# Effective Descent for Distributed Archives with Auditable Hybrid Search

Anonymous

8 November 2026 • Manuscript v1.0

**Abstract**

A distributed archive can answer a keyword or semantic query before all its index shards are reachable. A client then needs distinct evidence for a particular result, the completeness of its ranking, and the recoverability of the cited PDF. We give a space-valued semantics for these claims over a finite site of snapshot obligations. The associated $\infty$-sheaf retains verification histories up to explicitly permitted rewrites. Its descent theorem yields finite completion certificates even when the witnesses arrive in nonoverlapping network windows. A fixed-order top-$K$ merge law and a hash-checked Reed–Solomon reconstruction theorem specify exact search and byte recovery. A conditional opportunity bound relates asynchronous retries to deadline probabilities and exposes the effect of shared outages. We specify platform admission through hash, signature and audit checks and give a JVM reference implementation with a platform-native launcher build. Tests cover image-only PDFs, token and concept search, staggered peer availability, reconstruction, repair, and audit rollback and equivocation.

**Keywords** $\infty$-sheaves, distributed archives, semantic retrieval, erasure coding, audit trails, virtual machines

## Contents

**1 The problem and the claims** **2**

**2 The finite site of evidence** **3**

**3 Authenticated PDF restoration** **5**

**4 Keyword, semantic and hybrid search** **6**

**5 Delivery probability and latency** **8**

**6 The operational protocol** **9**

**7 From specification to platform binaries** **11**

**8 Reference implementation and evaluation** **14**

**9 Related systems and conclusion** **15**

**A Binary framing and service interface** **16**

# 1 The problem and the claims

An archive should remain useful when its operators cannot maintain a continuously available fleet. A PDF might survive on four of six devices, while a searchable representation of its text is on a different set of devices. A query should be able to show a verified passage now, add new passages later, and finish when the evidence covers its declared search scope. The time horizon may range from seconds to days. An application that returns an empty list during an outage and an application that certifies an empty answer are making different mathematical claims.

We index all claims by an authenticated, immutable context. This makes three facts independently checkable. A passage is bound to an indexed page in a snapshot. A ranked answer is complete relative to a specified scorer and the eligible pages in that snapshot. An original PDF is recoverable from verified encoded fragments. A search completion certificate may precede the restoration of any cited PDF. Reproducibility of extraction and measurement of relevance are separate questions about the index.

Table 1: Claims and the evidence a verifier checks.

| Claim | Positive certificate | What can remain open at that point |
|---|---|---|
| A passage is indexed | Snapshot signature, index commitment, page binding and score recomputation | Other shards and original PDF bytes |
| Exact result under a scorer | All eligible shards verified, or sound exclusion certificates for uncovered candidates | Human relevance and availability of originals |
| Original PDF is recovered | Verified distinct fragment slots, decoding and matching document digest | Freshness of the snapshot and correctness of its extracted text |
| Audit trail extends a pin | Valid signed checkpoint and a proof of prefix extension | Forks shown only to clients outside the pinned view |

The setting is a finite corpus per snapshot, eventual contacts among peers, and untrusted transport and storage. Operators may drop, replay, equivocate, or corrupt responses. A client starts with a publisher trust anchor, a selected snapshot, and a deterministic verification profile. The publisher sets the corpus and the search policy. A valid signature binds those choices to a key. It gives no independent assertion that a passage was transcribed correctly or a document belongs in the corpus. The reliability theorems are conditional on verifiable delivery and explicitly stated probability hypotheses. The reference implementation exercises the protocol with independent peer processes and synthetic PDFs.

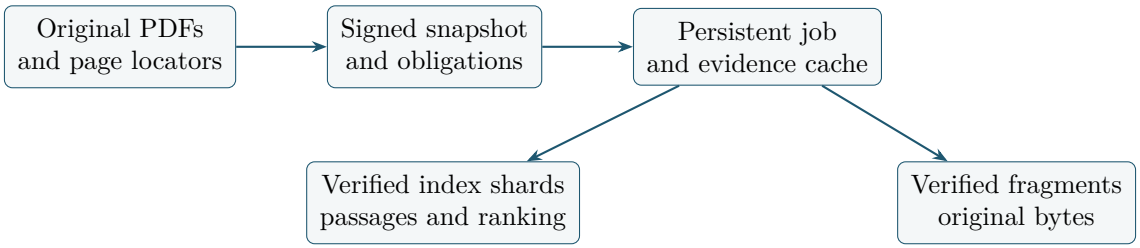



Figure 1: One snapshot binds two independent retrieval paths. Each path may collect evidence over several periods of connectivity.

# 2 The finite site of evidence

## 2.1 Contexts and obligations

**Definition 2.1** (Pinned context)**.** A context $\gamma$ consists of an archive identifier, an authenticated snapshot root, a policy identity or authorization scope, and an operation with its fixed evaluator. A query operation includes its normalized query, mode, $K$, model and statistics identifiers, and tie order. A restoration operation specifies a document and a codec profile. Each $\gamma$ determines a finite set $R_\gamma$ of obligation coordinates.

For full search $R_\gamma$ may be all eligible index shard slots. A shard slot is discharged by a validated whole block or by a verified proof for the answer claimed for that shard. For restoration use one coordinate per encoded slot, together with any manifest obligations the client has not already discharged. A restoration predicate then asks for at least $k$ distinct verified slots of each stripe, while search coverage in the full-block profile asks for all eligible slots. These are predicates on evidence over one site. The site itself does not declare every cover physically attainable.

Put $\mathcal{B}_\gamma = \mathcal{P}(R_\gamma)$ ordered by inclusion. A family $(U_i \hookrightarrow U)_{i \in I}$ covers when $\bigcup_i U_i = U$. The empty family covers $\varnothing$. Pullback is intersection, so pullback preserves covers, and taking unions verifies transitivity. We treat each epoch and policy as part of the context. A client can transport witnesses between contexts when it is given explicit equalities of relevant commitments and compatibility of verification rules. Merely seeing a higher epoch number supplies no such morphism.

At coordinate $r$, let $A_r$ be a discrete set of canonical verified assertions, such as committed index bytes or a validated exclusion bound. Let $T_r$ be a small category of finite accepted transcripts. An object includes the assertion it realizes. Generating arrows encode permitted transcript rewrites, for example removing redundant local hash rechecks or commuting two independent read-only checks. An arrow is permitted only when its guard preserves the assertion and all observations promised by the profile. One may take the free category on these generators modulo checked commuting diamonds. Let $W_r$ be a subcategory generated by the reversible-for-this-observation rewrites and identities. Define

$$E_r = \left(T_r[W_r^{-1}]\right)^{\simeq}, \qquad E_\gamma(U) = \prod_{r \in U} E_r.$$

Here $^{\simeq}$ denotes the maximal $\infty$-groupoid and localization is in the $\infty$-categorical sense. The assertion functor descends to a map $E_r \to A_r$. This use of relative categories follows the localization viewpoint of Dwyer and Kan and of Barwick and Kan [1, 2]. No inverse is added to a physical write, a revocation, a changed conclusion, or a causally ordered audit event. An implementation may retain complete raw logs even when the observation functor identifies redundant verifier steps.

For example, two peers can deliver copies of a committed index block with different authenticated transport histories. They realize the same assertion. If the profile observes source identity, their source records remain distinct. Checking the block hash and checking a pinned receipt signature can commute when both are read-only and take immutable inputs. A path expressing that square witnesses independence of checks. It does not permit reordering two signed entries in the peer's own audit chain.

## 2.2 Descent and effective completion

**Theorem 2.2** (Descent over a snapshot)**.** *The presheaf $E_\gamma\colon \mathcal{B}_\gamma^{\mathrm{op}} \to \mathcal{S}$ satisfies $\infty$-sheaf descent. For a cover $(U_i \hookrightarrow U)$, the restriction map*

$$E_\gamma(U) \longrightarrow \operatorname*{holim}_{[p]\in\Delta} \prod_{i_0,\dots,i_p} E_\gamma(U_{i_0} \cap \cdots \cap U_{i_p})$$

*is an equivalence. Evaluation at the singleton obligations identifies $\mathrm{Sh}_\infty(\mathcal{B}_\gamma)$ with $\mathcal{S}^{R_\gamma}$.*

*Proof.* For fixed $r \in U$, put $I_r = \{i : r \in U_i\}$, which is nonempty. Rearranging the products separates the Čech diagram into factors $E_r^{I_r^{p+1}}$. The simplicial set $[p] \mapsto I_r^{p+1}$ is the nerve of the groupoid with exactly one morphism between each ordered pair of elements of $I_r$. It is contractible. The limit of the corresponding cotensor diagram is therefore equivalent to $E_r$. Taking the product proves the displayed assertion. Conversely, the singleton cover of $U$ has empty intersection between distinct singletons. Any sheaf takes $\varnothing$ to a terminal object and its descent equivalence for that cover gives the product of its singleton values. These equivalences are natural for inclusions and for maps of sheaves. This gives the stated equivalence of $\infty$-categories. □

This is a finite product $\infty$-topos, using space-valued descent as in [3]. Its size and topology reflect the choice to put global constraints in the signed context before distributing obligations. If two index shards used different normalization rules, the resulting data would not be sections for the same context. If the answer required a joint constraint beyond those local checks, it would need an additional obligation or a different site. The Grothendieck construction on $E_\gamma$ organizes witnesses in a cartesian fibration over $\mathcal{B}_\gamma$. Proof-relevant histories inhabit its fibers. Propositional truncation gives a separate readiness assertion $\mathsf{Ready}_r = \|E_r\|_{-1}$ at each coordinate.

**Definition 2.3** (Effective verifier profile)**.** The profile gives decidable encodings of finite contexts, obligations and bounded candidate witnesses. For each candidate, a verifier terminates with acceptance and the canonical assertion, or with rejection. A positive realizer of readiness for a finite set is a tuple of accepted finite witnesses, tagged by coordinates. These witnesses need not be computable from the context without communication.

Effectivity here is operational. We use a constructive interface for finite positive evidence inside the site. We make no identification of this $\infty$-topos with a realizability topos. A proof certificate is the finite representative data and the named check, rather than a computed presentation of all paths of the localized space.

**Proposition 2.4** (Finite completion after staggered arrivals)**.** *Let $C_t \subseteq R_\gamma$ be the coordinates for which a verifier has durably retained a valid witness by step $t$. Assume the accepted witnesses are recheckable and the context stays pinned. Then $C_t \subseteq C_{t+1}$. If a finite set $U \subseteq \bigcup_t C_t$, there is a finite $t_*$ with $U \subseteq C_{t_*}$. The same conclusion holds for a monotone predicate on coverage that has a finite positive witness, including a per-stripe threshold.*

*Proof.* For each $r \in U$, choose its first accepted step $t_r$. The maximum of the finitely many $t_r$ suffices. The cached tuple verifies the positive claim regardless of which peers remain online at that step. For the predicate choose a finite supporting subset $U$ and repeat the argument. If $U = \varnothing$, take $t_* = 0$. □

**Proposition 2.5** (No sound timeout for global absence)**.** *In an asynchronous model permitting an eligible shard to remain silent for any finite time, a client with no verified exclusion for that shard cannot in general certify that an unseen matching row is absent merely because a timeout elapsed.*

*Proof.* Take the prefix seen by the client through the timeout. In one continuation the missing shard has a matching row and is delivered later. In another continuation the committed bytes remain unavailable. The same prefix is consistent with delayed delivery of a match. An opaque authenticated digest is a binding commitment, not a nonmembership proof. The verifier must therefore retain a possibility of a match unless the prefix supplies independent evidence that rules it out. This reasoning applies to an empty partial result as well as to a candidate top-$K$ result. □

More formally, the unavailability of a preimage oracle is part of the black-box commitment interface in proposition 2.5. The proposition does not rule out deciding a particular absence from other evidence, such as a verified authenticated inverted list or a previously cached block. For future availability itself the indistinguishability is exact. Two executions with the same signed root and the same received prefix can have permanent withholding in one and delivery after the timeout in the other. A finite local transcript cannot certify which continuation will occur. The positive completion predicate is semidecidable by a fair traversal of valid delivery attempts. Under permanent withholding that traversal may run forever. A deadline can stop a poll or change a client policy; the underlying certificate state remains partial.

## 3 Authenticated PDF restoration

A snapshot is signed over deterministic bytes. Its manifest binds the archive and codec profiles, document identities, ordered fragment commitments, the page count and extraction profile, index roots and routing information. Hashes use type domains and length-delimited encodings. A representative fragment commitment is

$$d_i = \mathsf{H}(\mathsf{frame}(\texttt{fragment/v1}, \mathsf{docID}, \mathsf{stripe}, i, |y_i|, y_i)) .$$

The document digest binds the original byte sequence and length. A separate commitment binds its derived page records. The reference uses SHA-256 [4] and Ed25519 [5]. A production profile may use deterministic CBOR [6, Section 4.2] with an explicit schema; the reference fixes its own deterministic binary encoding.

Let $F = \mathbb{F}_{2^8}$, with the polynomial basis obtained from $X^8 + X^4 + X^3 + X^2 + 1$. Choose $n \le 255$ distinct nonzero elements $\alpha_i$. Let $V_{ij} = \alpha_i^j$ for $0 \le j < k$, and let $V_0$ be the first $k$ rows. Encoding each column of a padded stripe uses the systematic matrix

$$G = V V_0^{-1} \in F^{n \times k}.$$

Each set of $k$ rows of $G$ is invertible because the corresponding Vandermonde determinant is nonzero. This is the erasure-code property used in the Reed–Solomon profile [7]. Code parameters, field basis, slot order and padding rule are part of the signed profile. The reference uses this construction and its own framing, with $n \le 16$.

**Theorem 3.1** (Verified reconstruction and repair)**.** *Suppose an authorized manifest is pinned and its fragment commitments were made from a codeword of $G$. In a trace without a successful signature forgery or a digest substitution, any $k$ distinct valid slots suffice to recover a stripe uniquely. Verifying the document digest after concatenation and unpadding binds the output to the committed original bytes. A replacement slot obtained by re-encoding and checked against its commitment has the committed bytes.*

*Proof.* Authenticated framing binds a valid slot to its document, stripe and position. Any corrupt response rejected by the verifier is an erasure. For an accepted $k$-set $I$, the observed symbols are

$y_I = G_I x$ in each column. Since $G_I$ is invertible, the only candidate is $x = G_I^{-1} y_I$. The recorded length determines the unpadded byte sequence. The final digest check establishes the byte identity under the stated binding hypothesis. Re-encoding yields $G_i x$, and verification of the replacement commitment checks its identity even at a peer that did not retain the old bytes. □

If a malformed but signed manifest does not describe a codeword, decoders may reject after reconstruction. The verified document digest continues to govern whether a byte sequence can be returned. A malicious signer can choose a different archive to publish. The protocol's integrity result is relative to the archive actually pinned by the client. The existence of $k$ fragments is a statement about eventual delivery and storage health, not a consequence of cryptographic binding.

Production archives may choose fixed or content-defined chunking before erasure coding. Record the chunker and ordering and domain-separate deduplication by authorization boundary. A rebuilder collects and verifies slots, regenerates the replacement, fsyncs and atomically publishes it, and emits an auditable repair result. The storage plan should place distinct slots across actual administrative and failure domains. A content-addressed copy on another account in the same failed domain may give additional read throughput but contribute little to fault diversity.

# 4 Keyword, semantic and hybrid search

## 4.1 The indexed object

Store original PDF bytes under their own identity. A separate index row contains $(d, p, j)$, a page text or passage, extraction status, and model-specific features or sufficient data to recompute them. Here $d$ is the document digest, $p$ the physical page number starting at one, and $j$ a stable passage ordinal. Coordinates or offsets can be added when the extraction profile defines their reference frame. The row and a snippet are attributable to a known page even if retrieval of the original PDF needs additional contacts.

Digital text may be extracted directly. For image-only or unusable text layers, render in a resource-bounded worker and run OCR with the snapshot's versioned language packs and preprocessing. PDF structure and OCR tooling are described in [8, 9]. Record extraction status as one of text, OCR, verified blank, or failed; a permissive policy may also retain an OCR-empty state with its known uncertainty. A strict coverage policy treats failed or uncertain pages as explicit unresolved extraction obligations. A permissive policy records those pages and makes its completeness claim only about the indexed representation. Rotation, mixed scripts and reading order are part of the extraction policy. A new OCR or embedding model produces a new snapshot, even if every original PDF digest stays the same.

Keyword search can evaluate a precise predicate on normalized tokens or use BM25 [10]. For example, with global snapshot statistics, one choice is

$$\mathrm{BM25}(q, x) = \sum_{w \in q} \log\!\left(1 + \frac{N - \mathrm{df}(w) + 0.5}{\mathrm{df}(w) + 0.5}\right) \frac{f(w, x)(k_1 + 1)}{f(w, x) + k_1(1 - b + b|x|/\overline{|x|})}.$$

The set or multiset convention for query terms is fixed by the profile. Using local document frequencies or average lengths at each shard can change cross-shard comparisons. A manifest must bind the shared statistics or their verifiable derivation.

A semantic scorer may use a frozen encoder and cosine similarity, as in sentence-embedding retrieval [11]. Pin the tokenizer, model weights, pooling and numeric profile together with the embedding dimension. A practical exact profile computes fixed-point dot products and specifies rounding and accumulator bounds. Floating-point model execution can vary between implementations.

Deterministic inference or conformance checks with a fully specified quantization boundary are part of a portable scoring profile.

A hybrid score can use a fixed monotone fusion of comparable lexical and vector scores, with a deterministic tie order on $(d, p, j)$. Two other useful modes are literal token search and semantic search alone. Their answers can differ on the same corpus. Human relevance is evaluated against a judged query set, for example with Recall@$K$ and nDCG@$K$; a benchmark such as BEIR illustrates why model performance must be evaluated across domains [12]. A cryptographic proof of the exact ranking of an encoder says nothing by itself about its recall on a new language or corpus.

### 4.2 Merge and completeness

Fix a finite candidate set with stable keys and a strict total order $\prec_\gamma$ derived from scores and the tie order. All copies of a row under a key must agree, or the verifier rejects the inconsistent evidence. Let $\mathrm{Top}_K(X)$ select the $K$ greatest distinct candidates of $X$.

**Theorem 4.1** (Stable top-$K$ merge)**.** *For finite $X, Y$ and fixed $\prec_\gamma$,*

$$\mathrm{Top}_K(X \cup Y) = \mathrm{Top}_K\big(\mathrm{Top}_K(X) \cup \mathrm{Top}_K(Y)\big).$$

*On sets of at most $K$ candidates, $A \vee_K B = \mathrm{Top}_K(A \cup B)$ is commutative, associative and idempotent with identity $\varnothing$.*

*Proof.* If a candidate $x \in X$ is absent from $\mathrm{Top}_K(X)$, at least $K$ distinct members of $X$ outrank $x$. They are still present in $X \cup Y$, so $x$ cannot belong to $\mathrm{Top}_K(X \cup Y)$. The same argument applies to $Y$. Every global winner is therefore present in the trimmed union. Trimming cannot improve a nonwinner over a retained global winner. Repeated use of the equation proves associativity, and commutativity and idempotence follow from set union and the cardinality bound. □

For a job, merge both the top set and a map of verified coverage slots. The coverage operation is set union (or agreement-checked map union) and is monotone. Ranked prefixes may change when better candidates arrive. The combined merge is independent of delivery order because both components satisfy their merge laws. Copies of a signed answer claim from an untrusted worker do not show that it evaluated the whole shard. The full-block verifier reads the committed shard, checks its coverage and recomputes its scoring. A proof-carrying variant must authenticate each reported result and establish a sound upper bound in the same total order for every excluded row. If the result is full, the greatest omitted rank must be strictly below the $K$-th returned rank. With fewer than $K$ results, every uncovered eligible match must be excluded. Authenticated postings may serve as negative evidence for a defined lexical predicate. Approximate nearest-neighbor search can yield verified hits and empirical recall, but by itself does not provide the exclusion proof for exact vector search. Threshold algorithms [13] give relevant techniques when their access and monotonicity assumptions hold.

**Example 4.2** (Local rank truncation can discard the fused winner)**.** Reciprocal rank fusion combines global rankings using $\sum_m (c + \mathrm{rank}_m(x))^{-1}$ [14]. For two channels let their orderings be $(a, x, y, b)$ and $(b, x, y, a)$. At $c > 2$,

$$\frac{2}{c+2} > \frac{1}{c+1} + \frac{1}{c+4} \qquad \text{and} \qquad \frac{2}{c+2} > \frac{2}{c+3}.$$

The fused winner is $x$, which is absent from both channel top-one lists. The merge theorem applies after a common fused order has been defined. It does not allow each channel or shard to assign globally comparable rank positions from its own truncated data.

## 5 Delivery probability and latency

For each obligation define $T_r$ as the first time a valid witness is durably available to a job, with $T_r = \infty$ if it never arrives. Let $D$ be time to a verified directory or manifest if it is not already pinned locally. For full-block exact search and one-stripe recovery, respectively,

$$T_{\text{search}} = \max(D, \max_{r \in R_{\text{index}}} T_r), \qquad T_{\text{pdf}} = \max(D, T_{(k)}),$$

where $T_{(k)}$ is the $k$-th order statistic of the fragment arrival times. For multiple stripes, take the maximum of their threshold times. For an end-to-end request asking both for exact search and retrieval of a specified original, take the maximum of the required times. First-hit latency is a different random variable and may be infinite for queries with no matches. Time accounting includes verification and any waiting required to persist the witness.

If $n$ fragment delivery events are independent conditional on a network state $z$, and $p_i(t \mid z) = \Pr(T_i \le t \mid z)$, then

$$P_{\text{rec}}(t \mid z) = \sum_{\substack{I \subseteq [n] \\ |I| \ge k}} \prod_{i \in I} p_i(t \mid z) \prod_{j \notin I} (1 - p_j(t \mid z)).$$

Equal $p_i = p$ gives the binomial upper tail. A shard served by $r$ independent replicas with delivery probability $p$ has coverage probability $1 - (1-p)^r$. Independence must be justified at the event and failure-domain levels. Even if independence is plausible within each requirement, shared replicas or network links can couple different requirements. If $F_a(t)$ is the failure event for requirement $a$, arbitrary dependence still permits the bound

$$\Pr(T_{\text{finish}} \le t) \;\ge\; \max\left(0, 1 - \sum_a \Pr(F_a(t))\right).$$

The sum may include metadata, index shards and stripe thresholds. Cryptographic failure is a separate computational term. For an event asserting verified success with a correct output, one may subtract an upper bound $\epsilon_{\text{crypt}}$ for the forgery and substitution events in the workload. A confidence estimate for network delivery is not a cryptographic error rate.

Table 2: Illustrative cumulative valid-delivery probabilities. This is an analytic scenario. The values of $p$ and $d$ are inputs, not measurements.

| Horizon | $p$ | $d$ | One stripe | Exact search | Both | Bound for both |
|---|---|---|---|---|---|---|
| 2 s | .35 | .750 | .1174 | .0575 | .0068 | .0000 |
| 5 min | .65 | .910 | .6471 | .6409 | .4147 | .2141 |
| 1 day | .92 | .990 | .9915 | .9860 | .9776 | .9774 |
| 7 days | .98 | .998 | .9998 | .9979 | .9978 | .9978 |

In table 2, the PDF has one $(n, k) = (6, 4)$ stripe and the index has eight shards, each with three independently available replicas. The directory availability is $d$. The stripe column uses the binomial tail without the directory. The exact search column is $d[1 - (1-p)^3]^8$. The joint column assumes independence across the stripe, directory and shard requirements, while the rightmost column only needs their marginal failure probabilities. For example, if a shared outage of probability .20 prevents any completion by one day, and the one-day conditional model of the table holds outside the outage, the joint probability falls to approximately .7820. Placement across named machines cannot overcome a common dependency that cuts off every useful route. Operators should measure delivery curves per role and failure domain and publish the horizon and workload used for a percentile or service objective.

**Theorem 5.1** (Retry opportunities with dependent rounds)**.** *Let $0 = t_0 < t_1 < \cdots$ be retry horizons and $T$ the completion time, with retained progress and $\Pr(T > t_0) > 0$. Suppose that on every reachable history with $T > t_{j-1}$, the conditional probability of completing by $t_j$ is at least $a_j \in [0, 1]$. Then*

$$\Pr(T > t_m \mid T > t_0) \leq \prod_{j=1}^{m} (1 - a_j).$$

*If $\sum_j a_j = \infty$, completion is almost sure under these hypotheses.*

*Proof.* The noncompletion events are nested. Condition on the history through $t_{j-1}$ on the event $T > t_{j-1}$; the hypothesis gives a bound of $1 - a_j$ for the conditional chance of remaining incomplete. Taking conditional expectations and iterating proves the product bound. Since $1 - a \leq e^{-a}$, the product tends to zero under the divergence hypothesis. Continuity of probability for the decreasing events gives the almost-sure claim. □

The hypothesis concerns the chance of finishing the remaining job given the prior history, including persistent outage information. It is stronger than saying each round has a high marginal success probability. If a network has a permanently isolated class of clients, a uniform positive conditional lower bound cannot apply to its histories. Repair adds a feedback loop. Scrubbing can restore a previously damaged slot while $k$ other verified slots remain obtainable. It cannot bootstrap from fewer than $k$ distinct valid slots without a separate source of data.

# 6 The operational protocol

## 6.1 Bootstrapping, ingestion and jobs

An invitation carried by a side channel contains the archive identifier, publisher key or fingerprint, an authenticated manifest or locator, and one or more candidate peer routes. An inviter already in the network may pass the invitation to a new node. Trust derives from the pinned authority and the verification rules. Multiple signed locators, cached manifests and rendezvous relays permit progress through different network paths. An endpoint change is a signed directory update under its own policy. Fetching the next endpoint cannot silently alter the immutable obligations of an existing query. Mobile or intermittent hosts may make outbound connections to authorized relays, subject to the host's networking policy. An unreachable host cannot serve until a valid path becomes available.

An ingest authority computes original and derived content identities, forms the fragment and index layouts, and signs a snapshot only after checking profile consistency. For a production publish policy, make minimum independently verifiable placement receipts a separate obligation. A placement receipt is a statement made by an operator. A read challenge later establishes the ability to deliver particular bytes under the challenge timing model. Continued availability depends on placement, refresh and repair. A deployment can place cold archives on media slower than the index, but must make the difference visible to clients.

In the service protocol a request receives a durable identifier and may return partial results or a completion certificate. A caller can poll, use long polling with a bounded wall-clock budget, or resume later against its own job cache or a durable scheduler. Use backoff and jitter across peers. An idempotent request is keyed by an explicit caller nonce and an authenticated job specification. A content deduplication key alone is insufficient where policy or snapshot changes. Retain coverage only after verification and durable commit. Persist progress locally when possible, so later polls can finish without simultaneous connectivity to the first responders. If a cache is lost, revalidate and reconstruct the evidence state from surviving copies before asserting the lost coverage.

Table 3: Job transitions for a pinned context.

| Action | Guard | Effect |
|---|---|---|
| Open | Known authorized context | Persist query, evaluator, snapshot and initial empty evidence cache. |
| Poll | Open job or resumable wait | Attempt bounded fetches. Validate evidence before installing it. Report coverage and ranked prefix. |
| Finish | Full or soundly excluded search scope | Emit exact results and evidence references. An empty result has the same coverage requirement. |
| Suspend | Caller deadline or backpressure | Return a durable job identity and current verified coverage. Keep the job resumable. |
| Resume | Same authorized context and intact job | Recheck cached evidence, retry missing obligations and keep the original scorer and scope. |
| Restore | Document and codec pinned | Cache valid slots, reconstruct at the threshold and verify the original digest before writing output. |

The API distinguishes job status from transport status. A failed network call reports unavailability for that attempt. An incomplete query reports the known search scope, verified slots and the current candidate list. An exact answer certifies a specific snapshot. A freshness request is a new requirement for a sufficiently recent authorized checkpoint, and cannot be inferred from successful search under an older pinned root. A response may attach a PDF restoration job to each citation while preserving the independence of search and original recovery.

### 6.2 Audit and node qualification

Every service-capable peer must implement the profile's hashing and audit verification rules. SHA-256 known-answer tests, signature verification on valid and invalid messages, canonical decoding, and positive and negative audit fixtures are minimum admission checks. The peer demonstrates possession of its authorized Ed25519 identity key by signing a fresh, single-use challenge bound to the archive, role and snapshot or membership epoch. Hashing and signature verification must be available to all peer roles even when a hardware or remote key provider performs signing. A peer that cannot perform the required checks is ineligible to participate as a peer. A thin display client may consume independently verified results under a separate trust arrangement.

A signed checkpoint names an audit log, epoch, tree size and Merkle root. Verifying an event requires checking its typed payload and inclusion proof under an authenticated checkpoint. Verifying extension requires checking a consistency proof from a locally pinned checkpoint to the new checkpoint. This can use the tree construction of RFC 9162 [15]; the archive's event schema and witness policy are its own. An implementation should retain the last accepted checkpoint or its authenticated recovery evidence across restarts. The reference uses a smaller scheme that replays the full prefix of a signed hash chain rather than Merkle proofs. Its verification and communication cost grow with the log.

A malicious log signer can present individually valid divergent histories to isolated clients. Comparing signed incompatible heads exposes a fork. Comparing roots of different lengths requires prefix or consistency information; distinct root values at distinct lengths are normal. An optional witness policy constrains this behavior.

**Proposition 6.1** (Witness intersection)**.** *Fix a witness configuration with $N = 3f + 1$ witnesses, at most $f$ Byzantine. Require $2f + 1$ signatures on each accepted checkpoint. Each honest witness durably signs only extensions of its previous checkpoint for the same log and epoch, after verifying consistency. Two accepted checkpoints in that epoch cannot represent incompatible prefixes.*

*Proof.* Two signer sets of size $2f + 1$ intersect in at least $2(2f + 1) - (3f + 1) = f + 1$ witnesses. An honest witness belongs to the intersection. In its local signing order the later checkpoint extends the earlier one by its signing rule. This contradicts incompatibility, up to a successful cryptographic substitution. □

Quorum intersection is a safety argument. A partition can prevent $2f + 1$ witnesses from signing. Changing witness configurations requires a verified epoch transition and an explicit link between old and new checkpoint policies. In deployments with long periods of offline operation, gossiping retained signed heads may be preferable to requiring a live quorum for each read. This trades fork prevention for detection when views can be compared. A log proves what an actor committed to in the log. A separate challenge protocol is needed to test whether bytes remain retrievable, and its interpretation must account for proxies and colluding storage providers.

### 6.3 Scrubbing and repair

A scrub verifies stored bytes against the manifest. Choose its cadence from observed fault patterns and operational load. If $b$ of $B$ blocks are already corrupt at the time of an unpredictable simple random sample of $s$ distinct blocks, the probability of discovering at least one is

$$1 - \frac{\binom{B-b}{s}}{\binom{B}{s}},$$

taking the numerator as zero for $s > B - b$. The formula is obtained by counting all samples that avoid the $b$ corrupt blocks. It does not apply to an adaptive adversary that can learn the sample and move data before answering. A fixed sample set also cannot diagnose damage outside that set. Use a combination of randomized sampling and planned complete verification. On discovering a corrupt slot, record the failed check, source any $k$ valid slots, verify the reconstructed original or stripe, and publish a replacement with a signed repair outcome. LOCKSS provides a related design precedent for peer auditing through rate-limited sampled voting [16].

For crash recovery, a durable repair intent can bind the snapshot, slot, replacement digest and operation identity before the write. Completion records are committed after installation and are idempotent on the intent identity. On restart, reverify an installed replacement or resume reconstruction, then close the same intent. This sequence makes the gap between a filesystem change and an audit append explicit. The simpler reference repair command uses atomic file replacement for each object or log and can be re-run after interruption.

## 7 From specification to platform binaries

### 7.1 A trace specification with refinements

Define abstract commands

$$\mathsf{Sig} = \{\mathsf{open}, \mathsf{acceptIndex}, \mathsf{acceptSlot}, \mathsf{finish}, \mathsf{restore}, \mathsf{advanceAudit}, \mathsf{repair}, \mathsf{wait}\}.$$

For each context, a state includes a pinned manifest, a verified index map, a verified fragment map, output and job state, and audit pins. The transition for each command is guarded by the verifier and the preconditions already stated. The verification of a manifest, of an index block, and of a fragment are different operations. A restore step requires $k$ distinct slots, decoding, and a document digest check. Finish has the exact-search coverage guard. An audit-pin advance requires prefix extension. Outputs are typed by their context.

One presentation uses states as objects, finite command traces as 1-cells, and invertible 2-cells generated by observation-preserving swaps of independent steps and their coherence equations. Independent steps read compatible pinned state, use disjoint write coordinates and emit no ordered public events. A verified update to one shard and a verified update to a different shard can commute in this observation. Two consecutive updates to the same audit pin have a defined order. The nerve of the presentation yields a higher trace specification; localizing additionally marked silent steps gives a comparison interface. Projecting a trace to its validated per-coordinate witnesses gives data in $E_\gamma$. The projection sends a permitted swap to the specified identification of witness histories. Durations and raw physical side effects may be observed by a deployment monitor outside this quotient.

A platform interpretation of witnesses supplies a map $v_{P,r} \colon E_r^P \to E_r$ for each coordinate it serves. Here $E_r^P$ is formed from its concrete transcripts with the permitted equivalences, and $v_{P,r}$ erases platform detail and preserves verified assertions. For any supported $U$, the product of these maps commutes with restriction because both routes discard exactly the coordinates outside $U$. Thus different platform choices at different $r$ can supply compatible local sections of the same sheaf. Applying theorem 2.2 to the verified sections requires agreement of assertions and the specified coherence on overlaps. It requires no common machine architecture.

**Definition 7.1** (Platform interpretation and simulation)**.** A role implementation supplies a relation $L$ from its reachable concrete states to abstract states, relating their initial states. For each concrete transition from a related pair there is a finite abstract path, possibly empty, with the same visible observation and with related target states. Any response declaring verified data, exact search, recovery or audit advancement is visible. Network arrival and timeout can be modeled as shared input events.

**Theorem 7.2** (Compositional safety of role lowering)**.** *Forward simulations in the sense of definition 7.1 compose. A concrete implementation related by such a simulation to the abstract commands preserves all prefix safety claims of the abstract verifier. For example, it cannot emit an exact answer before meeting the declared coverage guard. This statement is conditional on correct adapter and VM behavior.*

*Proof.* Given relations $L_{21}$ and $L_{10}$, use relational composition. A concrete step is simulated by a finite intermediate path. Apply $L_{10}$ successively to its steps and concatenate the resulting abstract paths. The observations agree because they are compared under composition and silent observations are units. The endpoints remain related. Induction on a concrete finite trace proves preservation of any trace-prefix safety claim. A response disallowed by the abstract verifier would give a forbidden observation in the simulating trace. □

The theorem isolates what a new backend must show. To transfer liveness, it must also maintain progress under the stated opportunity hypotheses, including recovery from reboot and fairness for retry work. A simulation that allows indefinite silent stuttering does not establish that. A backend with a different model or a weaker audit policy uses a different profile and must publish the corresponding contract.

## 7.2 Portable payload and platform contract

The build has two stages. A profile-specific lowering supplies the portable verifier and role procedures. A platform interpretation supplies a VM, a launcher and capability adapters for storage, transport, identity keys, and optional extraction or inference. The resulting build family is indexed by platform $P$ and role $r$:

$$\mathsf{Build}(P, r) = \big(\mathsf{payload}_r, \mathsf{VM}_P, \mathsf{launcher}_P, \mathsf{adapters}_{P,r}, \mathsf{checks}_{P,r}\big).$$

A deployment may share the VM across its roles and select one role with a subcommand. Specializing the same observation contract to different admissible hosts gives binaries whose evidence can be checked by each other under one profile.

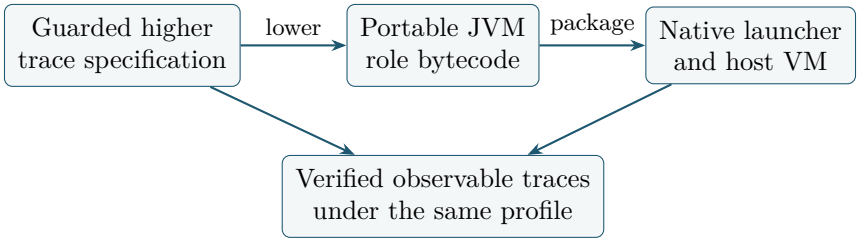



Figure 2: The simulation obligation relates the abstract verifier and a platform build after projecting away internal, silent operations.

For the reference profile a platform qualifies for a role when it can execute the JVM 17 payload [17], has a functioning crypto provider and secure entropy, and passes the role's audit and admission checks. It also needs the relevant durable I/O and usable network transport. A node that processes untrusted PDFs needs a resource-bounded extraction adapter. A storage node serving already committed blocks has no need to run OCR or an embedding model. An index builder that claims another model must bind its artifacts and meet its numerical conformance profile. The JVM and adapters can be compiled or packaged for each platform. The availability of a VM port by itself does not establish role conformance.

Table 4: Platform capabilities by role.

| Role | Shared admission floor | Additional capability |
|---|---|---|
| Object and index serving | VM, hash, identity proof and audit verification | Durable objects, peer endpoint or relay, bounded response serving |
| Query and restore worker | Same peer admission floor | Durable job cache, score evaluator and erasure decoder for selected profiles |
| Ingest and index builder | Same peer admission floor | PDF parser, OCR and declared language packs, model artifacts if required |
| Audit witness and repair | Same peer admission floor | Durable checkpoint pins, repair staging or witness signing policy |

The reference compiles Java sources to class files and bundles them as an executable JAR. A build script invokes `jpackage` for the selected platform, generating a native launcher and application runtime image [18]. Build on the target operating system and architecture, or use an appropriately qualified target build environment. A platform lacking a JVM host or equivalent verified interpreter, the necessary operating permissions, or the crypto and audit capabilities cannot join under this profile. The broader semantics permits a different virtual-machine backend when its adapters and observation relation are established.

The lowering here is the executable implementation of the specified profile plus a reproducible build pipeline. The simulation theorem is a proof obligation and a composition result. The conformance checks exercise that interface on a finite set of traces. There is no mechanized compiler-correctness proof for the Java runtime or the operating system.

# 8 Reference implementation and evaluation

## 8.1 Executable profile

The companion's Java 17 implementation uses a signed static manifest and $1 \le k \le n \le 16$, with a whole-file stripe for each PDF. The index has one block per peer slot. A row records a document digest, physical page, extraction method and extracted text. The assignment function

$$\mathsf{slot}(d, p) = \left(\mathsf{hex}_{0:4}(d) + p - 1\right) \bmod n$$

uses the first four hexadecimal characters of the document digest. The verifier checks the assigned slot, membership of each document, page bounds, absence of duplicate page keys, and the expected row count from the manifest. Because the possible keys for a block are already finite and known, these checks certify that each expected page key appears exactly once. The manifest is ordered by document digest. The ingester deduplicates identical PDFs.

Poppler provides page counting and text extraction. Text-empty pages are rasterized and passed to Tesseract with the English model. The profile records an empty OCR outcome explicitly. The archived object is always the original byte sequence. Ingest has input, page and process-time limits. The tests use trusted fixtures. Deployments that ingest hostile files need an isolation boundary with hard resource limits for the native tools.

For a reproducible semantic baseline, each row and query is mapped to a bit vector of ten named English concepts. Tokenized synonyms such as *deterioration*, *corruption* and *bitrot* activate the same coordinate. The semantic score is

$$\left\lfloor 10^6 \frac{\operatorname{popcount}(u \,\&\, v)}{\sqrt{\operatorname{popcount}(u)\operatorname{popcount}(v)}} \right\rfloor$$

for nonempty overlapping vectors, and zero otherwise. The code computes it using integer arithmetic and an integer square root. Keyword mode accepts pages containing all query tokens after Unicode NFKC normalization, root-locale lowercasing and tokenization into letter, mark and number runs. Hybrid mode adds $10^6$ for such a lexical match to the semantic score. Positive scores are ordered decreasingly, then by document digest and page ascending. The lexical match in this profile is token-level, without phrase or stemming semantics. Unknown concepts yield zero in semantic mode. A production embedding can replace the ontology by defining a new verified evaluator. The prototype evaluates a deliberately small English vocabulary and does not report corpus-level retrieval quality.

A client copies the verified signed manifest and fingerprint into a job. It writes each verified index block into a content-addressed cache and checks cached data on every poll. Until every expected block verifies, it displays `PARTIAL`. An exact job can replay after all peers go offline. Fragment retrieval uses a separate cache and can span nonoverlapping availability windows. Recovery returns a retryable waiting status below threshold. Audit checking uses signed event replay with a durable verifier cursor, rejecting both older views and inconsistent extensions of a pinned head.

In the staggered test, only peer 0 is initially reachable. An exact job over all peers reports an early semantic match for *deterioration*, which appears in none of the matching pages. After peer 0 stops, peer 1 provides a second distinct fragment. The original PDF is recovered from the two temporally separated slots. Peer 2 then supplies the final index block in a separate window. The search moves to complete and rejects the unrelated railway PDF. The image-only fixture contains the words *Scanned Garden*; its keyword hit carries an OCR method marker. A separate query for the literal token *deterioration* completes with no hits.

The experiments ran on Linux x86-64 with OpenJDK 17.0.20, Poppler 24.02.0 and Tesseract 5.3.4. The native launcher was built using `jpackage` with a Linux OpenJDK runtime image. The JAR,

Table 5: Reference checks in the reproducibility companion.

| Check | Observed behavior or test scope |
|---|---|
| Verification profile | SHA-256 and Ed25519 known-answer checks, fresh possession challenge, negative signed-audit test and wrong publisher fingerprint rejection. |
| Staggered index delivery | Three separate peer windows grow coverage from 0/3 to 3/3. A first hit appears while the job is partial. An offline replay reproduces the completed result. |
| PDF search | Concept-only matches across a three-page text PDF, a keyword hit in an image-only PDF after OCR, and a deterministic hybrid result with page attribution. |
| Index and fragment faults | Altered index bytes do not count toward coverage. A corrupted fragment is treated as an erasure, and two valid slots restore a byte-identical PDF. |
| Repair and audit | Repair regenerates the committed slot and appends a verified event. A changed signature, older signed prefix, and incompatible same-key branch against a pin are rejected. |
| Erasure matrix tests | 2,825 successful subset round trips across five $(k, n)$ profiles and five input lengths. |
| Platform build | JVM bytecode and a Linux x86-64 native application launcher both exercise the peer fixture. |

source and launcher build command are included in the companion. The application image is generated on demand. The tests use separate processes on a single host, and the timed probability examples in table 2 are analytic scenarios. There are no observed multi-day delivery rates or comparative embedding benchmarks in this evaluation.

### 8.2 Operational boundaries

The reference stores index blocks in plaintext and is intended for authorized public corpus fixtures. A private archive must bind the policy and key epoch to each job, authorize access before disclosing text or embeddings, and isolate any client-side or trusted search index from unauthorized storage operators. Index vectors and cached snippets can disclose information about the corpus. Byte encryption alone does not make a separately exposed index private. Confidential retrieval is a different verifier and threat model.

The reference uses one committed index block per peer, manually invoked polls and repair, and a trusted local filesystem. It has no distributed epoch consensus or automatic placement scheduler. Its object and audit updates are separately atomic on filesystems supporting the requested rename operation, and the test evidence covers process restarts rather than arbitrary power-loss behavior. A deployment targeting stronger crash guarantees must durably commit both job state and parent-directory changes, reconcile repair intents, and qualify the I/O adapter. The signed HTTPS endpoint profile supports a TLS reverse proxy at each peer. That path is specified in the companion, while the executable fixture uses loopback.

## 9 Related systems and conclusion

OceanStore provides an architectural precedent for persistent wide-area storage with untrusted infrastructure [19]. LOCKSS addresses audit and repair of replicated archives through rate-limited peer voting [16]. Kademlia shows how a structured overlay can discover candidate routes [20]. Route discovery is replaceable here, since responses are verified against a signed context. The query side builds on lexical and embedding retrieval and on the monotone aggregation ideas of [10, 11, 13]. Certificate Transparency supplies append-only proof constructions [15]. The present construction

combines these techniques through a site of finite verification obligations with explicit evidence for completion. Its $\infty$-sheaf structure describes how valid local claims and their history equivalences glue; it does not model the stochastic dynamics that determine whether or when they arrive.

The practical consequence is an archive interface in which delayed answers remain usable and verifiable. The same job distinguishes incomplete evidence, exact search under a pinned scorer, and recovery of original bytes. A qualified platform implements the same verification rules and can contribute according to its storage, compute and network capabilities. The probability of completion follows from the actual delivery model, the placement of independent copies, and the continued availability of a valid route.

**Authorship and reproducibility.** The responsible human contributor uses an anonymous byline and authorized this submission. An AI assistant assisted in drafting the paper, producing the reference code and running the checks. The theorems are presented with paper proofs. The reproducibility companion contains all Java sources, a built JAR, packaging scripts, the synthetic test generator and recorded verification output. The work uses synthetic PDFs and newly generated fixture keys.

# A Binary framing and service interface

The reference encodes bounded nonnegative integers as four-byte big-endian signed Java integers restricted to their stated ranges. A blob is its length followed by bytes. A string is a blob of strict UTF-8. Encoders emit fields in the order listed below. Parsers check field bounds and reject trailing bytes. Let typed$(t, b)$ be the concatenation of a string containing $t$ and a blob containing $b$. The identifier of a payload under type $t$ is the lower-case hexadecimal SHA-256 of typed(`semarchive/v1/`$t, b$). A signature is Ed25519 over typed(`semarchive/v1/sign/`$t, b$). The signed envelope is a blob containing $b$, followed by a blob containing its 64-byte signature. Public keys use DER SubjectPublicKeyInfo. The publisher fingerprint is an identifier under `publisher-key`.

Table 6: Reference record layouts in encoded order.

| Record | Fields |
|---|---|
| Snapshot body | Profile string, archive nonce as 64 hex characters, $k$, $n$, $n$ endpoint and public key pairs, document count, sorted document records, $n$ index IDs in slot order. |
| Document record | PDF ID, display label, original byte length, page count, $n$ fragment IDs in slot order. |
| Fragment payload | PDF ID, slot number from zero, encoded data blob. |
| Index payload | Index slot from zero, row count, then rows of PDF ID, page number from one, extraction method, text. |
| Audit event | Snapshot ID, peer slot, sequence number from one, previous signed event ID (all-zero genesis), action, subject ID. The log is the entry count and a length-delimited sequence of signed envelopes. |
| Peer challenge | Snapshot ID, peer slot, 64-character hexadecimal nonce, signed under the `hello` type. |

The role binary exposes `build`, `build-net`, `serve`, `new-job`, `poll`, `restore`, `repair`, `probe`, `audit`, `inspect` and `self-test`. The HTTP peer exposes `GET /obj/{id}`, `GET /hello/{nonce}` and `GET /audit`. Content fetched over any path must pass the same application verifier. The companion documents command arguments and specifies the executable bounds.

## References


[1] W. G. Dwyer and D. M. Kan. Calculating simplicial localizations. *Journal of Pure and Applied Algebra* 18(1), 17–35, 1980. Author paper.

[2] C. Barwick and D. M. Kan. Relative categories: Another model for the homotopy theory of homotopy theories. *Indagationes Mathematicae* 23(1–2), 42–68, 2012. arXiv:1011.1691.

[3] J. Lurie. *Higher Topos Theory*. Annals of Mathematics Studies 170, Princeton University Press, 2009. arXiv:math/0608040.

[4] National Institute of Standards and Technology. *Secure Hash Standard (SHS)*. FIPS PUB 180-4, 2015. NIST publication.

[5] S. Josefsson and I. Liusvaara. *Edwards-Curve Digital Signature Algorithm (EdDSA)*. RFC 8032, 2017. RFC 8032.

[6] C. Bormann and P. Hoffman. *Concise Binary Object Representation (CBOR)*. RFC 8949, 2020. RFC 8949.

[7] J. Lacan, V. Roca, J. Peltotalo, and S. Peltotalo. *Reed-Solomon Forward Error Correction (FEC) Schemes*. RFC 5510, 2009. RFC 5510.

[8] International Organization for Standardization. *Document management, Portable document format, Part 2, PDF 2.0*. ISO 32000-2:2020. PDF Association edition and errata.

[9] Tesseract project. *Command Line Usage*. Tesseract documentation. Documentation.

[10] S. Robertson and H. Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval* 3(4), 333–389, 2009. DOI.

[11] N. Reimers and I. Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. *EMNLP-IJCNLP*, 3982–3992, 2019. ACL Anthology.

[12] N. Thakur, N. Reimers, A. Rücklé, A. Srivastava, and I. Gurevych. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. *NeurIPS Datasets and Benchmarks*, 2021. arXiv:2104.08663.

[13] R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. *Journal of Computer and System Sciences* 66(4), 614–656, 2003. arXiv:cs/0204046.

[14] G. V. Cormack, C. L. A. Clarke, and S. Büttcher. Reciprocal rank fusion outperforms Condorcet and individual rank learning methods. *SIGIR*, 758–759, 2009. Author paper.

[15] B. Laurie, E. Messeri, and R. Stradling. *Certificate Transparency Version 2.0*. RFC 9162, 2021. RFC 9162.

[16] P. Maniatis, M. Roussopoulos, T. J. Giuli, D. S. H. Rosenthal, M. Baker, and Y. Muliadi. Preserving peer replicas by rate-limited sampled voting in LOCKSS. *SOSP*, 2003. Author publication record.

[17] T. Lindholm, F. Yellin, G. Bracha, A. Buckley, and D. Smith. *The Java Virtual Machine Specification, Java SE 17 Edition*. Oracle, 2021. Specification.

[18] Oracle. *Packaging Overview*. Java SE 17 Packaging Tool User's Guide. Documentation.

[19] J. Kubiatowicz et al. OceanStore: An architecture for global-scale persistent storage. *ASPLOS*, 190–201, 2000. Author paper.

[20] P. Maymounkov and D. Mazières. Kademlia: A peer-to-peer information system based on the XOR metric. *IPTPS*, 53–65, 2002. Author paper.