OpenSSL CONFERENCE

A BOUNCY CASTLE ENGINEERING CASE STUDY

When Hallucinations Become Vulnerabilities

Building Reliable AI for Cryptography

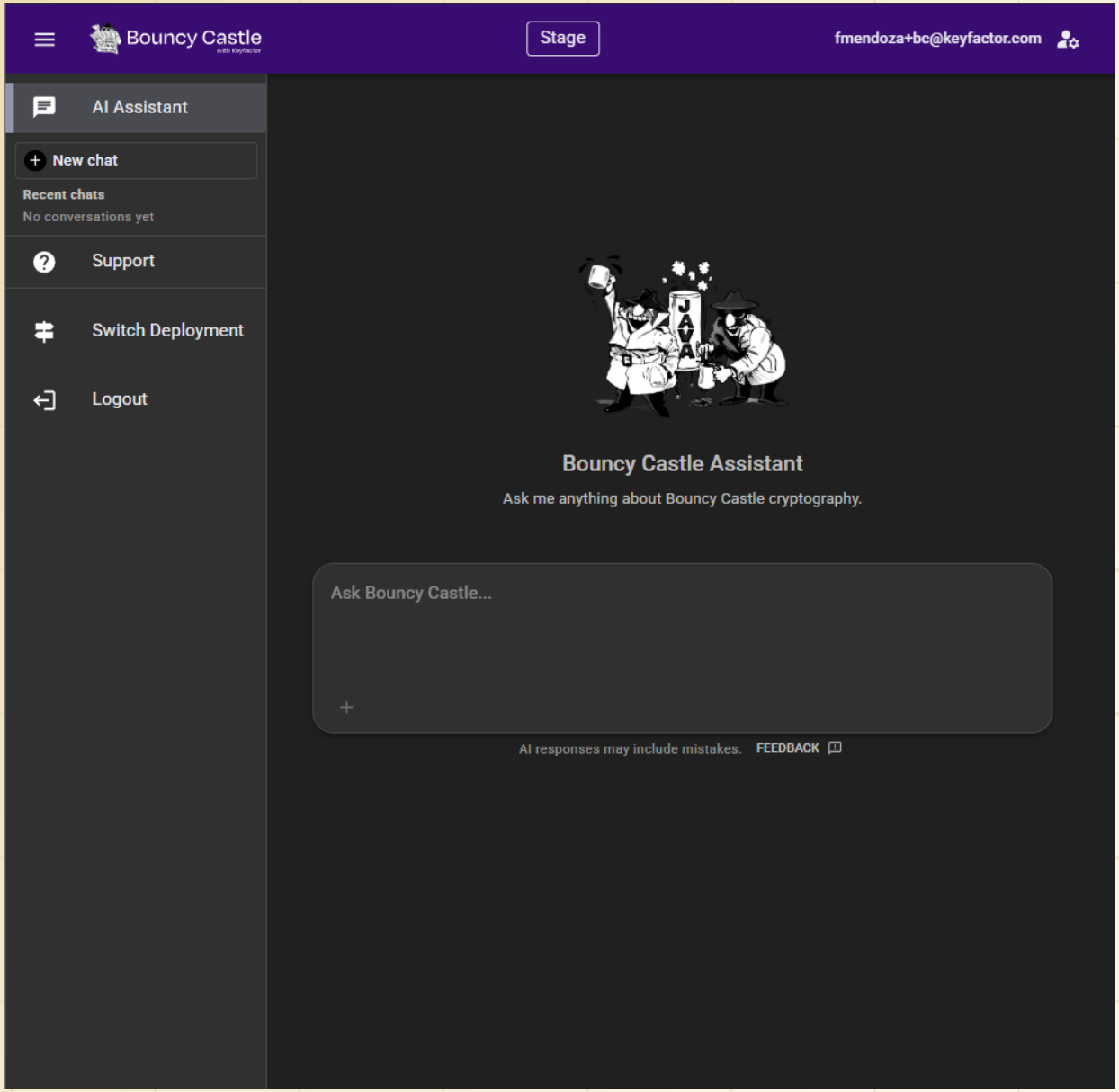
Francis Mendoza
Keyfactor / Bouncy Castle

OPENSSL CONFERENCE / PRAGUE / OCTOBER 2026

A Formidable Engineering Marvel

An “Unholy Marriage” Of Cryptography And LLMs

- Can an LLM produce valid cryptography code and advice pertaining to it?
- Large Language Models probabilistically generate tokens for “plausibly valid”, nondeterministic outputs
- Cryptography requires guarantees and rigorous correctness
- However, nondeterminism is ***not*** mutually exclusive to validity



OpenSSL CONFERENCE

A working demo can still be insecure

Looks plausible.
Uses a real API.
Still unsafe.

A fresh cipher object does not make a reused key / nonce pair safe.

ILLUSTRATIVE UNSAFE SKETCH / BC JAVA

```
byte[] nonce = new byte[12];
for (byte[] msg : messages) {
  var gcm = GCMBlockCipher.newInstance(
      AESEngine.newInstance());
  gcm.init(true, new AEADParameters(
      new KeyParameter(key), 128, nonce));
   / encrypt msg    .
}
```

A round-trip test does not establish nonce safety.

RFC 5116, section 5.1.1 / Bouncy Castle GCMBlockCipher API

OpenSSL CONFERENCE

So A ChatGPT Clone... but for Cryptography?

01 / DECISION

Guidance

Which construction is appropriate for this application?

02 / IMPLEMENTATION

Library usage

How do I use this API, in this language and release?

03 / ARTIFACT

Code assistance

Can you show an example with its assumptions and source?

Bouncy Castle adds real context: Java or C#; FIPS or non-FIPS; release-specific APIs.

Case-study scope: author abstract / Bouncy Castle documentation

OpenSSL CONFERENCE

Six questions; one engineering argument

01 Risk — 4:00

What can a wrong answer break?

02 Contract — 4:15

What must a releasable answer satisfy?

03 Evidence — 7:30

How does the system obtain support?

04 Gates — 10:15

Which checks constrain the output?

05 Evaluation — 7:00

What fails, and how do we test it?

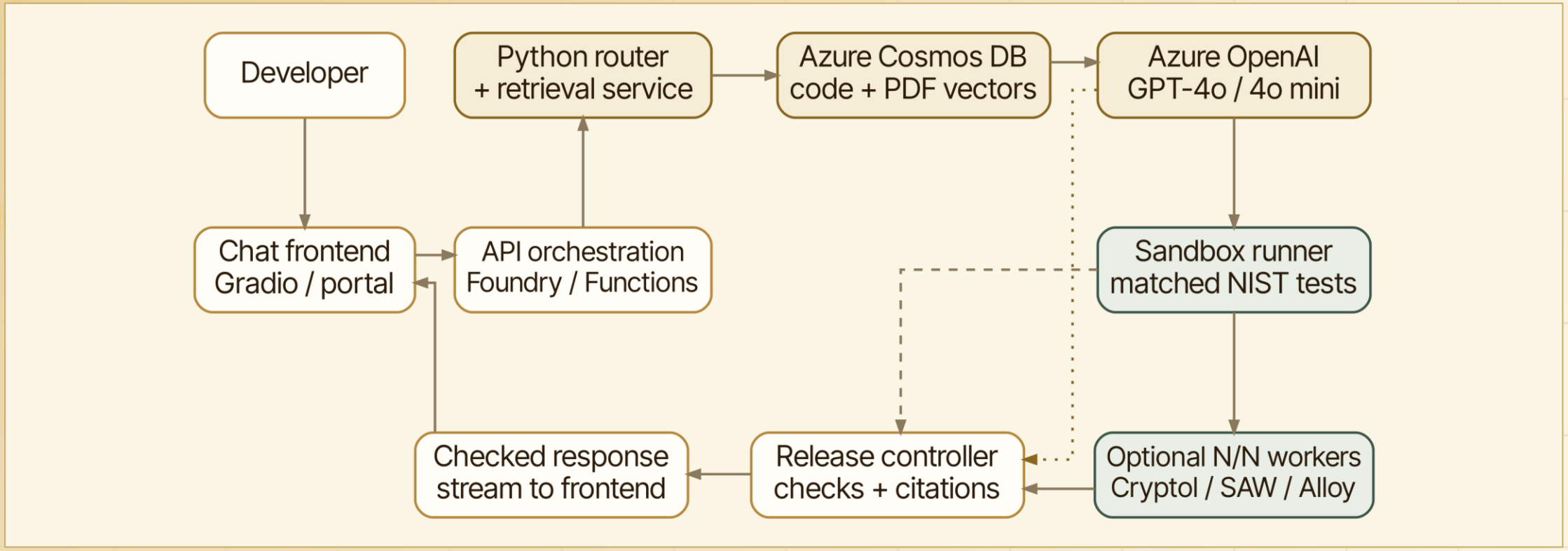
06 Transfer — 2:00

What carries over to your tooling?

35 MINUTES OF MATERIAL + 5 MINUTES OF QUESTIONS

OpenSSL CONFERENCE

One application; two technical backends

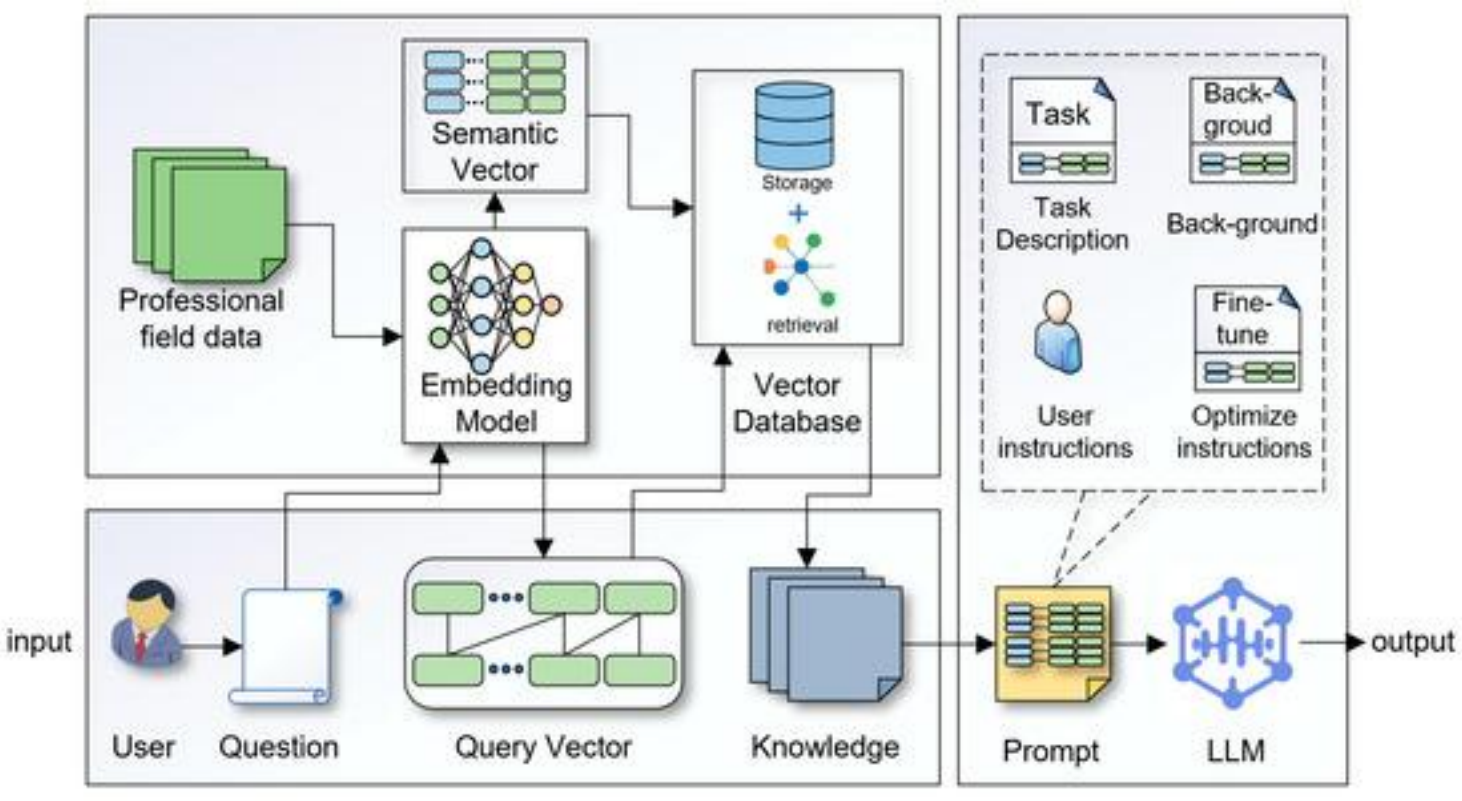


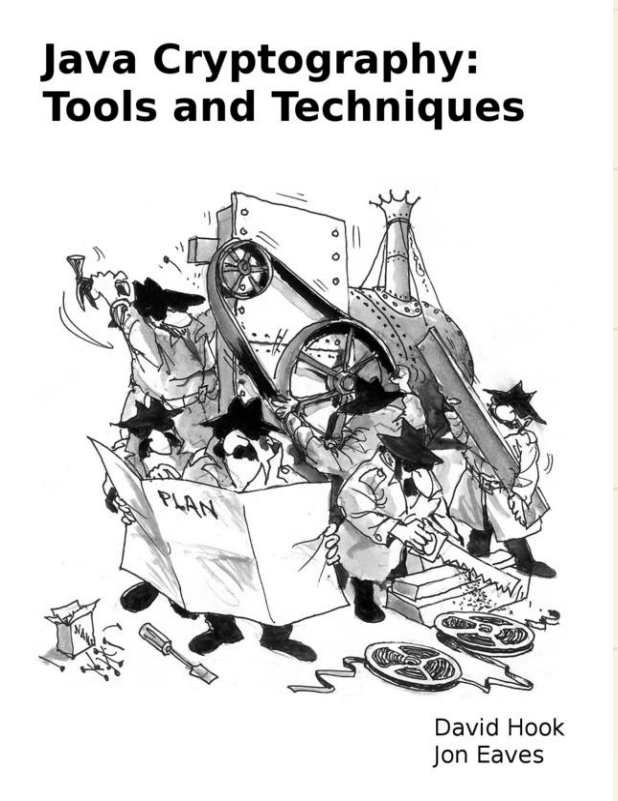
The AI backend proposes. The crypto backend checks. The application releases.

CONSOLIDATED PROJECT ARCHITECTURE / DASHED: STANDARD PATH / DOTTED: NON-CODE PATH

OpenSSL CONFERENCE

The Retrieval Augmented Generation (RAG) Backend





Since training the model is NOT sufficient, you need RAG: for both an offline and online corpus

CONSOLIDATED PROJECT ARCHITECTURE / DASHED: STANDARD PATH / DOTTED: NON-CODE PATH

OpenSSL CONFERENCE

We're going to need a MUCH BIGGER harness

BOUNCY CASTLE ASSISTANT / SCHEMATIC UI

Java	FIPS / non-FIPS	Model	Deep check: off

"Adapt this AES-GCM example to decrypt a decrypt a message and reject a bad tag."

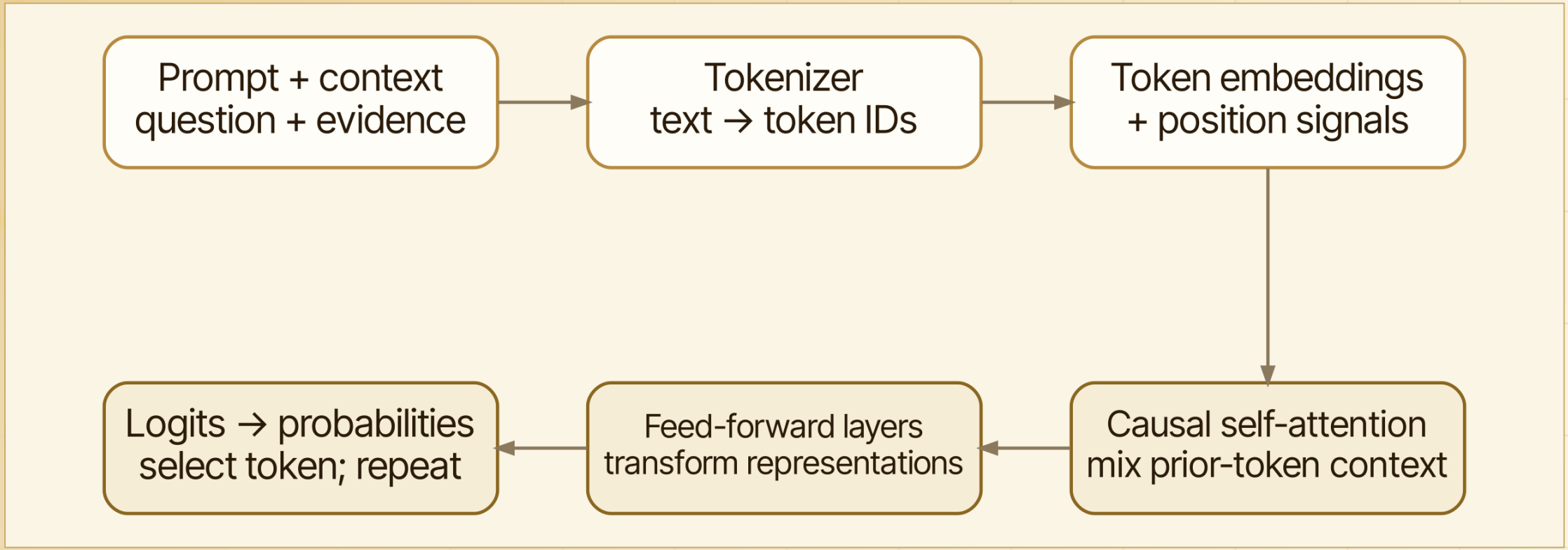
Retrieving evidence -> preparing candidate -> checking -> checking -> answer

01 Library scope
02 Evidence and source links
03 Check status and limitations

A fast-looking chat interface still needs a slow, explicit acceptance decision.

OpenSSL CONFERENCE

A language model predicts the next token



P(next token | context) is not P(the answer is true). Probability is NOT validity!

OpenSSL CONFERENCE

Three model roles; three different jobs

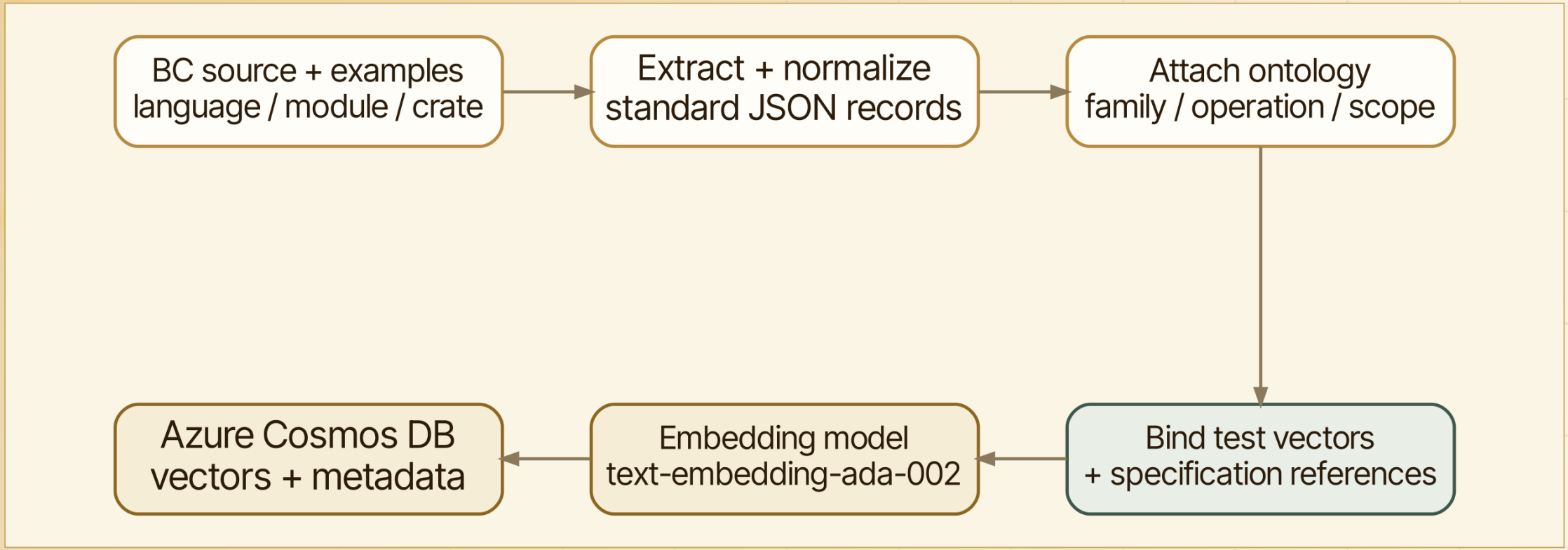
ROLE	HISTORICAL BASELINE / CHOICE	WHAT IT DOES
Embedding	text-embedding-ada-002 1,536-dimensional vectors	Maps queries and evidence into a similarity space.
Generation	GPT-4o / GPT-4o mini code-capable model on CODE path	Explains and adapts code from retrieved context.
Reranking	Query-passage scoring model choice varies by revision	Reorders retrieved candidates; does not prove their correctness.

RAG supplies context at inference time; it is not the same as retraining the generator.

HISTORICAL MODEL NAMES / NOT A CLAIM ABOUT TODAY'S FRONTIER

OpenSSL CONFERENCE

The part more important than the model: the data



Code, scope metadata and verification artifacts were prepared together.

AUTHOR-REPORTED PIPELINE / JSON EXAMPLES ON LATER SLIDES ARE ILLUSTRATIVE

OpenSSL CONFERENCE

A snippet was more than a string of code

ILLUSTRATIVE NORMALIZED RECORD

```
{
  "family": "symmetric / AEAD",
  "unit": "mode implementation",
  "algorithm": "AES-GCM",
  "operation": "decrypt",
  "scope": { "language": "Java",
             "edition": "selected",
             "release": "pinned" },
  "code_ref": "snippet-id",
  "source_ids": ["guide-section-id"],
  "test_vector_ids": ["matched-set-id"]
}
```

Retrieve the right operation.

Resolve the source.

Fetch the matching tests.

Stable identifiers bind the code to its documentation and its test oracle.

OpenSSL CONFERENCE

Snippet comments were vital to diff embeddings

Encryption

```
/*==
    id: "ecb_mode_encryption_ex6"
    title: "AES-ECB encrypt (PKCS7Padding)"
    desc: "Operation: encrypt | Primitive: AES/ECB + PKCS7Padding | Input format: plaintext byte[] (Strings.toByteArra
==*/

byte[] plainText = Strings.toByteArray("The password will be sent on Tuesday");

Cipher cipher = Cipher.getInstance("AES/ECB/PKCS7Padding", "BCFIPS");
cipher.init(Cipher.ENCRYPT_MODE, key);
byte[] cipherText = cipher.doFinal(plainText);

System.out.println(Hex.toHexString(cipherText));

/*== .. ==*/
```

Decryption

```
/*==
    id: "ecb_mode_decryption_ex6"
    title: "AES-ECB decrypt (PKCS7Padding)"
    desc: "Operation: decrypt | Primitive: AES/ECB + PKCS7Padding | Input format: ciphertext hex -> byte[] (Hex.decode
==*/
byte[] cipherText = Hex.decode("c711383e1d2f265f70e607b4a3ce12ae6cbb8263079a2693c2d31913ae2b937209a24fa4482602b748c9ee
Cipher cipher = Cipher.getInstance("AES/ECB/PKCS7Padding", "BCFIPS");
cipher.init(Cipher.DECRYPT_MODE, key);
byte[] plainText = cipher.doFinal(cipherText);

System.out.println(Strings.fromByteArray(plainText));

/*== .. ==*/
```

operation=encrypt | seal | init-for-encryption

operation=decrypt | open | authentication-failure

Explicit operation metadata prevents semantic neighbors from becoming false substitutes.

AEAD CONTRACTS DIFFER; SOME MODES REUSE A PRIMITIVE BUT NOT THE WHOLE CALLER CONTRACT.

OpenSSL CONFERENCE

JSONL of snippet ontology for RAG harness

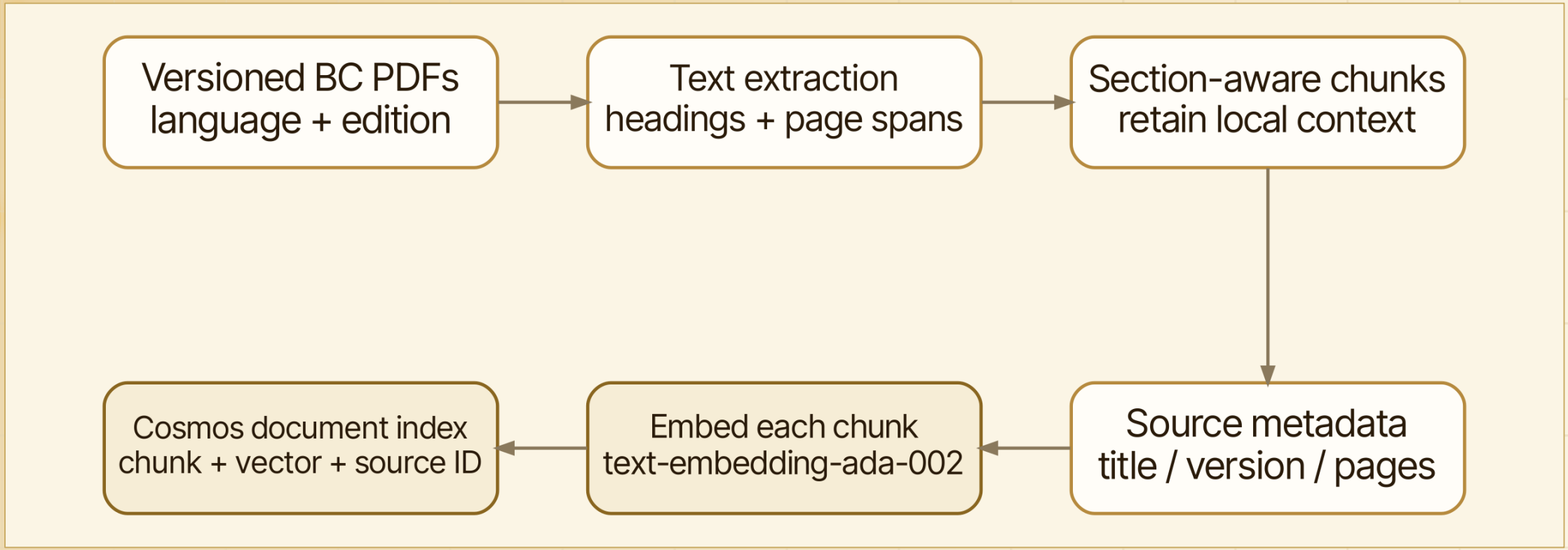
output > {} fips-java.jsonl


```
{"id": "to_hex_string", "title": "Encode a byte array using the Hex encoder", "desc": "Operation: encode | Primitive: hexadecimal (base-16) | Input format: byte[] (arbitrary binary dat
{"id": "to_base64_string", "title": "Encode a byte array using the Base64 encoder", "desc": "Operation: encode | Primitive: Base64 (RFC 4648) | Input format: byte[] (arbitrary binary d
{"id": "decode_hex_string", "title": "Decode a byte array from a Hex encoded string.", "desc": "Operation: decode | Primitive: hexadecimal (base-16) | Input format: hex string (even le
{"id": "print_byte_array", "title": "Print byte array", "desc": "Operation: debug output | Primitive: hexadecimal (base-16) | Input format: byte[] | Output: hex string printed to stdou
{"id": "from_base64_string", "title": "Decode a byte array encoded as a Base64 string", "desc": "Operation: decode | Primitive: Base64 (RFC 4648) | Input format: Base64 string (ASCII,
{"id": "print_byte_array_base64", "title": "Print byte array in Base64", "desc": "Operation: debug output | Primitive: Base64 | Input format: byte[] | Output: Base64 string printed to
{"id": "create_mldsa_keypair", "title": "Create an ML-DSA Key Pair using the provider", "desc": "Operation: generate | Primitive: ML-DSA (post-quantum signature) | Input format: parame
{"id": "sign_msg_mldsa", "title": "Sign a message using MLDSA and the provider", "desc": "Operation: sign | Primitive: ML-DSA | Input format: byte[] message (raw bytes) | Output: byte[
{"id": "verify_msg_mldsa", "title": "Verify a message using ML-DSA and the provider", "desc": "Operation: verify | Primitive: ML-DSA | Input format: byte[] message + byte[] signature |
{"id": "ml_kem_key_generator_512", "title": "Create a ML-KEM-512 key pair using the provider", "desc": "Operation: generate | Primitive: ML-KEM-512 (post-quantum KEM) | Input format: p
{"id": "ml_kem_key_factor_public_512", "title": "Recreate an ML-KEM-512 public key from encoded ML-KEM public key", "desc": "Operation: decode | Primitive: ML-KEM-512 public key | Inpu
{"id": "ml_kem_key_factor_private_512", "title": "Recreate an ML-KEM-512 private key from encoded ML-KEM private key", "desc": "Operation: decode | Primitive: ML-KEM-512 private key |
{"id": "ml_kem_key_generator_768", "title": "Create a ML-KEM-768 key pair using the provider", "desc": "Operation: generate | Primitive: ML-KEM-768 (post-quantum KEM) | Input format: p
{"id": "ml_kem_key_factor_public_768", "title": "Recreate an ML-KEM-768 public key from encoded ML-KEM public key", "desc": "Operation: decode | Primitive: ML-KEM-768 public key | Inpu
{"id": "ml_kem_key_factor_private_768", "title": "Recreate an ML-KEM-768 private key from encoded ML-KEM private key", "desc": "Operation: decode | Primitive: ML-KEM-768 private key |
{"id": "ml_kem_key_generator_1024", "title": "Create a ML-KEM-1024 key pair using the provider", "desc": "Operation: generate | Primitive: ML-KEM-1024 (post-quantum KEM) | Input format
{"id": "ml_kem_key_factor_public_1024", "title": "Recreate an ML-KEM-1024 public key from encoded ML-KEM public key", "desc": "Operation: decode | Primitive: ML-KEM-1024 public key | I
{"id": "ml_kem_key_factor_private_1024", "title": "Recreate an ML-KEM-1024 private key from encoded ML-KEM private key", "desc": "Operation: decode | Primitive: ML-KEM-1024 private key
{"id": "create_slh-dsa-signature", "title": "Sign bytes with SLH-DSA (slh_dsa_sha2_128f)", "desc": "Operation: sign | Primitive: SLH-DSA (SLHDSAParameterSpec.slh_dsa_sha2_128f) | Input
{"id": "verify_slh-dsa-signature", "title": "Verify SLH-DSA signature (slh_dsa_sha2_128f)", "desc": "Operation: verify | Primitive: SLH-DSA (SLHDSAParameterSpec.slh_dsa_sha2_128f) | In
{"id": "key_gen_id_slh_dsa_sha2_128f", "title": "Generate SLH-DSA keypair (slh_dsa_sha2_128f)", "desc": "Operation: keygen | Primitive: SLH-DSA (SHA2-128f) | Input format: SecureRandom
{"id": "decode_pub_id_slh_dsa_sha2_128f", "title": "Decode SLH-DSA public key (X.509 SPKI, sha2_128f)", "desc": "Operation: decode public key | Primitive: SLH-DSA (SHA2-128f) | Input f
{"id": "decode_priv_id_slh_dsa_sha2_128f", "title": "Decode SLH-DSA private key (PKCS#8, sha2_128f)", "desc": "Operation: decode private key | Primitive: SLH-DSA (SHA2-128f) | Input fo
{"id": "key_gen_id_slh_dsa_sha2_128s", "title": "Generate SLH-DSA keypair (slh_dsa_sha2_128s)", "desc": "Operation: keygen | Primitive: SLH-DSA (SHA2-128s) | Input format: SecureRandom
```


The “desc” field in particular had various subfields for high specificity; this was merely a seed input

OpenSSL CONFERENCE

PDFs became page-addressable evidence



Chunk for retrieval without severing the passage from its document and page span.

TITLE + EDITION + VERSION + PAGE SPAN + SOURCE ID TRAVEL WITH EACH CHUNK

OpenSSL CONFERENCE

Cached doc explanations, code, etc. for citations

2.1.2 Enabling GCM for TLS 1.2 in the BCJSSE Provider in FIPS mode

For TLS 1.2, the GCM nonce used in FIPS mode is required to be generated in the cryptographic boundary of the FIPS module providing the cryptographic services being used by the BCJSSE.

In the case of the bctls-fips distribution of the BCJSSE, the BCJSSE jar is configured already to recognise the BCFIPS provider, no work on a user's part is required.

Where the BCFIPS module is not in use and some other Java FIPS provider is being used an override for the JcaTlsCrypto class needs to be provided which is used to fetch the in-boundary nonce generator by using an implementation of the AEADNonceGeneratorFactory. Briefly the following is required:

```
public class FipsJcaTlsCrypto extends JcaTlsCrypto
{
   public FipsJcaTlsCrypto(
         JcaJceHelper helper, SecureRandom entropySource, SecureRandom nonceEntropySource)
   {
      super(helper, entropySource, nonceEntropySource);
   }

   @Override
   public AEADNonceGeneratorFactory getFipsGCMNonceGeneratorFactory()
   {
      return <in-boundary derived implementation of AEADNonceGeneratorFactory goes here>
   }
}
```

citations > {} citations_bctls_java_2_1_20.json > { } 1 > abc pattern

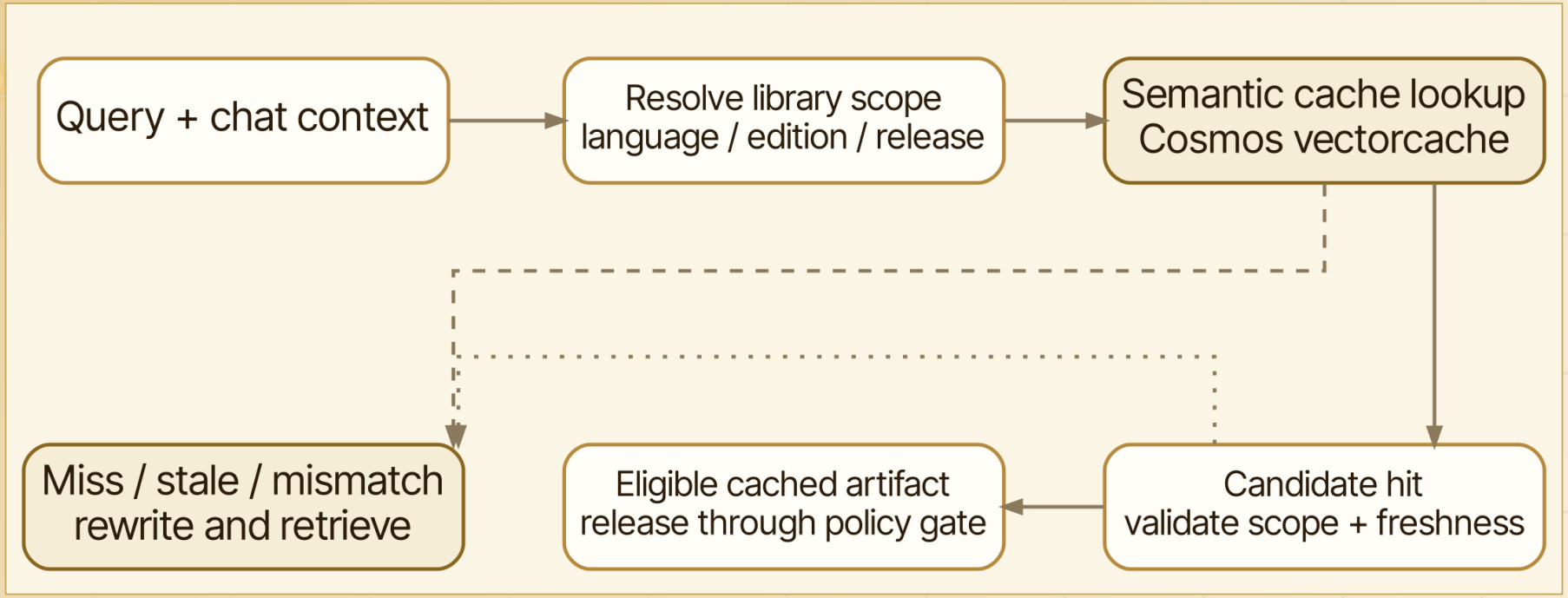

```
30    {
31      "example_number": "2.1.2",
32      "example_title": "Enabling GCM for TLS 1.2 in the BCJSSE Provider
33      "page_number": 4,
34      "title": "Java (D)TLS API and JSSE Provider User Guide",
35      "version": "2.1.20",
36      "date": "2025-01-12",
37      "authors": [
38        "Legion of the Bouncy Castle Inc."
39      ],
40      "filename": "BC-FJA-(D)TLSUserGuide-2.1.20.pdf",
41      "pattern": "section_3part",
42      "raw_line": "2.1.2 Enabling GCM for TLS 1.2 in the BCJSSE Provider
43    },
```

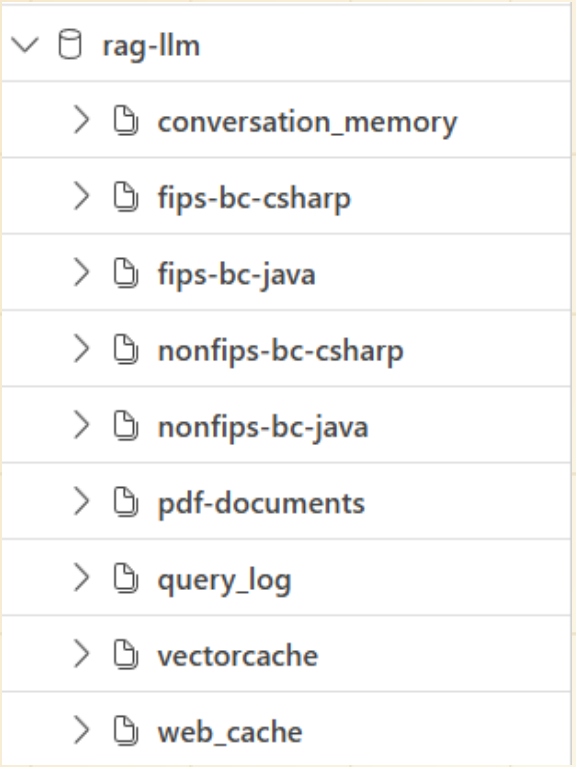

As part of the ontology, we also chunked according to language, FIPS/nonFIPS, and then topic

TITLE + EDITION + VERSION + PAGE SPAN + SOURCE ID TRAVEL WITH EACH CHUNK

OpenSSL CONFERENCE

First resolve scope, then try the cache



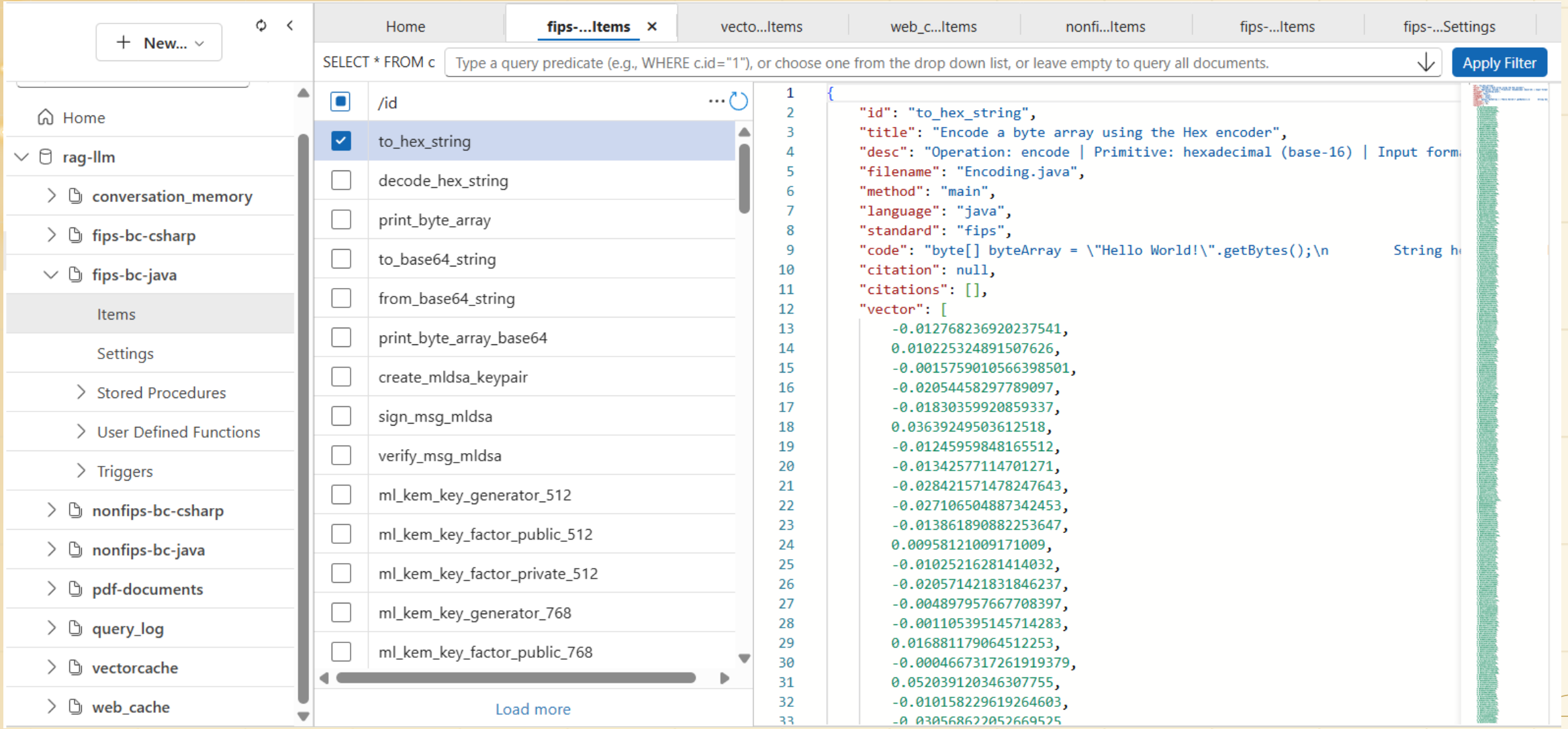


Semantic similarity is only a cache candidate—not permission to reuse a crypto answer.

REUSE CONTRACT: MATCH SCOPE, ARTIFACT VERSION AND APPLICABLE VERIFICATION POLICY

OpenSSL CONFERENCE

Earlier Embeddings in a production CosmosDB



OpenSSL CONFERENCE

Determinism in the harness: sentiment-analyze intent

ROUTE	EXAMPLE REQUEST	BACKEND CONSEQUENCE
QUESTION	Explain GCM nonce requirements. requirements.	Retrieve evidence; compose a grounded explanation.
CODE	Write a decrypt function that rejects invalid tags.	Fetch code and tests; generate; execute verification.
ARBITRATION	Which of these two API paths meets my constraints?	Compare applicable evidence; route any code through CODE.

Arbitration is an application route—not an exemption from code verification.

OpenSSL CONFERENCE

Hybrid Intent Classifier: Lexical + Vector Search

```
score_c(q) = alpha * L_c(q) + (1 - alpha) * S_c(q)

c in {QUESTION, CODE, ARBITRATION};  0 ⩽ alpha ⩽ 1
```

01	**Unambiguous explicit rule rule**	Honor the configured route. route.
02	**No decisive rule**	Use the highest score with a with a margin.
03	**Conflict or low margin**	Clarify instead of forcing a a class.

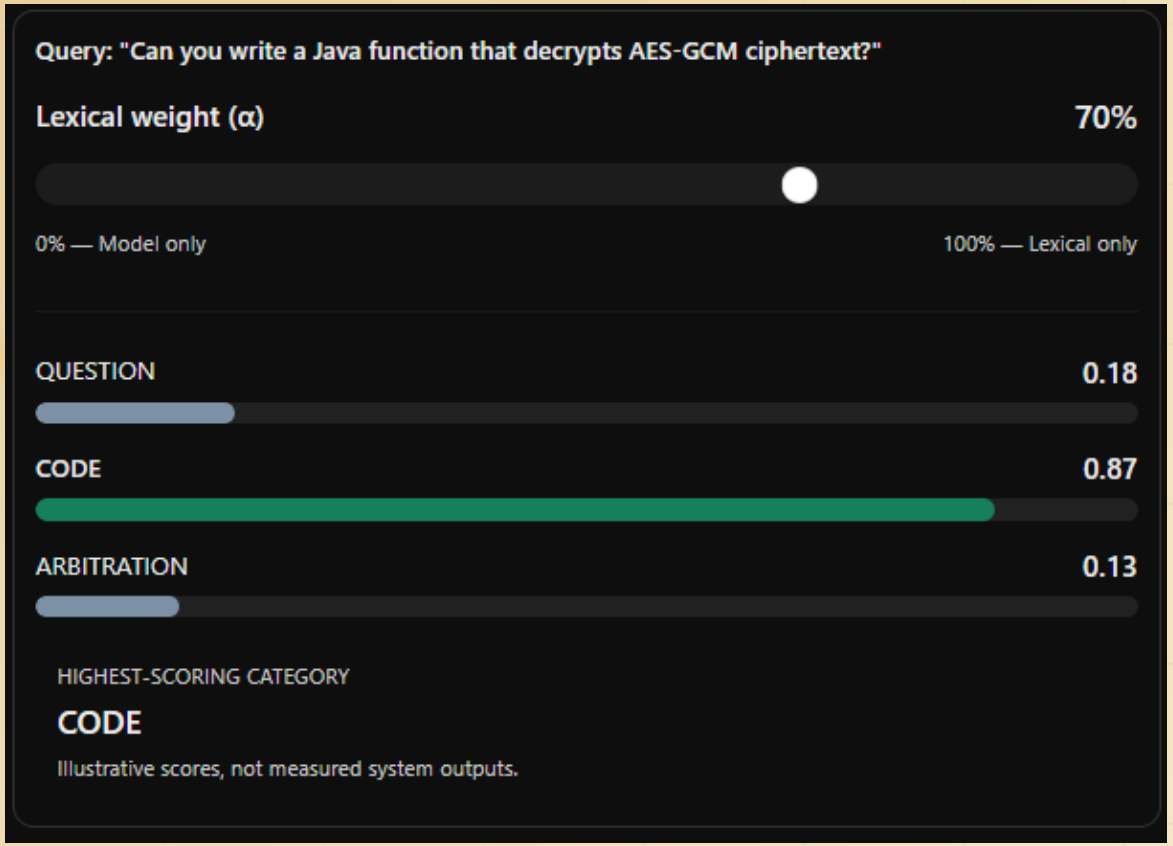
Symbol	Meaning
q	The user's query
c	A candidate classification: QUESTION, CODE, or ARBITRATION
$L_c(q)$	Lexical score: how strongly the query's words or phrases indicate category c
$S_c(q)$	Semantic score: how strongly the query's meaning indicates category c, as determined by a model
α	Weight assigned to lexical evidence, between 0 and 1
$1 - \alpha$	Weight assigned to semantic evidence
$\text{score}_c(q)$	Combined score for category c

BM25 ranks lexical evidence; deterministic policy supplies the hard guarantee.

ILLUSTRATIVE SCORING MODEL / NOT CALIBRATED PROBABILITIES OR CLAIMED DEPLOYED WEIGHTS

OpenSSL CONFERENCE

Hybrid Intent Classifier Example: pick highest score



$$\text{score}_c(q) = \alpha L_c(q) + (1 - \alpha) S_c(q)$$

- Lexical score for CODE = 0.9
- Semantic score for CODE = 0.8
- Alpha = 0.7 (70% lexical, 30% semantic)

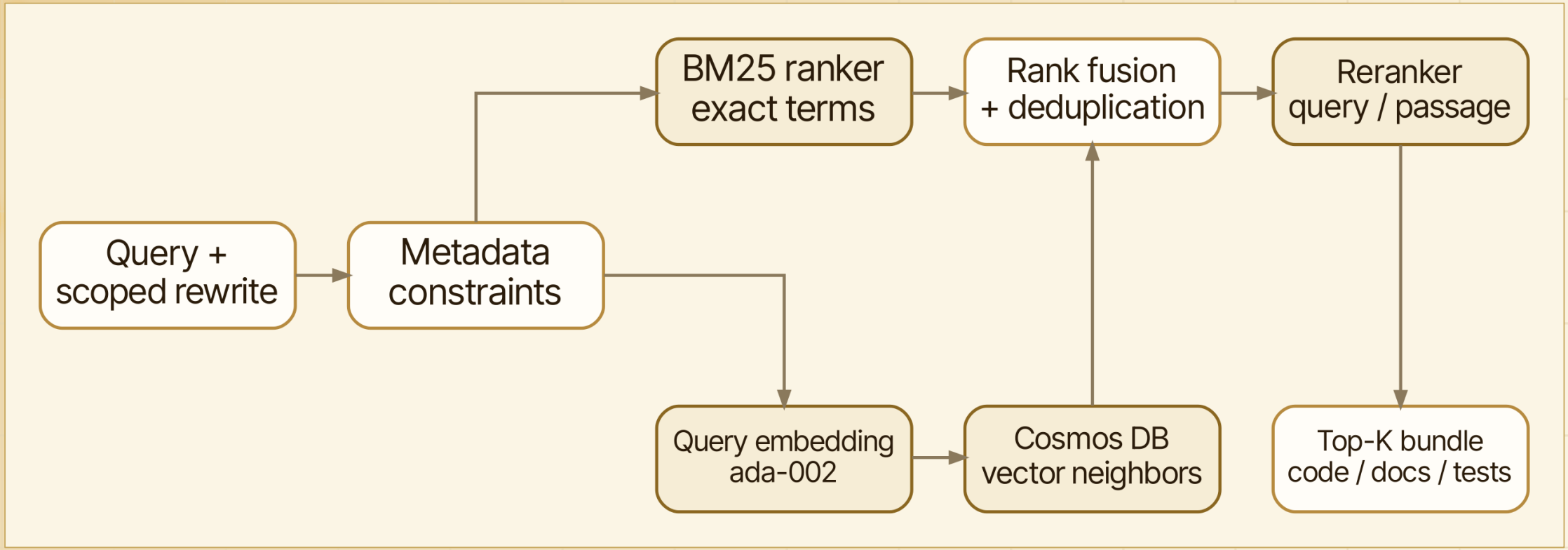
$$\text{score}_{CODE}(q) = 0.7(0.9) + 0.3(0.8)$$

$$\boxed{\text{score}_{CODE}(q) = 0.87}$$

We repeat this calculation for QUESTION, CODE, and ARBITRATION with different lex + semantic weights

OpenSSL CONFERENCE

Hybrid Snippet Retrieval finds names and meaning



Similar logic for snippets. Metadata differentiates, BM25 and vectors improve candidate recall.

BM25: EXACT SYMBOLS / TERMS | DENSE: PARAPHRASES | RERANKER: QUERY-PASSAGE FIT

OpenSSL CONFERENCE

Fetch a bundle—not an isolated snippet

CODE ARTIFACT

What to adapt

Operation, signature, dependencies, input constraints and pinned library scope.

SOURCE ARTIFACT

What supports it

Relevant guide passages, source identifiers, edition, edition, release and page spans.

TEST ARTIFACT

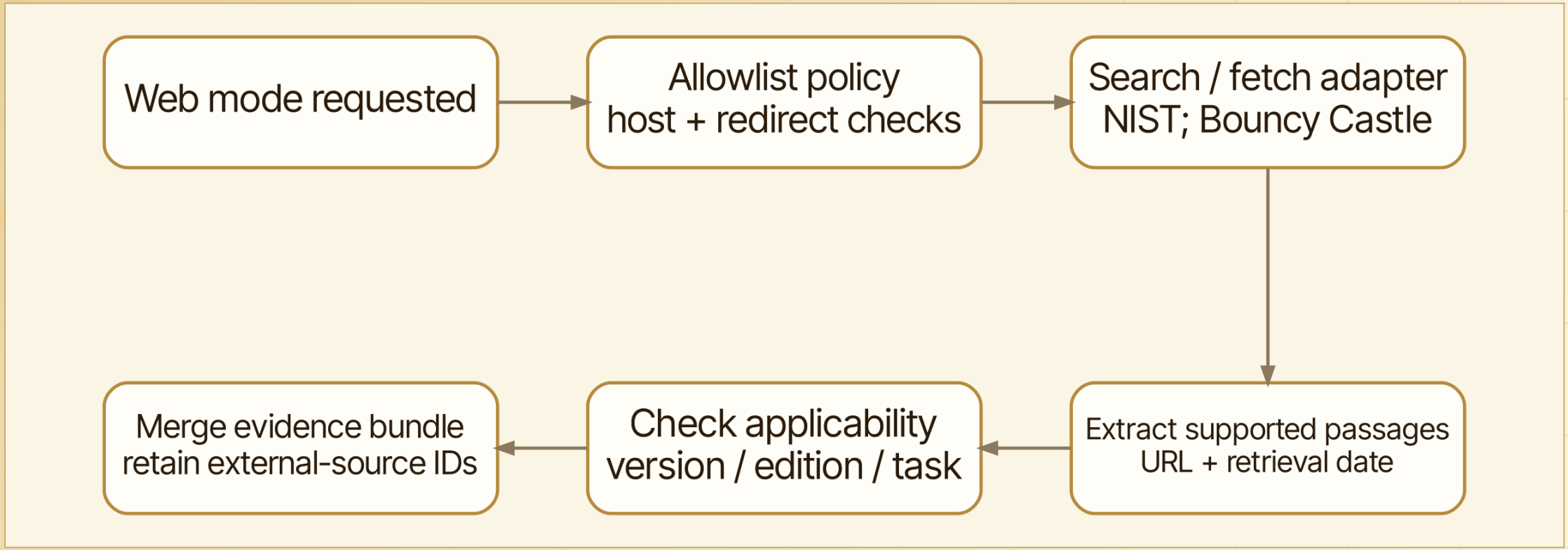
What will judge it

Matching NIST vectors plus plus expected outputs and and operation-specific failure cases.

Match test sets to algorithm, mode, operation and parameters—not merely a similar title.

OpenSSL CONFERENCE

Web RAG/Online Search

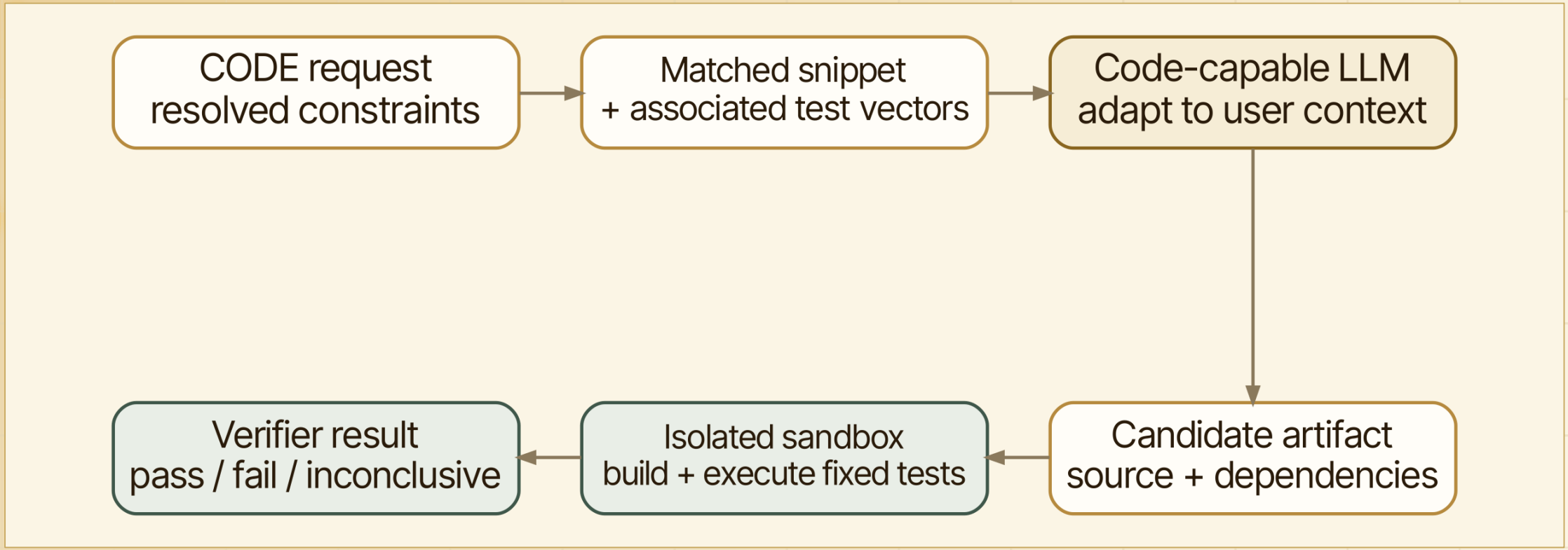


Allowlisted source content remains data; it must not become application instructions.

EXAMPLES: NIST / BOUNCY CASTLE / CHECK HOSTS, REDIRECTS, APPLICABILITY AND SOURCE IDENTITY

OpenSSL CONFERENCE

After generating the code, verify the outputs



The LLM-generated artifact then gets tested in a sandbox and then verified before being displayed

What To Check For In An LLM-Generated Artifact

PROPERTY	REQUIRED MODEL / CHECK	CLAIM BOUNDARY
Memory safety	Explicit pointer, bounds and heap obligations; obligations; SAW memory assertions where supported.	A functional proof alone does not does not prove no overflow or no leaks.
Side-channel resistance	Explicit secret/public split and leakage model; model; check secret-independent observable traces.	Functional equivalence does not not imply constant-time execution.
Protocol behavior behavior	Alloy model of roles, messages and invariants; invariants; search the declared scope when applicable.	No counterexample within scope is not an unrestricted proof.

“Checked” is meaningful only when the property and the model are named.

ALLOY CAN USE OTHER ANALYSIS MODES; REPORT THE ACTUAL SIGNATURE AND TIME SCOPES USED.

OpenSSL CONFERENCE

The sandbox tests the adapted candidate

CHECK	INPUT / EXPECTATION	WHAT A FAILURE MEANS
Build + API fit	Pinned dependencies; expected signatures.	The adaptation cannot run in its declared scope.
NIST known answers	Fixed inputs -> independently expected expected outputs.	The candidate disagrees with the selected vector set.
Contract failures	Bad tag, bad length, wrong parameters. parameters.	Error handling or wrapper behavior violates the contract.
Execution controls	Resource budget; restricted environment.	The run is unsafe, stalled or inconclusive—not a pass.

A test class is generated over code. Docker containers ran either Java (Maven + JUnit) or C# (.NET SDK + xUnit)

NIST PUBLIC VECTORS DO NOT REPLACE CAVP VALIDATION

OpenSSL CONFERENCE

A green test suite is NOT a universal proof

TESTS

These selected cases matched the oracle.

PROOF

This modeled property holds under these assumptions.

Neither statement means: "all of Bouncy Castle, every wrapper, every side channel, every protocol”

State exactly what was checked, on which artifact, and what remains outside scope.

VERIFICATION CLAIMS ARE PARAMETERIZED BY SPECIFICATION, ASSUMPTIONS AND COVERAGE.

OpenSSL CONFERENCE

(Deprecated) N/N Deep mode can only verify a subset

STANDARD CODE PATH

Fast enough to converse

Build the candidate. Run matching vectors and contract tests. Return test test scope and limitations.

OPTIONAL N/N PATH

Wait for deeper evidence

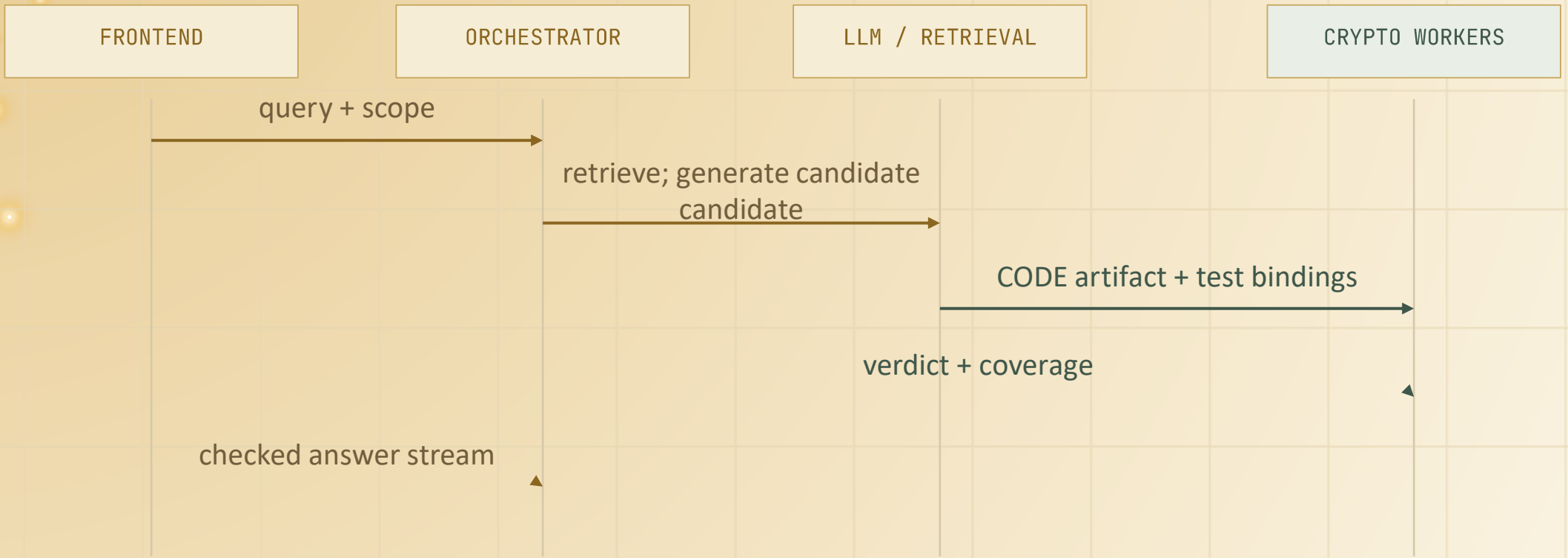
Longer jobs and higher compute cost. cost. Only supported core AES engines, engines, selected ECC arithmetic and and core hash functions. SAW only worked for Java

A promising but scrapped feature for prod due to limited ontological coverage

AUTHOR-REPORTED BETA COVERAGE / EXACT ALGORITHMS, TARGETS AND PROOF MANIFESTS MUST BE ENUMERATED

OpenSSL CONFERENCE

Stream progress; release checked content

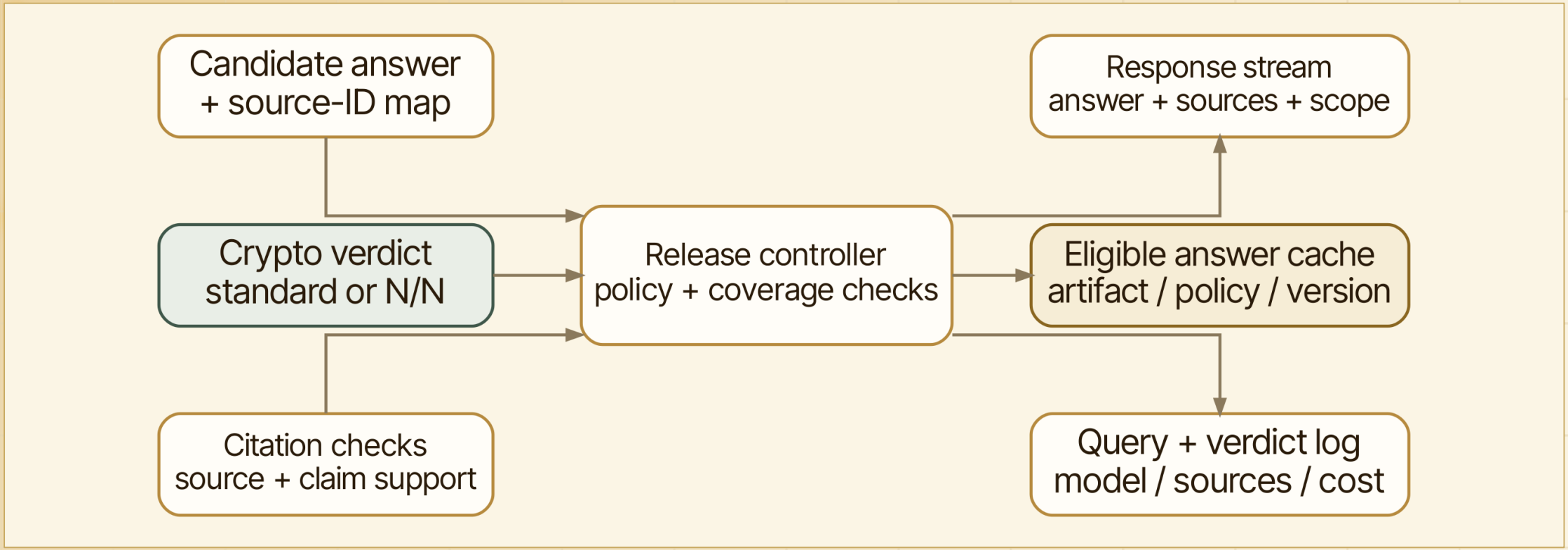


Immediate status updates do not require exposing an unchecked code artifact.

RELEASE CONTRACT / STRUCTURED SEGMENTS CAN STREAM ONLY AFTER THEIR APPLICABLE CHECKS

OpenSSL CONFERENCE

Release the evidence with the answer



Valid source ID + supporting passage + applicable scope: all three matter.

CACHE ONLY ELIGIBLE ARTIFACTS / RETAIN VERDICT AND SOURCE LINEAGE FOR REPLAY

Bouncy Castle | Stage | fmendoza+bc@keyfactor.com

AI Assistant

New chat

Recent chats

Give me Bouncy Castle code to...

Support

Switch Deployment

Logout

Give me Bouncy Castle code to create a SHA-1 message digest of a byte array using java.sec...

Give me Bouncy Castle code to create a SHA-1 message digest of a byte array using java.security.MessageDigest

FindRelevantSources ✓ 5443ms

SearchDocs ✓ 491ms

SearchDocs ✓ 366ms

SearchDocs ✓ 386ms

CompileJava ✓ 1679ms

Here is a complete Bouncy Castle FIPS-compliant Java example for calculating a SHA-1 message digest on a byte array using `java.security.MessageDigest`. This code ensures the FIPS provider (`BCFIPS`) is registered and active:

java

```java
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.security.NoSuchProviderException;
import java.security.Security;
import org.bouncycastle.jcajce.provider.BouncyCastleFipsProvider;

public class Sha1DigestExample {
    public static byte[] computeSha1Digest(byte[] data) throws Exception {
        // Register the Bouncy Castle FIPS provider if not already registered
        if (Security.getProvider("BCFIPS") == null) {
            Security.addProvider(new BouncyCastleFipsProvider());
        }
        MessageDigest digest = MessageDigest.getInstance("SHA-1", "BCFIPS");
        return digest.digest(data);
    }
```

Ask Bouncy Castle...

AI responses may include mistakes. FEEDBACK

Bouncy Castle | Stage | fmendoza+bc@keyfactor.com

AI Assistant

New chat

Recent chats

Bouncy Castle SHA-1 Message...

Support

Switch Deployment

Logout

Bouncy Castle SHA-1 MessageDigest Example

```java
public class Sha1DigestExample {
    public static byte[] computeSha1Digest(byte[] data) throws Exception {
        // Register the Bouncy Castle FIPS provider if not already registered
        if (Security.getProvider("BCFIPS") == null) {
            Security.addProvider(new BouncyCastleFipsProvider());
        }
        MessageDigest digest = MessageDigest.getInstance("SHA-1", "BCFIPS");
        return digest.digest(data);
    }

    // Example main method to demonstrate usage
    public static void main(String[] args) throws Exception {
        byte[] input = "Hello World!".getBytes(); // Example input, change as needed
        byte[] hash = computeSha1Digest(input);
        // Print the digest bytes in hex format
        for (byte b : hash) {
            System.out.format("%02x", b);
        }
        System.out.println();
    }
}
```

How it works:

- Ensures the Bouncy Castle FIPS provider is registered.
- Uses the JCA standard `MessageDigest` interface, requesting `"SHA-1"` from `"BCFIPS"`.
- Computes and prints the SHA-1 digest in hexadecimal.

Security notes:

- SHA-1 is considered weak for new applications; only use if you must interoperate with systems that require it. Prefer SHA-256 or higher for new designs.
- The `"BCFIPS"` provider string ensures you are using the FIPS 140-2 validated implementation.

References:

- BC-FJA (Bouncy Castle FIPS Java API) User Guide 2.1.2, 11. Message Digest Algorithms (p. 39-41)
- Java Cryptography: Tools and Techniques, 3. Message Digests, MACs, HMACs, KDFs, and XOFs

Confirmed by compilation, not documented: `org.bouncycastle.jcajce.provider.BouncyCastleFipsProvider`

Compile-verified

Ask Bouncy Castle...

AI responses may include mistakes. FEEDBACK

OpenSSL CONFERENCE

Lessons Learned From The GPT-4o Era

Correctly rejecting a bad answer still leaves the user without an answer...

MODEL	HARNESS	USER
Weak context adaptation	Repeated test / gate failures failures	Latency, retries, frustration

A verifier can reject a weak candidate; it cannot supply the missing model capability.

AUTHOR-REPORTED EXPERIENCE / NO CONTROLLED ERROR-RATE BENCHMARK CLAIMED

OpenSSL CONFERENCE

Lesson #1: Better models relieved the bottleneck

GPT-4o-ERA BASELINE

The harness said not yet

Frequent adaptation failures meant meant more repair attempts, blocked blocked answers and frustrated users. users.

FEBRUARY 2026 / OPUS 4.6

A stronger candidate generator

We saw fewer broken adaptations and a markedly better experience with stronger coding and reasoning.

Keep the gate; upgrade the model. Better first candidates reduce avoidable rejection.

OPUS 4.6 RELEASED 5 FEBRUARY 2026 / PROJECT IMPROVEMENT IS QUALITATIVE AUTHOR OBSERVATION

OpenSSL CONFERENCE

Lesson #2: Curation cost grew with the data

```
T_maintenance = sum(i=1..N) c_i  ~  O(N)
```

Metadata	Family, operation and scope
Evidence	Documents, pages and provenance
Checks	Vectors, specs and build contracts contracts
Evolution	API versions and regression cases cases

Every added artifact brings recurring metadata, evidence and verification work.

COST MODEL: N MAINTAINED ARTIFACTS, ROUGHLY COMPARABLE PER-ARTIFACT WORK

OpenSSL CONFERENCE

Lesson #3: Measure acceptance, coverage and waiting

MEASURE	WHY IT MATTERS	COMPARISON RULE
Useful accepted answers answers	How often did the user get a correct, applicable result?	Keep the task set and release policy fixed.
False acceptance + abstention	Did a bad candidate pass? Did a solvable task get rejected?	Use independent review and adversarial cases.
Latency, retries and cost cost	How much work did it take to reach an eligible answer?	Record model, corpus, toolchain and coverage versions.

A more capable model is valuable when it improves the complete checked workflow.

EVALUATION FRAMEWORK / THIS SLIDE DOES NOT REPORT MEASURED RESULTS

OpenSSL CONFERENCE

Lesson #4: Reliability and usefulness are separate jobs

01 **Prepare evidence as typed, versioned artifacts.**

02 **Verify the generated artifact—not its ancestor.**

03 **State the exact property, assumptions and coverage.**

04 **Model capability still limits useful accepted answers.**

We controlled evidence and acceptance. We did not control the foundation model.

Questions

OpenSSL CONFERENCE

Where should an AI assistant be allowed to say "verified"?

Francis Mendoza (francis.mendoza@keyfactor.com)

Keyfactor / Bouncy Castle