SITG-Consulting

INDEPENDENT FORENSIC REVIEW

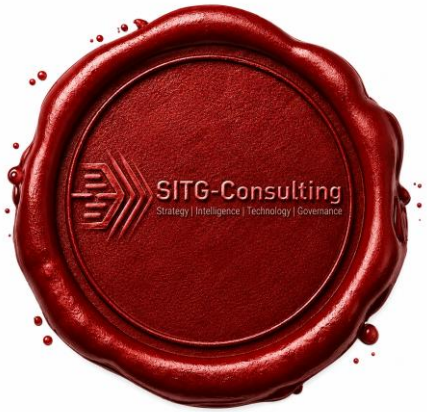
Forensic Review of IBM's Cryptographic Agility Framework

An independent technical, architectural and governance assessment of "Cryptographic Agility for Applications: An Assessment Framework and Principled API Design"

Subject Rameshan & Messmer, IBM Research Europe (Zurich), 2026

Prepared for CISOs · Chief & Enterprise Architects · Cryptographers · Risk & Board Advisors · Standards Bodies

Prepared by SITG-Consulting · Independent assurance and validation



REVIEWER'S NOTE

Scope, method and basis of assessment

What this review is

An evidence-based audit of a research artefact against five tests: whether the framework is complete, internally consistent, technically sound, practically implementable and enterprise-ready.

Claims are assessed only against what the authors assert. Reviewer opinion is labelled as commentary and separated from source fact.

The authors' explicit exclusions are noted as scope, not counted as failure.

Primary sources reviewed

- Assessment framework paper - arXiv:2606.13425 (11 Jun 2026)
- Intent-based API design paper - arXiv:2606.13445 (11 Jun 2026)
- Eurocrypt 2026 / MAgiCS workshop paper (combined)
- IBM Research, “It’s time for cryptography to get its own abstraction layer” (17 Jul 2026)
- Reference API specification - agile-crypto/api (Protocol Buffers)
- Context: NIST CSWP 39, ETSI TR 104 016, agility systematic review, Software-Defined Cryptography

REVIEWER INDEPENDENCE *This review reflects the independent technical analysis of SITG-Consulting. Any advisory services offered by SITG-Consulting are separate from the findings presented here.*

Overall assessment and scorecard

OVERALL - ENTERPRISE-WEIGHTED

7.2 / 10

Conditional adopt

8.5 Application-level contribution

5.5 Enterprise readiness (largely out of scope)

On the problem it set itself, the paper scores 8.5. The composite is weighted for an enterprise audience and pulled down only by dimensions the authors place outside scope.

8.5 RESEARCH QUALITY	**8.0** TECHNICAL DEPTH
7.5 INNOVATION	**8.0** ARCHITECTURE
8.0 API DESIGN	**6.0** PRACTICALITY
5.0 GOVERNANCE · OUT OF SCOPE	**5.5** ENTERPRISE READINESS · OUT OF SCOPE

EXECUTIVE SUMMARY

Key strengths, key weaknesses, verdict

Key strengths

- A rigorous decomposition of cryptographic agility into seven orthogonal, assessable dimensions
- Clean separation of intent from algorithm; migration reframed as an operational act
- Concrete realisation in Protocol Buffers with a public reference specification

Limitations, classified

- Out of scope: enterprise governance, inventory and CBOM sit beyond the stated API boundary
- Implementation: the reference implementation is explicitly partial; evaluation is qualitative
- Future work: a brownfield path for existing call sites is not yet demonstrated

Final verdict

- Two distinct problems - IBM solves one of them exceptionally well
- Application-level agility: adopt as the target API contract for new and refactored services
- Enterprise transformation: a separate challenge, addressed beyond the paper’s scope

CONTEXT

About the authors and the work

AUTHORS & VENUE

Navaneeth Rameshan · Grégoire Messmer

IBM Research Europe, Zurich, Switzerland

Two companion papers submitted 11 Jun 2026:

arXiv:2606.13425 - assessment framework

arXiv:2606.13445 - intent-based API design

Reference spec published as agile-crypto/api (Protocol Buffers).

Eurocrypt 2026 / MAgiCS

Presented at the Migration and Agility in Cryptographic Systems workshop, co-located with Eurocrypt 2026 (Rome, 10 May 2026); proceedings via Springer CCIS.

Earlier airing

Ideas first shared at the PKI Consortium conference, Kuala Lumpur, November 2025 - practitioner feedback shaped the framing.

IBM Quantum Safe

Distinct from IBM's existing Quantum Safe platform (inventory, discovery, CBOM). This work sits one layer down, at the application API.

MOTIVATION

Why this paper matters

Every organisation that expects to survive multiple cryptographic transitions has exactly two choices:

Redesign its applications once. → **Rewrite** them every decade.

2030
112-bit classical algorithms deprecated by NIST

2035
The same algorithms disallowed entirely

FIPS 203 / 204 / 205
ML-KEM, ML-DSA and SLH-DSA standardised (2024)

Architectural, not cryptographic

The hard part is not choosing ML-DSA; it is that algorithm names are compiled across thousands of call sites.

Technical debt compounds

Every hard-coded identifier is a deferred migration cost accruing silently across the portfolio.

Transitions repeat

Each future break - a weakness, a deprecation, a deadline - repeats the same code-change grind.

Legacy APIs lack agility

Decades of progress still govern who may use cryptography, not which algorithms run.

POSITIONING

Where IBM fits in the stack

Enterprise governance	Board accountability & risk	STILL OPEN
Transformation programme	Prioritisation, planning, assurance	
Discovery	Find where cryptography is used	
CBOM	Cryptographic bill of materials	
Policy	Which algorithms are permitted	
IBM framework	Intent-based API · key evolution	SOLVED
Application	Business logic, by intent only	
Cryptographic provider	HSM, library or service	

READING THE STACK

IBM's framework occupies one layer - the interface between application code and the cryptographic provider.

It gives that layer a policy hook above it and a stable key identity below it. That is a real, well-defined contribution - and a deliberately bounded one.

The layers above - discovery, CBOM, programme governance, board accountability - are where enterprise cryptographic transformation actually lives. The framework does not claim them.

This one layer - solved.
Everything above it - still open.

PROBLEM STATEMENT

What problem is IBM trying to solve?

THE DIAGNOSIS

"Algorithm transition is largely a software engineering problem."

Intent-Based Cryptographic API Design (arXiv:2606.13445)

If transition is a software problem, the fix lives in API design - not in choosing the next cipher.

1 **Hard-coded algorithms** APIs expect explicit, named algorithms at every call site.

2 **Tight coupling** Application code knows the algorithm, its parameters and often the provider.

3 **Migration as code change** Swapping an algorithm means finding, editing, testing and redeploying source.

4 **No policy selection** Little or no support for choosing algorithms centrally by policy.

5 **No key transformation** No first-class path to move an existing key to a new algorithm.

FRAMEWORK OVERVIEW

Four contributions, one coherent argument

01

Assessment framework

Decomposes application-level agility into seven orthogonal dimensions - a non-linear profile, not a maturity ladder.

02

Design principles

Thirteen API design principles derived from five foundational architectural properties.

03

Concrete realisation

Demonstrated through Protocol Buffers API patterns and a public reference specification.

04

Empirical evaluation

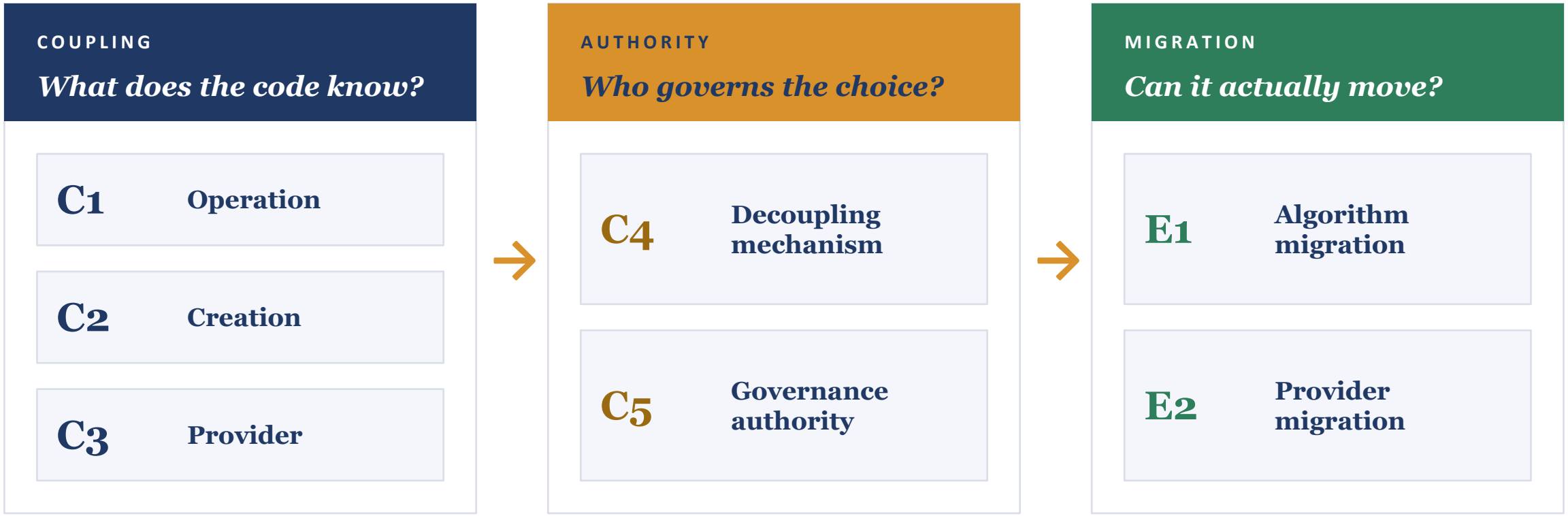
Six representative systems assessed, exposing three pervasive and independent gaps.

REVIEWER COMMENTARY Why this matters: most agility guidance is a wish-list of aspirations. Here requirement, mechanism and evidence reinforce one another, which is precisely what lets “agility” be assessed rather than asserted. The chain of reasoning holds; the open question is reach beyond the API boundary.

ASSESSMENT FRAMEWORK

Seven orthogonal dimensions

Read as a chain of questions, not a maturity ladder - a system can score well on one link and badly on the next.



TIERS *C1-C5 form the paper's Decoupling Assessment tier; E1-E2 form the Enabler Assessment tier.*

ASSESSMENT FRAMEWORK

How the dimensions are read

Dimension	What good looks like	What poor looks like
Operation / creation coupling	Algorithm never appears at the call site or at key creation	Algorithm and parameters named explicitly in application code
Provider coupling	Provider is selectable at runtime, invisible to callers	Provider compiled in; no runtime substitution
Decoupling mechanism	Choices resolved by an externalised policy engine	Choices hard-coded as literals in source
Governance authority	A governance function decides which algorithms are permitted	Only access control - who may call, not what runs
Agility enablers	First-class transform and migrate on existing keys	Re-key by hand; rewrite call sites

REVIEWER COMMENTARY Why this matters: orthogonality kills the single-number agility “score”. A system can rate high on operation decoupling yet low on creation decoupling, so one figure hides the gap that will actually block migration. Google Tink proves the point - it hides algorithm details almost completely, yet those algorithms are compiled in and no alternative provider can be loaded at runtime.

Five architectural properties

Abstraction	Callers express intent, not mechanism; the algorithm is never named at the call site.	***Decouples code from cryptographic choices***
Stability	Keys carry stable logical identifiers; identity survives rotation and algorithm change.	***Call sites keep working across change***
Temporal flexibility	Cryptographic decisions can be made or changed at any point in the application lifecycle, not fixed once at build time.	***Decisions stay revisable after deployment***
Separation	Intent, policy, algorithm and provider are distinct concerns with clean boundaries.	***Each can change independently***
Extensibility	New algorithms enter the catalogue as data, not as API or client changes.	***PQC additions do not break clients***

IBM derives thirteen API design principles from these five properties *- the properties are the reasoning; the principles are the checklist. Done.*

INTENT-BASED CRYPTOGRAPHY

From “how” to “what”: the model

Scope

Typed intent: authenticated encryption, signature, KEM, KDF, MAC…

→

Policy

Abstract, opaque governance object decides permitted templates

→

Template

Concrete config (e.g. ml-dsa-65) with strength, FIPS, quantum-safety

→

Named key

Stable logical name; key material never exposed to the caller

THE CALL SITE

ILLUSTRATIVE (SIMPLIFIED)

```
CreateKeyRequest {
  name: "contract-signing-key",
  policy: "prod-signing",
  scope: SIGNATURE_STANDARD }
SignRequest { key_name: "contract-signing-key", … }
```

KEY-EVOLUTION ENABLERS

TransformKey → new algorithm **MigrateKey** → new provider

Identifier is preserved, so every existing call site keeps working - no code change.

No algorithm. No provider. No key material. The caller states intent - the platform decides the rest.

Advantages, limits and adoption barriers

Advantages

- Migration becomes an operational act, not a code change
- Governance gains a real control point - which algorithms, not just who
- PQC additions arrive as catalogue data, transparent to clients

Limitations

- Assumes callers adopt a new API surface - brownfield reach is unproven
- Correctness shifts into policy and catalogue - new failure and audit surface
- Reported “six-fold” agility gain is a secondary-source claim, not independently reproduced

Adoption barriers

- Requires a policy authority and catalogue owner that many organisations lack
- A stable-identity key store must exist and be trusted enterprise-wide
- Value depends on breadth of adoption across services

EVALUATION

Six systems assessed - three pervasive gaps

System	Algorithm abstraction	Provider swap-in	Intent-based key creation	Policy-driven selection	Key transformation
PKCS#11	Partial	Strong	Absent	Absent	Absent
OpenSSL 3.0	Partial	Strong	Absent	Absent	Absent
Java Cryptography Arch.	Partial	Strong	Absent	Absent	Absent
Google Tink	Strong	Absent	Absent	Absent	Absent
AWS KMS	Partial	Strong	Absent	Absent	Absent
HashiCorp Vault Transit	Strong	Strong	Absent	Absent	Absent

■ Strong ■ Partial ■ Absent *The three right-hand columns are red for every system - the gaps are structural, not incidental.*

Orthogonality made visible: Tink hides the algorithm beautifully yet cannot swap providers; PKCS#11, OpenSSL, JCA and AWS KMS swap providers freely yet still expose the algorithm. Vault Transit is the only system that does both. None closes the three agility gaps.

Reviewer note *IBM's abstract states that "most" systems lack the ability to transform existing keys; the detailed evaluation scores all six as absent at the exposed API - a difference of wording rather than of assessment.*

EVALUATION - CRITIQUE

Is the comparison fair?

Why these six were chosen

The set spans the field a practitioner actually meets: a hardware standard (PKCS#11), a ubiquitous library (OpenSSL), a platform SDK (JCA), a misuse-resistant library (Tink), and two managed services (AWS KMS, Vault Transit).

That breadth is a strength - the gaps recur across very different design philosophies, which supports the claim that they are structural rather than incidental.

Reviewer caveats

- Assessment is qualitative. Dimensions are scored by expert reading, not a reproducible instrument or public rubric.
- Managed services are judged at their API surface; back-end capabilities they do not expose are, reasonably, out of view.
- Version-bound: OpenSSL 3.0 providers and KMS features evolve; findings are a 2026 snapshot.
- No quantitative migration cost is measured; the argument rests on capability presence, not effort saved.

ASSESSMENT

Where the framework succeeds

Conceptual clarity

Replaces a contested buzzword with an assessable decomposition. “Agility” becomes measurable.

Technical originality

Orthogonal, non-hierarchical dimensions capture profiles that linear maturity models cannot express.

Architecture

Clean separation of intent, policy, algorithm and provider - each concern can change independently.

API design

Stable key identities plus transform/migrate turn a rewrite into an operation.

Future-proofing

Extensibility by catalogue data means PQC - and whatever follows - lands without client changes.

Engineering value

A public Protocol Buffers reference gives the field something to test and standardise around.

ASSESSMENT

Where it falls short - an honest scope map

Intentionally out of scope

- Cryptographic discovery & inventory
- CBOM generation
- Dependency mapping
- Migration prioritisation
- Programme governance
- Business case & economics

Partially addressed

- Policy-driven selection (hook, not engine)
- Batch policy impact evaluation
- Provider migration path
- Audit trail via evolution history
- Third-party / supplier crypto

Enterprise responsibilities outside framework scope

- Board accountability & risk ownership
- Procurement & supplier governance
- Assurance, compliance evidence
- Lifecycle management at scale
- Transformation planning

Reviewer commentary: the authors are explicit about the API boundary. The risk is not that they overreach - it is that adopters mistake an API contract for a whole agility programme.

POSITIONING

Comparison with adjacent work

Approach	Focus	How IBM's framework relates
NIST CSWP 39	Strategies & practices for crypto-agility across systems	IBM operationalises at the API layer what NIST frames at policy level
ETSI TR 104 016	Repeatable framework for quantum-safe migration	Complementary: ETSI governs the programme; IBM the code interface
Agility systematic review (2024)	Clarifies a vague term across the literature	IBM answers the call with a concrete, assessable decomposition
Software-Defined Cryptography	Central governance & automated policy enforcement	Shared intent; IBM adds a principled API and key-evolution model
Enterprise transformation governance (CTM-class)	Discovery, CBOM & programme orchestration across the whole estate	Operates above IBM's boundary: the framework lives inside applications; these tools govern everything around them

REVIEWER COMMENTARY Why this is significant: IBM's distinctive move is technical, not rhetorical. First-class transform and migrate operations on stable key identities turn migration from a source-code change into a runtime event - the step NIST, ETSI and the SDC literature call for but stop short of specifying. The estate-wide transformation layer (highlighted) still sits above the framework, not inside it.

ADOPTION

Enterprise applicability

Adopt now

New services, platform teams and internal crypto libraries able to define an API contract and run a policy authority.

Approach with care

Large brownfield estates: value is real but gated on refactoring call sites and standing up a key-identity store.

Not yet

Organisations without cryptographic inventory or governance ownership - foundations must come first.

PREREQUISITES FOR VALUE

Organisational maturity

A named cryptographic-governance owner distinct from access control.

Implementation complexity

A policy engine, an algorithm catalogue and a stable-identity key store.

Skills required

Cryptographic engineering plus platform and policy-as-code capability.

Likely barrier

Ownership, not technology: someone must hold the policy and the catalogue.

INDEPENDENT SCORING

Scores with justification

8.5	**Research quality**	Clear problem, sound method, credible venue and reference artefact.
8.0	**Technical depth**	Orthogonal model and key-evolution semantics are well reasoned.
7.5	**Innovation**	Novel decomposition; underlying governance goal is shared with prior work.
8.0	**Architecture**	Clean separation of intent, policy, algorithm and provider.
8.0	**API design**	Stable identities and transform/migrate are the strongest ideas.
6.0	**Practicality**	Reference implementation is explicitly partial; brownfield path unproven.
5.5	**Enterprise readiness**	API-level only; inventory, CBOM and programme governance excluded.
5.0	**Governance**	Provides a policy hook, not an organisational governance model.
7.5	**Evidence**	Six diverse systems, but qualitative and version-bound.
6.5	**Completeness**	Complete at the API boundary; silent beyond it.
8.5	**Clarity**	Precise, well-structured, honest about scope.
8.5	**Future relevance**	Directly aligned with the multi-decade PQC transition.

HOW THE 7.2 IS DERIVED A calibrated reviewer judgement, not a flat average (which would read 7.3). The headline deliberately gives greater weight to the dimensions an enterprise audience depends on - practicality, enterprise readiness and governance - holding the composite just below the mean at **7.2 / 10**.

CONCLUSION

Final verdict and recommendations

VERDICT

7.2 / 10

Recommended, conditionally.

Application agility - solved. Enterprise transformation - a separate challenge.

One of the strongest application-level treatments of cryptographic agility published to date - internally consistent, technically well argued and practically valuable. It deliberately stops at the application boundary; enterprise resilience still requires governance, discovery and a transformation programme around it.

Who should implement it

Platform and security-engineering teams building new services or refactoring internal crypto libraries, backed by a governance owner.

What must be added first

Discovery and CBOM, a policy authority, a stable-identity key store, and migration-programme governance around the API.

Future research required

A reproducible scoring instrument, quantified migration-cost evidence, a brownfield adoption path, and scalability of batch policy evaluation.

FORENSIC CONCLUSION

Final independent assessment

OVERALL CONCLUSION

Based on the published literature reviewed, this represents the strongest application-level cryptographic agility framework identified during this assessment. Within its stated scope it is internally consistent, technically well argued and more complete than any comparable published work examined here.

Relationship to NIST and standards

IBM complements the published standards rather than replacing them. NIST articulates why cryptographic agility is required; IBM contributes one of the first comprehensive engineering architectures showing how application-level agility can be implemented. The two are complementary, not competing.

The enterprise context

The paper solves agility at the application layer. Enterprise transformation still depends on discovery, cryptographic inventory, a CBOM, governance, assurance and board accountability, together with transformation management, all of which sit outside the paper's stated scope.

Where the field must go next

Progress now depends on standardising intent-based cryptographic interfaces and interoperable policy models, integrating them with CBOM and cryptographic discovery, establishing reproducible measurement of cryptographic agility, and layering enterprise governance around application-level agility.

FINAL STATEMENT

IBM has solved an important engineering problem. The industry must now build the governance, operational and assurance capabilities needed to transform application-level cryptographic agility into enterprise cryptographic resilience.

FROM APPLICATION AGILITY TO ENTERPRISE RESILIENCE

Applying the framework in practice

Adopting the API contract is the first step. Translating application-level agility into enterprise resilience requires the following capabilities - none of which the paper claims to provide, and each of which sits outside its stated scope:

01 Discover

- Cryptographic discovery
- Inventory of keys & algorithms
- CBOM - cryptographic bill of materials
- Dependency mapping

02 Govern

- Policy & governance authority
- Migration programme management
- Prioritisation & sequencing

03 Assure

- Board-level reporting
- Assurance & compliance evidence
- Ongoing posture monitoring

Independent validation and assurance available from SITG-Consulting. brian.couzens@sitg-consulting.com · www.sitg-consulting.com

APPENDIX

References - primary and supporting sources

Primary sources

- Rameshan & Messmer, An Assessment Framework for Application-Level Cryptographic Agility - arXiv:2606.13425 (2026)
- Rameshan & Messmer, Intent-Based Cryptographic API Design for Cryptographic Agility - arXiv:2606.13445 (2026)
- Cryptographic Agility for Applications: An Assessment Framework and Principled API Design - Eurocrypt 2026 / MAgiCS workshop
- IBM Research, “It’s time for cryptography to get its own abstraction layer” (17 Jul 2026)
- agile-crypto/api - abstract cryptographic API specification (Protocol Buffers)

Supporting & contextual

- Näther et al., Toward a Common Understanding of Cryptographic Agility - arXiv:2411.08781 (2024/25)
- Cho et al., Software-Defined Cryptography - arXiv:2404.01808 (2024)
- NIST CSWP 39, Considerations for Achieving Cryptographic Agility (2025)
- ETSI TR 104 016, A Repeatable Framework for Quantum-Safe Migrations (2024)
- NIST FIPS 203 / 204 / 205 - ML-KEM, ML-DSA, SLH-DSA (2024)

Independent review prepared 20 July 2026. Reviewer commentary reflects the author's analysis, not the position of IBM Research.