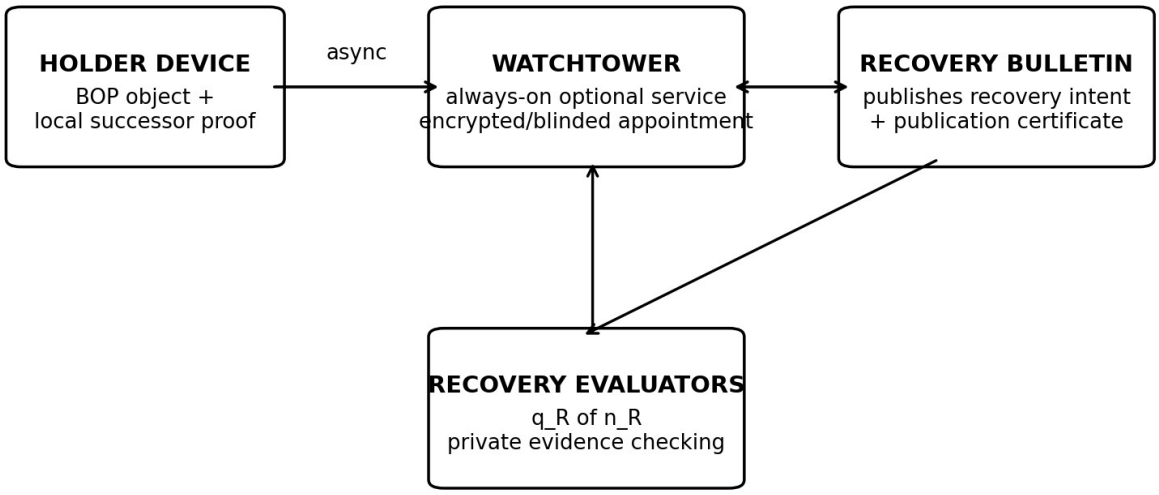
F0 no recovery freshness: bearer-only assurance
F1 cached signed RFS snapshot: bounded-staleness assurance
F2 current RFS or privacy-preserving lookup: fresh-recovery assurance
F3 application-specific online issuer/registered check: strongest profile-specific assurance

Applications should specify the minimum freshness level required for RECEIVE, USE, PRESENT or CONSUME.

14. Watchtowers: Optional Recovery-Veto Availability

14.1 Where the watchtower sits in reality

A watchtower is not a consensus node, custodian or transaction relay. It is an optional always-on service outside the MOVE path whose sole function is to monitor the recovery bulletin and submit valid stale-successor or terminal evidence during a challenge window. It may run on a user's home server or NAS, an issuer backend, an enterprise service, or one or more independent cloud operators.



The tower is never in MOVE. It only watches recovery intents and can submit stale-successor evidence.

Figure 5. A watchtower sits beside the recovery control plane, never between the two devices performing MOVE.

14.2 Appointment lifetime

Under Closure-Recoverable, an appointment only needs to survive the bounded interval from receipt until a quorum-signed Closure Receipt is obtained. The tower can then delete the appointment. Under pure Challenge-Recoverable, the liveness obligation may persist for the lifetime of the unclosed predecessor state.

14.3 Blinding requirement

An appointment locator derived from public lineage state is insufficiently blind. Any deployment claiming watchtower privacy must derive appointment locators and decryption material from per-transfer secret material that is not included in downstream lineage. The exact blind-appointment construction is intentionally non-normative in v0.6 and should receive separate cryptographic analysis before production use.

Candidate building blocks include oblivious pseudorandom functions and verifiable random functions, but BOP Core does not depend on one construction. RFC 9497 specifies VOPRF/OPRF mechanisms; RFC 9381 specifies VRFs.

15. Recovery Evaluators and Bulletin Integrity

15.1 Threshold evaluators

Recovery is not decided by a single evaluator. A recovery profile defines n_R evaluators and a threshold q_R. Evaluators privately validate the claimed state, recovery authorization, closure status, destination eligibility and any descendant or terminal objections.

15.2 Publication quorum

A separate bulletin quorum provides availability evidence. A challenge interval begins only after a PublicationCertificate proves that the PRI was accepted by the required bulletin replicas with a defined start and end time.

PublicationCertificate = q_B-of-n_B attestations over PRI hash and challenge window

Finalization requires both the recovery-evaluator decision and valid publication evidence for the full challenge interval. A holder or tower therefore only needs access to one honest bulletin replica during the window, under the bulletin availability assumption.

15.3 Public versus private recovery data

The public bulletin should reveal only an opaque challenge identifier or blinded recovery handle, challenge deadline, publication certificate and final result. Claimed lineage, transfer certificates and recovery credentials are submitted privately to evaluators. A successful challenge need only publish DEFEATED plus the permanent stale marker, not the ownership history.

16. Recovery-Key Architecture and Blast Radius

16.1 Per-object recovery authority

A recoverable state includes a recovery authority R_i. The protocol does not require every object to use one unrelated manually backed-up key, nor does it require one universal account root.

16.2 Recovery Root options

A consumer implementation may derive per-object recovery authorities from a protected Recovery Root or hierarchical recovery domain. This improves UX but increases blast radius: compromise of a single root may permit claims against many current or historical states.

16.3 Recommended profiles

- Low-value consumer objects: managed or account-assisted sub-root may be acceptable.
- Resaleable objects: use Closure-Recoverable so sold historical states are permanently closed.
- High-value objects: isolated per-object or per-domain recovery factors, preferably threshold protected.

- If the holder still controls an object but suspects recovery-factor compromise, rotate R_i through an authenticated self-successor and close the predecessor recovery state.

16.4 Loss of both holder and recovery authority

If both live holder authority and all required recovery factors are lost, recovery intentionally fails. An administrative master override would simply move ultimate custody into the administrator and is therefore not part of BOP Core.

17. CONSUME and Recovery Policy

Consumable rights require an explicit recovery policy because local terminal consumption and later recovery from the pre-consumption state would otherwise enable double use.

Policy	Rule
Bearer terminal	CONSUME is final; no recovery from consumed predecessor.
Evidence-backed recovery	Terminal certificate is lodged with closure/challenge infrastructure and permanently defeats recovery from the consumed state.
Issuer-mediated recovery	Issuer or verifier participates in deciding whether the right may be reissued after consumption/loss.

A profile must not advertise both recoverable consumption and fully private/offline terminal use without also specifying how terminal evidence becomes available to the recovery decision.

18. Hostile Storage and Application Isolation

18.1 Encrypted payloads

Object payloads are encrypted before reaching local filesystem or cloud storage. Storage providers may copy or inspect ciphertext but do not receive the UAP authority or object data-encryption keys by default.

18.2 Opaque application handles

Applications receive opaque handles and typed operations, not exportable holder keys. A compromised application should not automatically become a root-authority compromise.

```
handle.present(challenge)
handle.use(action)
handle.delegate(policy)
handle.move(destination)
handle.consume(verifierChallenge)
```

18.3 Host OS limitation

BOP can protect bearer authority from the ordinary host, but content that must be rendered in plaintext expands the trusted computing base. Capability-only objects can often remain opaque to the host; document/content objects cannot make that universal guarantee.

19. Threat Model and Security Assumptions

The core adversarial objectives are: produce conflicting irreversible outputs, continue operational use after retirement, redirect the unique MOVE to an unauthorized destination, resurrect obsolete authority, or make stale authority appear current to a verifier.

Assumption	Meaning
A1 Cryptography	Required signatures/hashes cannot feasibly be forged or collided.
A2 UAP output exclusivity	One predecessor serial cannot yield conflicting valid

	irreversible certificates.
A3 Rollback resistance	Consumed/retired predecessor state cannot be restored as active or answer fresh operational challenges.
A4 Typed execution	The UAP only authorizes protocol-valid operations and exact successor commitments.
A5 Eligibility integrity	Accepted destination eligibility evidence cannot be forged below the selected policy threshold.
A6 Recovery threshold	Fewer than the tolerated evaluators collude or violate policy.
A7 Bulletin availability	A valid challenge is observable through at least one honest bulletin path during its certified window.
A8 Freshness	The verifier obtains the recovery freshness required by its selected profile.
A9 Closure quorum	For Closure-Recoverable, enough closure replicas remain available and do not sign contradictory receipts.

19.1 Equivocation evidence

If A2/A3 fails and two conflicting outputs from one predecessor serial are later compared, the pair is cryptographic evidence of equivocation. BOP does not claim automatic detection: without a monitor, registry or shared recipient, conflicting branches may never meet.

20. Core Invariants

I1: Valid(C1, sigma) and Valid(C2, sigma) and Conflict(C1,C2) => security failure

I2: CommitMove(S_i,S_j) => predecessor S_i cannot authorize another conflicting irreversible output

I3: CommitMove(sigma_i) before fresh Challenge_V => no valid USE/PRESENT/DELEGATE proof under sigma_i

I4: DestinationIntegrity => committed successor key and eligibility evidence equal the authorized destination

I5: FinalizeRecovery(g -> g+1) => fresh verifiers reject every branch rooted solely in generation g

I6: ClosureReceipt(sigma_i) => recovery from S_i is permanently invalid

I7: TerminalCertificate(sigma_i) => recovery from S_i is invalid unless an explicit issuer-mediated policy overrides terminality

I8: Copy(ciphertext) does not imply Copy(authority)

I9: Fresh(F) => at most one branch for an OID is accepted under F

21. Implementation Plan: Stop Writing and Build

21.1 Phase 1 - UAP-H emulator

Build the exact state machine in a deterministic software emulator. The objective is not production security; it is to validate transition semantics, crash behavior, proof formats and adversarial games. The first test suite MUST attempt MOVE_A+MOVE_B, MOVE+CONSUME, CONSUME_A+CONSUME_B, MOVE followed by fresh USE/PRESENT/DELEGATE, closure versus stale recovery, competing closure commitments, recovery finalization versus stale generation, and every crash point in CommitMove().

21.2 Phase 2 - Programmable secure applet

Implement UAP-H on programmable secure hardware rather than relying solely on generic sign-key APIs. The applet must enforce typed commands, root/operational authority coupling, pending successor storage, atomic retirement and rollback resistance.

21.3 Phase 3 - empirical hardware matrix

Measure which shipping platforms actually satisfy A2-A5: rollback-resistant storage, fault behavior, protected confirmation, secure time where required, update/attestation behavior and storage limits. Platform claims should be based on experiments, not SDK descriptions alone.

21.4 Phase 4 - recovery prototype

Implement Challenge-Recoverable and Closure-Recoverable separately. Measure Delta_C, closure-quorum latency, batch sizes, timing leakage, challenge UX, RFS freshness and bulletin/evaluator failure behavior. The watchtower can initially be a simple local or cloud daemon protecting the pre-closure interval; blind-appointment cryptography should be added only after the base recovery state machine is stable.

21.5 Phase 5 - formal model

Model conflicting irreversible outputs, post-retirement operational safety, destination integrity, closure-quorum behavior, recovery finality and freshness-relative acceptance in a protocol/state-machine verification environment. After this pre-MVP specification, the paper should change only when implementation or formal analysis finds a defect.

22. Practical Deployment Topology

22.1 Personal deployment

A technically sophisticated user can run a watchtower on an always-on NAS, home server or VPS. It requires no object custody and no ability to sign as the holder.

22.2 Issuer deployment

For tickets, licences and access rights, the issuer may operate a tower because the issuer already has an availability role. If the same issuer also controls recovery adjudication, the trust profile should be labelled Issuer-Recoverable rather than presented as independent bearer recovery.

22.3 Independent service

A commercial tower can store blinded appointments for many users. Under Closure-Recoverable those appointments are short-lived and deletable after predecessor closure, reducing the tendency toward a long-term ownership registry.

22.4 Multi-tower

A holder may duplicate encrypted appointments across several independent towers. Because towers do not vote on state, this is redundancy rather than consensus. Any one tower capable of producing valid defeating evidence can stop a stale recovery attempt.

23. Prior Art and Positioning

BOP does not claim the invention of tamper-resistant offline scarcity, transferable usage rights, time-bounded leases, secure hardware attestation, or one-shot authorization concepts. The design deliberately builds on known ideas and narrows its contribution to their composition and the explicit separation of bearer core from recovery profiles.

23.1 Wallets with observers

Chaum and Pedersen described tamper-resistant observer components inside wallets decades before modern secure elements. This is important prior art for the idea that a small trusted module can enforce rules while the surrounding host remains less trusted.

23.2 OMA DRM

OMA DRM 2.0 separated encrypted content from transferable Rights Objects and defined a Move export mode in which the original right becomes unusable while protected content may remain. This is structurally close to replicable encrypted payload plus constrained transferable authority.

23.3 OPERA and offline cash

OPERA demonstrated offline peer-to-peer digital cash using one-time-readable secure memory and is direct prior art for hardware-enforced singularity without a transaction ledger.

23.4 Leases

Gray and Cheriton formalized time-bounded leases as a distributed-systems primitive in 1989. BOP does not claim the lease concept; recovery-exclusion leases are merely one application-specific use where a deployment chooses that profile.

23.5 Spent-serial and nullifier sets

Chaumian electronic-cash systems used spent-coin or serial databases to prevent repeated redemption, and modern privacy-preserving payment systems such as Zerocash/Zcash use public nullifier sets whose uniqueness prevents double spending. Closure-Recoverable is structurally similar because it maintains an append-only set keyed by spent predecessor serials. The distinction is functional: a BOP closure record does not determine ordinary MOVE validity or current ownership; it prevents a retired state from serving as a future recovery origin and can optionally bind a blinded successor commitment.

23.6 Watchtower pattern

Payment-channel watchtowers provide a useful availability pattern: encrypted or otherwise privacy-limited appointments allow an online agent to react to a bad state while the owner is offline. BOP does not assume Lightning-style privacy is automatically obtained; its watchtower blinding requires a separate construction.

24. Quantum-Money Motivation: Narrow Role

Quantum money is useful to BOP primarily as an idealized security analogy. Quantum no-cloning suggests a clean way to specify the desired property before choosing a hardware implementation: one valid bearer authority should not be duplicable into two independently usable outcomes. Quantum-lightning and one-shot-signature research also motivate serial-based security games and destructive authority-to-certificate transitions.

No security argument for UAP-H relies on quantum storage, quantum communication or a particular quantum-money candidate. A practical UAP-Q would additionally need a quantum channel or equivalent quantum-state-transfer mechanism and would require definitions for PRESENT, DELEGATE and application interaction that are outside this paper. Several historical public quantum-money/lightning candidates have also faced cryptanalysis or rest on non-standard assumptions. Accordingly the quantum material is positioning and future-interface motivation, not an implementation claim.

25. Open Questions for Prototype Review

1. Which shipping secure elements can enforce the full UAP-H state machine, including atomic retirement of transfer and operational authority?
2. What q_C/n_C closure topology gives acceptable censorship resistance, latency and deployment cost?
3. What values of Delta_C, Delta_P, Delta_M and W_R provide practical recovery UX without making offline receipt misleading?
4. What batching and delay strategy makes closure writes difficult to correlate with physical transfers?
5. Can blind watchtower appointments be specified with a standard OPRF/VRF construction without introducing linkability or denial-of-service weaknesses?

6. What recovery-freshness representation provides the best privacy/size trade-off: full sparse set, authenticated accumulator, PIR-backed lookup, or another mechanism?
7. Which hardware failures should be classified as prospective versus catastrophic revocation in the empirical hardware matrix?
8. Which application classes genuinely benefit from Closure-Recoverable rather than simpler Bearer or Issuer-Recoverable profiles?

26. Conclusion

Bearer Object Protocol v0.6 freezes the architecture at the point where implementation should become the primary source of truth. The core is a locally serialized bearer-authority system defined against an Unclonable Authority Primitive. Its principal games cover conflicting irreversible outputs, post-retirement operational use and destination integrity. Recovery freshness remains a separate verifier-relative property.

Recovery is optional. Challenge-Recoverable preserves fully unregistered MOVE but imposes liveness. Closure-Recoverable is the preferred peer-resale profile: one asynchronous opaque closure record per retired predecessor acts simultaneously as a recovery nullifier, a blinded successor commitment, a conditional equivocation detector and a temporal anchor. A quorum-signed Closure Receipt converts the holder's recovery-availability obligation from lifetime monitoring into a bounded post-receipt interval.

The architecture still acknowledges hard limits: finalized recovery may leave stale physical branches usable against stale verifiers; catastrophic hardware compromise can invalidate unanchored history; and recoverable terminal consumption needs externally available terminal evidence. These are explicit system boundaries, not hidden assumptions.

BOP Core = UAP + typed irreversible transitions + proof-carrying succession

Closure-Recoverable = local MOVE + lazy recovery nullifier + quorum Closure Receipt

Recovery = optional profile with explicit trust, state, privacy and liveness costs

This is the frozen pre-MVP specification. The next work is the UAP-H emulator/applet, closure-quorum service, adversarial fault testing and empirical hardware matrix. Future paper revisions should be driven by implementation or formal-analysis findings.

References

Chaum, D. and Pedersen, T. P. Wallet Databases with Observers. CRYPTO 1992. https://chaum.com/wp-content/uploads/2021/12/Wallet_Databases.pdf

Chaum, D., Fiat, A. and Naor, M. Untraceable Electronic Cash. CRYPTO 1988.

Ben-Sasson, E. et al. Zerocash: Decentralized Anonymous Payments from Bitcoin. IEEE Symposium on Security and Privacy, 2014. https://eprint.iacr.org/2014/349.pdf

Zcash Protocol Specification. Nullifier Sets. https://zips.z.cash/protocol/protocol.pdf

Gray, C. and Cheriton, D. Leases: An Efficient Fault-Tolerant Mechanism for Distributed File Cache Consistency. SOSP 1989. https://web.stanford.edu/class/cs240/WWW/readings/89-leases.pdf

Open Mobile Alliance. DRM Specification V2.0. https://www.openmobilealliance.org/release/DRM/V2_0-20050712-C/OMA-TS-DRM-DRM-V2_0-20050712-C.pdf

Park, K.-W. and Baek, S. H. OPERA: A Complete Offline and Anonymous Digital Cash Transaction System with a One-Time Readable Memory. IEICE, 2017. https://www.jstage.jst.go.jp/article/transinf/E100.D/10/E100.D_2016INP0008/_article

NIST FIPS 204. Module-Lattice-Based Digital Signature Standard (ML-DSA). https://csrc.nist.gov/pubs/fips/204/final

NIST FIPS 205. Stateless Hash-Based Digital Signature Standard (SLH-DSA). https://csrc.nist.gov/pubs/fips/205/final

RFC 9497. Oblivious Pseudorandom Functions (OPRFs) Using Prime-Order Groups. https://www.rfc-editor.org/rfc/rfc9497.html

RFC 9381. Verifiable Random Functions (VRFs). https://www.rfc-editor.org/rfc/rfc9381.html

Android Developers. KeyGenParameterSpec.Builder / setMaxUsageCount. https://developer.android.com/reference/android/security/keystore/KeyGenParameterSpec.Builder

Apple Developer Documentation. SecureElementCredential. https://developer.apple.com/documentation/SecureElementCredential

Zhandry, M. Quantum Lightning Never Strikes the Same State Twice. 2017. https://arxiv.org/abs/1711.02276

Ben-David, S. and Sattath, O. Quantum Tokens for Digital Signatures. 2016. https://arxiv.org/abs/1609.09047