isar innovations
Research Paper

Beam
Keyed Polarization for Search and Disclosure
from One Protected State

Search what you can open.

Author
Nino Wagensonner
Isar Innovations
info@isar-innovations.dev

June 2026

Abstract

Modern data products need search over protected records, but conventional systems often split the problem: vector search operates on exposed or separately protected embeddings, while payload encryption is applied after retrieval. This creates a product gap: a record can be found because of a field the user is not allowed to read.

Beam is a view-scoped architecture for protected search and selective disclosure from one compact stored state. Its central invariant is simple: *search what you can open.* Conceptually, Beam is a two-layer protected system: a payload-at-rest disclosure layer and a KPT-protected search-at-rest layer. ANN structures such as IVF/HNSW are acceleration infrastructure, not an additional cryptographic layer. The research contribution is the system-level use of a shared view projection as the binding contract between selective disclosure and searchable visibility. Grant/KMS handling and ANN indexing are engineering layers around that contract.

The Beam disclosure layer stores field buckets in one protected record and opens only authorized views. A bucket-local keyed permutation plus invertible dense block mixing preserves exact insert/open semantics while making authorized opens proportional to the selected view rather than the full record. Prototype evidence shows compact storage for a 200 KiB payload, rejects dense offline projection keys as a product path, and confirms the read-locality scaling law on streaming workloads.

1 Introduction

Encrypted application data often needs three capabilities at once: search should be fast enough for product use; search should only find data the user is allowed to read; and the stored record should not grow with every user, policy, or view. The failure mode is common in protected vector systems: a global embedding can produce a hit because of hidden fields, even if those fields are redacted after retrieval.

Beam addresses this as a systems-security invariant:

$$\mathsf{View}(k) = \mathsf{Scope}(k) = \mathsf{Index}(k) = \text{authorization boundary of } k.$$

Equivalently, for every key or session k and field f:

$$\mathsf{CanSearch}(k, f) \iff \mathsf{CanOpen}(k, f).$$

Beam builds on KPT-1 [8] for key-gated semantic retrieval, but moves the product boundary from record-level search toward view-level search and disclosure. The contribution is not a new ANN method, not a replacement for AEAD, and not a claim of malicious-provider privacy. It is a buildable architecture in which payload opening, search representation, and ANN index scope are all bound to the same view predicate.

Contributions. This paper makes four claims:

1. A view-scoped search/disclosure invariant: a key can search only what it can open.
2. A compact Beam record/open mechanism based on sealed field buckets, finite-field slots, and bucket-local invertible transforms.
3. A scope-binding contract for KPT and ANN acceleration: Beam view V, KPT search scope S_V, and ANN index scope I_V must coincide.
4. Systems evidence for compact storage, dense-key falsification, managed grants, and bucket-local readout scaling.

2 Threat Model and Boundaries

Beam v1 targets an honest-but-operational provider model: the deployed provider follows the Beam/KMS protocol, but database snapshots, object storage, backups, and index artifacts may leak. The attacker may see protected Beam records, protected KPT search/index material, nonces, layout metadata, and declared runtime access-pattern leakage. The attacker does not possess user keys, Authority/KMS secrets, or active runtime grant execution.

The trusted runtime Authority/KMS may issue, rotate, revoke, and verify grants, and may open authorized plaintext during a permitted runtime operation. Beam v1 does not claim protection against a provider that modifies runtime code, logs user/session keys, captures plaintext during authorized opens, or exfiltrates Authority/KMS secrets. Beam v1 also does not claim ORAM-style access-pattern hiding.

Payload confidentiality is carried by AEAD-sealed field buckets and the KMS boundary. Search-at-rest confidentiality is delegated to the KPT layer and evaluated separately. The Beam transform provides the algebraic stored-state/readout structure and locality needed for compact selective opening; it is not presented as a standalone confidentiality primitive.

3 Architecture

Beam has two protected layers and one acceleration layer:

Beam disclosure	payload-at-rest and selective disclosure,
KPT search	search-at-rest for semantic/vector search signals,
IVF/HNSW	acceleration only, inside authorized scopes.

On ingest, a record is encoded into a Beam payload envelope, KPT search representations are derived for authorized scopes, and ANN indexes are built only over those scoped representations. On query, a user key selects an authorized scope; IVF/HNSW returns candidate record identifiers from that scope only; Beam opens only the corresponding authorized payload view.

The architecture forbids a global all-fields embedding followed by payload redaction. Such a construction can return a record because of fields that the user cannot open. In Beam, search must factor through the same projection as disclosure:

$$\mathsf{SearchScore}(k, q, x) = \mathsf{score}\big(q, E(P_{\mathsf{View}(k)}x)\big),$$

not $\mathsf{score}(q, E(x))$ when k cannot open all of x.

4 Worked Example: Payload, Schema, and Views

Consider a customer-support record with five product fields:

$$f_1 = \mathsf{profile}, \quad f_2 = \mathsf{support\_notes}, \quad f_3 = \mathsf{billing},$$
$$f_4 = \mathsf{risk\_memo}, \quad f_5 = \mathsf{internal\_tags}.$$

In the application, the plaintext object might look as follows:

Field	Example plaintext value	Product role
`profile`	`name=Ada; plan=Pro`	shared
`support_notes`	`User reports sync error alpha-17`	support
`billing`	`Invoice overdue; VAT id ...`	finance
`risk_memo`	`Fraud review keyword: garnet`	restricted
`internal_tags`	`vip, manual-review`	internal

Example payload. Plain application payload before Beam encoding. Beam does not store this table as database plaintext.

The layout schema pins those product fields to stable positions and bucket sizes:

Position	Product field	Bucket	Support view	Finance view
f_1	`profile`	m_1	yes	yes
f_2	`support_notes`	m_2	yes	no
f_3	`billing`	m_3	no	yes
f_4	`risk_memo`	m_4	no	no
f_5	`internal_tags`	m_5	no	no

Example layout. Field names live in layout/authority metadata; the protected record stores mixed buckets, not plaintext columns.

The database does not store this object as plaintext fields. It stores one Beam envelope:

scheme	beam-v1
layout	customer_support_v3
field_count	5
bucket_sizes	$(m_1, m_2, m_3, m_4, m_5, m_{\mathsf{mask}})$
ρ	record nonce
y	$y_1 \| y_2 \| y_3 \| y_4 \| y_5 \| y_{\mathsf{mask}}$
tag	envelope authentication tag

Product field names are layout metadata known to the Authority/KMS and application, not plaintext labels inside the protected payload vector. Each y_f is the mixed form of a sealed field bucket.

Now define two views:

$$V_{\mathsf{support}} = \{\mathsf{profile}, \mathsf{support\_notes}\}, \qquad V_{\mathsf{finance}} = \{\mathsf{profile}, \mathsf{billing}\}.$$

A support key opens:

$$P_{V_{\mathsf{support}}} x = x_{\mathsf{profile}} \| x_{\mathsf{support\_notes}},$$

and cannot open x_{billing} or $x_{\mathsf{risk\_memo}}$. A finance key opens:

$$P_{V_{\mathsf{finance}}} x = x_{\mathsf{profile}} \| x_{\mathsf{billing}},$$

and cannot open the support notes or risk memo. Both keys address the same stored envelope y; no per-user payload copy is stored.

The search consequence is the point of Beam. A support query for *garnet* must not return this record if *garnet* appears only in risk_memo, because

$$E(P_{V_{\mathsf{support}}} x)$$

contains no contribution from the risk bucket. A finance query for *invoice* may return the record, because $\mathsf{billing} \in V_{\mathsf{finance}}$. Redaction after retrieval is too late; Beam requires the retrieval representation itself to be view-scoped.

5 Beam Payload Layer

Each payload field is encoded into a fixed-size sealed bucket. At the abstract level, a record is:

$$x = x_1 \| x_2 \| \cdots \| x_n \| x_{\mathsf{mask}},$$

where $x_f \in \mathbb{F}_p^{m_f}$ is the finite-field slot vector for field bucket f. The stored state is produced by a record transform:

$$y = R_{\mathsf{record}} x.$$

The current practical transform is bucket-local:

$$R_{\mathsf{record}} = \mathrm{blockdiag}(R_1, \ldots, R_n, R_{\mathsf{mask}}), \qquad R_f = M_f \Pi_f.$$

Here Π_f is a key-/nonce-derived permutation over bucket slots and M_f is an invertible dense block mix over $\mathbb{F}_p$. This design replaces global dense inversion and global slot scattering with local work. Opening field f reads y_f and applies R_f^{-1}; it does not touch the full record.

The stored envelope contains version/layout metadata, bucket sizing, prime, nonce, authentication tag, and the mixed vector y. It does not store plaintext values, field keys, raw inverse matrices, raw projection rows, or product field names.

6 Rebuildable Mechanism Specification

This section gives a reference construction precise enough to rebuild the mechanism. It intentionally leaves ordinary engineering choices such as the concrete AEAD library or KMS vendor open, but fixes the data flow, algebraic objects, and authorization boundary.

6.1 Record Encoding

Each record has a public scheme version, layout identifier, field count n, per-field bucket sizes m_f, mask bucket size m_{mask}, finite-field prime p, record nonce ρ, authentication tag, and mixed vector y. Product field names are not stored in the protected record; the Authority/KMS maps product names to stable field positions in the layout.

Parameter	Reference value or rule
Finite field	$\mathbb{F}_p$ with public prime p; prototype uses $p = 2^{61} - 1$
Slot encoding	7-byte chunks mapped into field elements; stored as 8-byte big-endian integers
Plaintext frame	4-byte big-endian length ‖ canonical bytes ‖ padding
Sealed bucket frame	4-byte big-endian length ‖ AEAD nonce/ciphertext/tag ‖ slot padding
Associated data	$(\mathsf{scheme}, \mathsf{layout}, f, \rho)$ plus view/policy id where used
Bucket size	fixed by layout; under-capacity and overflow are rejected

Table 1: Reference encoding contract. The exact AEAD may vary, but framing, layout binding, and field-position binding are part of the mechanism.

For field f, canonical payload bytes are padded and AEAD-sealed into a byte string c_f. The slotization map

$$\mathsf{Slotize}_{m_f}(c_f) = x_f \in \mathbb{F}_p^{m_f}$$

splits the framed sealed bytes into fixed-size chunks and pads remaining slots with zero. Decoding rejects malformed lengths, overflows, and slot values outside the field.

6.2 Transform Derivation

For every field bucket f, derive:

$$\Pi_f = \mathsf{DerivePermutation}(s, \rho, \mathsf{layout}, f, m_f), \qquad M_f = \mathsf{DeriveBlockMix}(s, \mathsf{layout}, f, b),$$

where s is Authority/KMS secret material and b is the local dense-block width. Π_f must be a bijection over $\{0, \ldots, m_f - 1\}$. M_f is block-diagonal with invertible blocks over $\mathbb{F}_p$; the final block is handled explicitly when m_f is not divisible by b. Implementations may use rejection sampling from a domain-separated KDF stream until an invertible block is obtained. Failure after the configured attempt bound is an insert/open failure, not a fallback to an unverified transform.

6.3 Insert

1. Sample a fresh record nonce ρ.
2. For each field position f, canonicalize the payload value, length-frame it, pad it, AEAD-seal it under associated data $\mathsf{AD} = (\mathsf{scheme}, \mathsf{layout}, f, \rho)$, and slotize it into $x_f \in \mathbb{F}_p^{m_f}$.
3. Derive Π_f and M_f, set $R_f = M_f \Pi_f$, and compute $y_f = R_f x_f$.
4. Sample or encode mask bucket slots x_{mask}, derive R_{mask}, and compute y_{mask}.
5. Store the envelope containing version/layout metadata, bucket sizing, p, ρ, authentication tag, and $y = y_1 \| \cdots \| y_n \| y_{\mathsf{mask}}$.

6.4 Open

Given an envelope, a requested view V, and a valid grant/session for V:

1. Verify envelope authentication and grant/session scope, TTL, epoch, audience, replay, and revocation state.
2. For each field $f \in V$, read only y_f from the mixed vector.
3. Derive or materialize $R_f^{-1} = \Pi_f^{-1} M_f^{-1}$ inside the authorized boundary.
4. Compute $x_f = R_f^{-1} y_f$, decode the sealed bucket, and AEAD-open with the same associated data.
5. Return exactly the fields in V.

No public grant exposes raw field keys, raw R_f^{-1}, raw projection rows, or dense offline key material. Managed grants are the product path because dense offline projection keys are intentionally rejected by the evaluation.

6.5 Scoped Search and Indexing

For each supported view or policy scope V, define:

$$S_V = \{E(P_V x_r) : r \in \mathcal{R}\}, \qquad I_V = \mathsf{ANNBuild}(S_V).$$

Queries under key/session k may evaluate only $I_{\mathsf{View}(k)}$. A full-record representation $E(x)$ followed by payload redaction is invalid for restricted keys. Unsupported dynamic views must either be built from field-level representations that already factor through P_V, trigger a scoped rebuild, or be declared unsupported by the product.

7 Algebraic Correctness

This section states the core proof. Detailed engineering choices such as AEAD selection, KDF labels, and grant token encoding are outside the algebraic argument.

7.1 Notation and Dimensions

Let m_f be the slot count of field bucket f and let

$$M = m_{\mathsf{mask}} + \sum_{f=1}^{n} m_f.$$

The unprotected slot state is $x \in \mathbb{F}_p^M$. The record transform is

$$R_{\mathsf{record}} \in GL_M(\mathbb{F}_p),$$

and the stored state is $y = R_{\mathsf{record}} x$. For a view V, let

$$M_V = \sum_{f \in V} m_f.$$

Then $P_V \in \mathbb{F}_p^{M_V \times M}$ selects exactly the authorized payload coordinates and drops every hidden or mask coordinate. Consequently,

$$K_V = P_V R_{\mathsf{record}}^{-1} \in \mathbb{F}_p^{M_V \times M}.$$

Definition 1 (View projection)**.** *For a view $V \subseteq \{1, \ldots, n\}$, let P_V be the projection that selects exactly the original buckets authorized by V and zeros all other payload and mask buckets.*

Definition 2 (View key)**.** *For stored state $y = R_{\mathsf{record}} x$, define the view key/operator*

$$K_V = P_V R_{\mathsf{record}}^{-1}.$$

Lemma 1 (Bucket transform invertibility)**.** *If Π_f is a permutation matrix and M_f is block-diagonal with invertible blocks over $\mathbb{F}_p$, then $R_f = M_f \Pi_f$ is invertible and*

$$R_f^{-1} = \Pi_f^{-1} M_f^{-1}.$$

Proof. A permutation matrix is invertible. A block-diagonal matrix with invertible blocks is invertible, with inverse obtained by inverting each block. The product of invertible matrices is invertible, giving $R_f^{-1} = \Pi_f^{-1} M_f^{-1}$. □

Theorem 1 (Authorized Beam collapse)**.** *For every record state x and view V,*

$$K_V y = P_V x.$$

Proof. By storage definition, $y = R_{\mathsf{record}} x$. By view-key definition, $K_V = P_V R_{\mathsf{record}}^{-1}$. Therefore

$$K_V y = P_V R_{\mathsf{record}}^{-1} y = P_V R_{\mathsf{record}}^{-1} R_{\mathsf{record}} x = P_V x.$$

□

Corollary 1 (Hidden fields contribute zero)**.** *If field $h \notin V$, then the slots of h are zeroed by P_V. Therefore hidden fields have zero contribution to $K_V y$. If two payload states x and x' agree on all fields in V but differ on hidden fields, then*

$$K_V R_{\mathsf{record}} x = K_V R_{\mathsf{record}} x'.$$

Theorem 2 (Bucket-local readout)**.** *Opening view V requires only buckets $f \in V$ and returns the same result as the global projection $P_V R_{\mathsf{record}}^{-1} y$.*

Proof. Since R_{record} is block-diagonal,

$$R_{\mathsf{record}}^{-1} y = R_1^{-1} y_1 \| \cdots \| R_n^{-1} y_n \| R_{\mathsf{mask}}^{-1} y_{\mathsf{mask}}.$$

The projection P_V selects only buckets $f \in V$ and discards all other terms. Thus computing $R_f^{-1} y_f$ for $f \in V$ is equivalent to applying the corresponding rows of the global operator. □

Theorem 3 (Read-locality bound)**.** *Let m_f be the slot count of field bucket f. If applying a fixed-width block mix costs linear time in the number of slots, then*

$$\mathsf{OpenCost}(f) = O(m_f), \qquad \mathsf{OpenCost}(V) = O\left(\sum_{f \in V} m_f\right).$$

Theorem 4 (Collusion gives union of views)**.** *If users hold $K_A = P_A R_{\mathsf{record}}^{-1}$ and $K_B = P_B R_{\mathsf{record}}^{-1}$, their combined outputs reveal at most the union projection $P_{A \cup B} x$.*

Proof. By authorized collapse, $K_A y = P_A x$ and $K_B y = P_B x$. The coordinates selected by P_A and P_B are subsets of those selected by $P_{A \cup B}$. Conversely, $P_{A \cup B} x$ is the union of the coordinates selected by $P_A x$ and $P_B x$. Thus combining issued view outputs yields the union of issued views, not fields outside the union. This theorem assumes colluding parties possess only issued view outputs/operators, not Authority secrets, raw field keys, or unrestricted inverse material. □

Corollary 2 (Search/readout invariant)**.** *If the search representation is defined as $E(P_{\mathsf{View}(k)} x)$, then hidden fields cannot be the causal source of a search hit under key/session k.*

Proof. For any hidden field $h \notin \mathsf{View}(k)$, the hidden-field corollary gives zero contribution to $P_{\mathsf{View}(k)} x$. Since the search representation is a function only of $P_{\mathsf{View}(k)} x$, changing hidden fields while keeping visible fields fixed leaves the input to E unchanged. Therefore any score computed from $E(P_{\mathsf{View}(k)} x)$ cannot depend on hidden fields. □

8 Search Scope and ANN Acceleration

Beam requires the search representation for a key/session k to be derived from the same projection as payload disclosure:

$$\mathsf{SearchRep}(k, \mathsf{record}) = E(P_{\mathsf{View}(k)}x).$$

If field f is hidden from $\mathsf{View}(k)$, the hidden-field corollary gives zero contribution to $P_{\mathsf{View}(k)}x$. Therefore hidden fields cannot be the causal reason for a search hit under that key.

IVF and HNSW are established ANN mechanisms [6, 7]. Beam does not claim to invent them. The rule is only that ANN acceleration occurs after authorization has selected the scoped corpus representation S_V. The index I_V is built over S_V and returns candidate identifiers from S_V only.

8.1 KPT-Beam Collapse Intuition

KPT-1 readers can think of the Beam adjustment as a two-stage collapse. In KPT-1, the full record contributes to the protected search signal; the key decides whether the phase geometry becomes coherent. A wrong key yields garbage or near-zero score, while the right key yields a coherent signal.

Beam adds a view collapse before that KPT key collapse:

$$x = x_a \| x_b \| x_c \| x_d \| x_{\mathsf{mask}}, \qquad P_V x = x_b \| x_c.$$

KPT must then run over $E(P_V x)$, not over $E(x)$. Thus hidden fields are not merely searched with the wrong key; they are absent from the search state for that key.

Case	Beam stage	KPT stage
Visible, right key	field passes P_V	coherent signal
Visible, wrong key	field passes P_V	garbage or near-zero score
Hidden-only	field removed by P_V	no search input for that term
Visible+hidden	visible passes; hidden is zero	signal only from visible part

Table 2: Beam adds a view collapse before KPT's key collapse. Hidden fields have zero contribution before KPT scoring begins.

9 Evaluation Evidence

The prototype evidence is intended to validate scaling laws and product feasibility, not production latency. The Python implementation does not yet claim a final cloud latency budget; native/SIMD implementation and real KMS/HSM round-trip measurements remain engineering follow-up.

Scenario	Meaning	Result
200 KiB payload	Stored Beam record	235304 bytes
200 KiB payload	Dense offline projection key	68866522240 bytes
20 views	Managed grant model	about 10240 bytes

Table 3: Storage and key-material evidence. Dense offline keys are intentionally rejected as the product path.

Scenario	Bucket-local read p95	Global read p95
20 KiB, 100 fields, open 1	0.648 ms	2.404 ms
200 KiB, 100 fields, open 1	4.274 ms	16.923 ms
2 MiB, 100 fields, open 1	42.563 ms	178.949 ms

Table 4: Read-locality evidence. The result supports the scaling law, not a final production latency claim.

A 1 GiB streaming run processed 5243 records of 200 KiB payload each, with 100 fields per record and one field opened per view. The run measured write p50/p95/p99 of 73.003/75.112/79.406 ms, read p50/p95/p99 of 4.285/4.480/4.617 ms, grant issue p95 of 3.783 ms, storage overhead of 1.165x, and 2384 bytes touched per one-field open.

Mixed permission search regression. The most direct test of the Beam invariant is a mixed permission search case: a term that appears only in a hidden field must not cause a hit, while the same term in a visible field must cause a hit under the same key and query configuration. The current testbed executes this regression before ANN acceleration; the same test should be repeated after IVF/HNSW candidate retrieval over the scoped index.

Case	Term location	View relation	Expected result
Hidden-only	t only in $h \notin V$	hidden	no hit
Visible	t in $v \in V$	visible	hit above threshold
Visible+hidden	t in $v \in V$ and $h \notin V$	mixed	hit via v only
Wrong scope	t hidden under V'	hidden	no hit in $S_{V'}$ or $I_{V'}$

Table 5: Required mixed permission search evaluation. This is the operational regression for $\mathsf{CanSearch}(k, f) \iff \mathsf{CanOpen}(k, f)$.

The implemented regression in the `rsrx-beta` Beam testbed uses 100 records, 10 fields, and 10 single-field keys. The query requests all fields, while each key can open exactly one field. The term *garnet* is placed deterministically across visible and hidden fields.

Check	Observed result	Interpretation
Visible term occurrences	143 exact hits	visible remains searchable
Hidden-only term occurrences	0 false hits	hidden fields do not cause hits
Shared term in every record	100/100 hits per key	common terms stay view-scoped
Hit basis	matched = search = read	same view boundary
Hidden query fields	9 per key/request	request does not widen auth

Table 6: Executed mixed-permission regression in the Beam testbed. The result directly exercises the "search what you can open" invariant before ANN acceleration.

The acceptance criterion is binary before latency tuning: hidden-only terms must have zero causal contribution to scoped search hits, and visible terms must remain searchable. ANN parameters such as n_{probe} or HNSW ef-search affect recall inside S_V; they must not change the authorization scope.

Reproducibility evidence. The reported prototype checks are intentionally framed as operational regressions rather than asymptotic security claims. The current release bundle keeps the falsification and positive-readout evidence separate: dense offline projection keys are rejected by the measured key-size path, streaming locality is measured on fixed 200 KiB records with one-field views, and the mixed-permission regression is a deterministic hidden-only versus visible-term test. A production replication should preserve the same three gates: compact stored-state size, view-local payload readout, and zero hidden-field causal contribution before ANN latency tuning.

10 Related Work and Positioning

Searchable symmetric encryption is the closest classical family for encrypted search [4, 3]. Beam does not claim to dominate SSE; it targets semantic/vector search signals and aligns search with selective payload disclosure. The Authority/KMS path is an explicit product trade-off: Beam rejects large dense offline keys and uses managed grants to keep storage compact and views operationally manageable.

Property-preserving encryption intentionally preserves searchable structure such as order or equality. Beam's intended property is narrower: under the right key/scope, the search representation is useful; outside that scope, hidden fields should not produce authorized hits.

Functional encryption, inner-product functional encryption, attribute-based encryption, and linear secret-sharing schemes are relevant future-work families for compact offline view keys [5, 2, 1]. Beam v1 does not claim to replace them. It deliberately stays on a product architecture path: scoped search material and scoped payload disclosure share a view predicate, while compact offline function keys remain future work.

Approximate nearest-neighbor systems such as FAISS and HNSW provide the practical search substrate [6, 7]. In Beam they are not security boundaries. They can improve latency only after the candidate set has already been restricted to the authorized view scope.

11 Limitations and Non-Claims

Beam v1 does not claim malicious live-provider privacy, ORAM-style access-pattern hiding, a completed FE/IPFE/ABE replacement, or a new ANN primitive. It also does not claim that the bucket-local transform itself is a new confidentiality primitive. AEAD protects field-bucket plaintexts; the Beam transform provides stored-state projection and read locality.

Open work includes a full CPA/KPA game for the payload/search composition, search-scope leakage formalization for KPT, persisted ANN index lifecycle tests, HNSW memory-wall measurements for many isolated views, and native/SIMD implementation.

12 Conclusion

Beam treats selective disclosure and searchable visibility as the same view-bound projection from one compact protected state. The main invariant is operationally simple: search what you can open. The algebraic mechanism gives exact authorized collapse, hidden-field zero contribution, bucket-local readout, and union-only collusion under issued views. The systems evidence shows that the design avoids payload/view explosion and changes the readout scaling law from record-linear to view-linear, while leaving KMS, ANN, and production hardening as engineering layers around the core model.

References

[1] Shweta Agrawal, David Mandell Freeman, and Vinod Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In *Advances in Cryptology – ASIACRYPT 2011*, pages 21–40, 2011.

[2] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *Theory of Cryptography*, pages 253–273, 2011.

[3] David Cash, Joseph Jaeger, Stanislaw Jarecki, Charanjit Jutla, Hugo Krawczyk, Marcel-Catalin Rosu, and Michael Steiner. Dynamic searchable encryption in very-large databases: Data structures and implementation. In *Proceedings of the 21st Annual Network and Distributed System Security Symposium*, 2014.

[4] Reza Curtmola, Juan Garay, Seny Kamara, and Rafail Ostrovsky. Searchable symmetric encryption: Improved definitions and efficient constructions. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, pages 79–88, 2006.

[5] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, pages 89–98, 2006.

[6] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.

[7] Yury A. Malkov and Dmitry A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(4):824–836, 2020.

[8] Nino Wagensonner and Dr. Tehseen Rug. Keyed phase transform: Authorized semantic search over protected vector data. Isar Innovations Research Paper, June 2026. Released June 2, 2026. https://isar-innovations.dev/research/kpt-whitepaper-reviewed-2026-06-02.pdf.