

SoK: Structural Cryptographic Attack Surface and Algorithm Longevity: A Secure-by-Design Framework for Long-Lived Cryptographic Systems

Abstract—Long-lived secure systems may remain operational beyond the period for which confidence in their original cryptographic dependencies can responsibly be projected. Conventional evaluation measures security strength, known attacks, implementation quality, and performance, but says less about the structural commitments that determine how assurance can erode or be restored over time.

This Systematization of Knowledge paper introduces *Structural Cryptographic Attack Surface* (SCAS), a qualitative framework for identifying and comparing security-relevant assumptions, state, composition, implementation behaviour, interfaces, and lifecycle dependencies. SCAS characterises their coupling, observability, failure propagation, mutability, and replaceability, and introduces *minimum necessary structure* (min-structure) as a requirements-relative Secure-by-Design principle.

The framework also distinguishes present security confidence from residual cryptographic confidence horizon, relates migration margin to Mosca’s migration inequality, and formalises the Cryptanalytic Visibility Gap between public and potentially non-public cryptanalytic capability. Recent AI-assisted cryptanalysis motivates treating the future rate and accessibility of discovery as an architectural uncertainty rather than a predictable constant.

Three case studies exercise the framework across progressive cryptanalytic erosion (SHA-1), interface-coupled exploitation (RSA/JNT), and prospective post-quantum transition design (ML-KEM-768 versus X25519MLKEM768). The resulting method supports long-lived architectures that make cryptographic dependencies explicit, minimise unnecessary structural commitment, contain failure, and preserve deliberate replacement paths as assumptions, standards, and threats evolve.

Index Terms—cryptographic attack surface, cryptographic longevity, crypto-agility, post-quantum cryptography, security architecture, long-lived systems, algorithm migration, cryptographic assurance, minimum necessary structure

1. Introduction

Cryptographic strength is commonly expressed as a hardness metric in bits: an estimate of the computational work required to defeat a cryptographic mechanism under the best applicable attack model [4]. This abstraction is indispensable, but it answers a particular question—how difficult the construction appears to attack under current knowledge and assumptions. It says much less about how

many security-relevant structures, interfaces, states, and life-cycle dependencies must continue to behave as expected throughout the operational life of the system.

Long-lived secure systems can outlive the confidence horizon of the cryptography with which they were deployed. Satellite communications, avionics, industrial control, telecommunications, energy, identity, and defence platforms may remain in service for decades while cryptanalytic knowledge, standards, implementation techniques, and available computing capabilities continue to change.

The post-quantum transition makes this problem explicit. NIST directs organisations to migrate from quantum-vulnerable public-key mechanisms while also treating crypto-agility—replacement across protocols, software, hardware, firmware, and infrastructure—as an engineering objective [1], [2]. Present approval therefore does not imply that an algorithm is a suitable immutable dependency for the complete lifetime of a future system. A construction can be secure at deployment and still be a poor long-term architectural commitment.

IRIS², the European Union’s long-lived secure-connectivity programme, illustrates the scale of the problem without being analysed here [3]. Conventional measures of cryptographic strength remain essential; SCAS addresses the surrounding architectural question: *what must continue to remain true for a construction to remain secure, and what is required if one of those conditions ceases to hold?*

1.1. A Changed Cryptographic Confidence Horizon

NIST defines *algorithm security lifetime* as the period during which data protected by an algorithm is expected to remain secure [4]. The historical record does not show a simple monotonic shortening of algorithm lifetimes; the more useful architectural quantity is the *residual cryptographic confidence horizon*: the remaining interval over which a dependency can reasonably be planned to remain acceptable at design time.

Where a credible retirement or deprecation horizon T_r exists,

$$H_c(t) = T_r - t, \quad (1)$$

with T_r interpreted as a policy or engineering planning horizon rather than a predicted break date. This exposes cases in which the expected system lifetime extends beyond the period for which a dependency can responsibly be assumed to remain acceptable.

Migration time can consume that horizon. NIST notes that major cryptographic transitions can require 10–20 years from standardisation to broad integration [22]. For estimated replacement time T_m , define

$$M(t) = H_c(t) - T_m. \quad (2)$$

When $M(t) \leq 0$, migration consumes the complete remaining planning horizon even if the algorithm remains secure today. These equations are planning notation, not security derivations; they make lifecycle assumptions explicit.

1.2. Relation to Mosca’s Migration Inequality

The migration-margin formulation deliberately reuses the timing logic of Mosca’s quantum-risk inequality. Mosca defines x as information security shelf-life, y as migration time, and z as the time until the deployed cryptography can be broken; $x + y > z$ indicates that risk already exists [21].

SCAS changes the object being managed. Mosca’s z is a collapse horizon for the cryptography protecting data, whereas $H_c(t)$ is a residual *confidence* horizon for any cryptographic dependency. That horizon can close before a practical break because of standards withdrawal, parameter deprecation, implementation or protocol findings, policy change, or increased assurance requirements. $M(t)$ therefore asks whether a dependency can be replaced before continued reliance becomes unacceptable.

Where data shelf-life matters, let T_s be the required remaining protection lifetime and define

$$M_s(t) = H_c(t) - (T_m + T_s). \quad (3)$$

Then $M_s(t) \leq 0$ maps directly to Mosca’s condition when T_s , T_m , and H_c correspond to x , y , and z . The distinction is architectural: SCAS generalises the timing question from data-protection collapse to the acceptable lifetime of any cryptographic dependency. Crypto-agility reduces lock-in, but its negotiation, compatibility, and migration mechanisms can add structure of their own [2].

1.3. AI-Assisted Cryptanalysis and Diffusion of Capability

A second uncertainty concerns the cryptanalytic process itself. In 2026, Anthropic reported an AI-assisted mathematical weakness in HAWK, then a NIST additional-signature candidate, together with a new reduced-round AES attack; HAWK was subsequently withdrawn. Neither result broke a deployed NIST PQC standard or full-round AES [23], [24]. CryptanalysisBench likewise reports frontier models reproducing known attacks and producing apparently novel results against selected research targets [25].

The relevant architectural point is not imminent failure of standardised cryptography but reduced predictability of future discovery. Literature synthesis, hypothesis generation, code construction, parameter exploration, and experimental iteration can increasingly be machine-assisted, lowering

the cost of sophisticated exploratory work. Broader cyber benchmarks also show rapid changes in frontier-model capability, although they should not be interpreted as a cryptanalytic growth law [26].

Consequently, long-lived architectures cannot confidently extrapolate today’s cryptanalytic workflow across five-, ten-, or twenty-year horizons. SCAS treats that uncertainty as a reason to minimise unnecessary structural opportunity and preserve containment and replacement paths, rather than as evidence that any particular deployed algorithm is already compromised.

1.4. The Cryptanalytic Visibility Gap

Security engineering observes the public cryptanalytic frontier, not necessarily the complete frontier. Let $\mathcal{C}_{\text{pub}}(t)$ be the capabilities publicly demonstrated by time t and $\mathcal{C}_{\text{actual}}(t)$ those that actually exist across all actors. Then

$$\mathcal{C}_{\text{pub}}(t) \subseteq \mathcal{C}_{\text{actual}}(t), \quad (4)$$

and the unobservable set difference

$$\mathcal{G}(t) = \mathcal{C}_{\text{actual}}(t) \setminus \mathcal{C}_{\text{pub}}(t) \quad (5)$$

is the *Cryptanalytic Visibility Gap* (CVG). The notation records an information boundary; it does not estimate the gap’s magnitude.

History provides precedent for important cryptographic capability remaining non-public: GCHQ’s work by Ellis, Cocks, and Williamson preceded the open publication of related public-key ideas by years [27]. Responsible disclosure also intentionally separates discovery from publication; the HAWK result was coordinated with its designers before release [23]. Commercial, defensive, intelligence, and national-security incentives can create similar asymmetry without implying any secret break of a standardised algorithm.

The engineering consequence is modest but important: absence of a published attack characterises public knowledge, not the totality of possible capability. The same applies to future quantum capability, whose first possession need not coincide with public confirmation. SCAS therefore treats discovery, capability, and disclosure uncertainty as lifecycle constraints: presently unexploited structure is neither labelled a vulnerability nor assigned zero relevance merely because no public attack exists.

1.5. Research Questions and Contributions

This paper investigates four questions:

- 1) Can structural cryptographic attack surface be described consistently across substantially different classes of cryptographic construction?
- 2) Which forms of mathematical, stateful, compositional, implementation, interface, and lifecycle structure create distinct security dependencies?
- 3) Can retrospective case studies show that SCAS identifies dependencies relevant to the later erosion,

retirement, or replacement of a construction without relying on those outcomes to define the framework?

- 4) Can the resulting structural profile provide useful architectural information for systems whose operational lifetime may exceed the confidence horizon of individual cryptographic dependencies?

As a Systematization of Knowledge contribution, the paper brings together results and practices that are usually considered separately—cryptanalytic hardness, attack surface, crypto-agility, algorithm transition, implementation exposure, and lifecycle engineering—and proposes a common structural lens for reasoning across them. Specifically, it contributes: (i) the SCAS vocabulary and an eight-stage comparative procedure; (ii) residual cryptographic confidence horizon and a Mosca-related migration margin that generalises lifecycle timing from data-protection collapse to arbitrary cryptographic dependencies; (iii) the Cryptanalytic Visibility Gap (CVG) as an explicit epistemic limitation on what can be inferred from public cryptanalysis; (iv) minimum necessary structure (min-structure) as a requirements-relative, partial-order Secure-by-Design principle; and (v) three case studies spanning progressive cryptanalytic erosion, interface-coupled structural exploitation, and prospective comparison of contemporary key-establishment constructions. The lifecycle and visibility concepts are usable independently of the complete SCAS procedure, while the case studies illustrate how they combine in architectural analysis.

2. Background and Related Concepts

2.1. Algorithm Security Lifetime

Algorithm security lifetime concerns the period over which protection provided by an algorithm can reasonably be expected to remain effective [4]. This differs from both the lifetime of a particular key and the operational lifetime of a system. A system may therefore outlive multiple algorithm or key-management generations even when its physical platform remains unchanged.

For SCAS, the distinction matters because algorithm lifetime is commonly treated as a property to be estimated, while system longevity is an architectural constraint to be managed. The framework is concerned primarily with the latter: if an assumption ceases to be acceptable, what else must change before assurance can be restored?

2.2. Cryptographic Agility

NIST defines crypto-agility in terms of replacing and adapting cryptographic algorithms across protocols, applications, software, hardware, firmware, and infrastructure while preserving security and ongoing operations [2]. This definition already implies that algorithm replacement is not purely a primitive-level operation; it can cross multiple implementation and governance layers.

SCAS treats agility as related to, but distinct from, structural attack surface. A design can support multiple algorithms and still be difficult to migrate if identifiers, certificates, wire formats, hardware accelerators, or certification profiles are tightly coupled to one choice. Conversely, mechanisms introduced to enable flexibility may create downgrade paths, compatibility states, or configuration dependencies that must themselves be secured. The objective is therefore not maximal configurability, but controlled replaceability with minimal additional exposure.

2.3. Conventional Attack Surface

In conventional system security, an attack surface generally refers to the set of interfaces, entry points, services, or behaviours through which an adversary can interact with a target. That model is necessary but incomplete for the question addressed here. Cryptographic security may depend on structures that are not externally callable at all: an algebraic relation, a distributional assumption, a state invariant, a nonce rule, or a parameter relationship may become exploitable only when combined with another element of the system.

SCAS therefore extends the attack-surface idea inward by identifying the structurally distinct conditions that carry security assurance and asking whether their failure or interaction creates a meaningful path to loss of a claimed security property.

3. Structural Cryptographic Attack Surface

3.1. Definition

For the purposes of this paper, *Structural Cryptographic Attack Surface* (SCAS) is defined as the set of structurally distinct assumptions, transformations, states, interfaces, parameter choices, and lifecycle dependencies whose compromise, misuse, degradation, or cryptanalytic failure can materially reduce a cryptographic construction’s intended security properties.

The definition is deliberately broader than conventional software attack surface. A software attack surface is usually concerned with externally reachable interfaces or behaviours through which an adversary may interact with a system. SCAS also includes internal mathematical and operational dependencies that may not be directly exposed as interfaces, but on which the security argument nevertheless relies. An algebraic relationship, nonce-generation rule, state transition, parameter constraint, encoding assumption, or replacement mechanism may therefore form part of the SCAS even when it is not directly invoked by an external caller.

SCAS distinguishes *structure* from *vulnerability*. A structural element identifies a condition that must continue to hold for the claimed security properties to remain valid, without implying that the element is weak, exploitable, or incorrectly designed. This distinction is necessary because useful security properties commonly require

additional structure. Authentication, forward secrecy, compromise recovery, state synchronisation, and cryptographic agility may all increase structural complexity while improving the security of the system as a whole.

3.2. The Unit of Structural Exposure

A central difficulty in any structural comparison is deciding what should count as a distinct element. Counting individual operations would be arbitrary: a cipher with more rounds would automatically appear to have a larger attack surface even when those rounds are repetitions of the same well-understood transformation. Conversely, treating an entire protocol as one element would hide dependencies that fail independently.

SCAS therefore uses a *structural dependency* as its basic unit of analysis. A dependency is included as a distinct SCAS element when all of the following conditions hold:

- 1) it contributes materially to one or more claimed security properties;
- 2) it relies on an identifiable assumption, invariant, state, interface, parameter relationship, or behavioural condition;
- 3) failure, misuse, or invalidation of that condition can be described independently of at least some other elements; and
- 4) the element is sufficiently distinct that analysing it separately provides information that would be lost by aggregating it with its neighbours.

This definition prevents SCAS from becoming a measure of code size or algorithmic verbosity by focusing on independently meaningful places in which security assurance resides.

3.3. Dimensions of Structural Exposure

SCAS groups dependencies into six analytical dimensions: *mathematical and cryptanalytic*; *key, entropy, and state*; *compositional*; *implementation*; *protocol and interface*; and *lifecycle and replacement*. Table 1 gives the operational question for each dimension and a case-study example. The categories are deliberately non-exclusive: one dependency may occupy several dimensions, and cross-dimension interactions are themselves relevant to coupling analysis.

3.4. Qualitative Properties of a SCAS Element

After a structural dependency has been identified, SCAS characterises it using a common set of qualitative properties. These properties are deliberately not converted into a numerical score.

The distinction between *mutability* and *replaceability* is useful in long-lived systems. A parameter may be mutable while the primitive itself remains fixed; similarly, an algorithm may be replaceable in software while a wire protocol, hardware accelerator, certificate profile, or certification regime still prevents practical migration.

3.5. Structural Coupling

Structural coupling occurs when the security significance of one SCAS element depends materially on its interaction with another. Coupling deserves separate attention because the combined attack surface may not be apparent when elements are analysed independently.

SCAS distinguishes at least four forms of coupling. *Functional coupling* occurs when two mechanisms must operate together to provide a security property. *State coupling* occurs when the correctness or secrecy of one component depends on synchronised or shared state elsewhere. *Interface coupling* occurs when an exposed operation changes the adversarial model of an otherwise sound primitive. *Lifecycle coupling* occurs when replacing one component requires co-ordinated changes to other components, protocols, hardware, or operational processes.

Coupling matters because some security properties emerge only from interactions between components and therefore cannot be assessed by inspecting each component in isolation. A new attack path may arise from the interaction of two individually acceptable structures, or the retirement of one component may be blocked by an otherwise unrelated architectural dependency.

3.6. Why SCAS Is Not a Security Score

SCAS intentionally avoids producing an aggregate numerical ranking. Such a score would imply a precision that the underlying evidence does not support. Ten well-understood, weakly coupled dependencies are not necessarily more dangerous than one poorly understood dependency with systemic failure propagation. Likewise, a structural element introduced to provide forward secrecy or compromise recovery should not be treated as harmful merely because it increases the number of mechanisms in the design.

The output of SCAS is therefore a *structural profile*. Two constructions can be compared in terms of where their assurance resides, which dependencies are exposed to adversarial influence, where coupling occurs, what happens when a dependency fails, and how readily affected components can be replaced. The framework is intended to expose differences that conventional bit-security or performance comparisons do not describe, not to replace those measures.

3.7. Minimum Necessary Structure (min-structure)

A qualitative structural profile can support a prescriptive design principle without being converted into a scalar security score. We call this principle *minimum necessary structure* (min-structure): the minimum structural commitment required to satisfy a stated set of security, functional, and lifecycle requirements under a defined deployment and threat model.

Let $\mathcal{R}$ denote those required properties and let $\mathcal{C}_{\mathcal{R}}$ be the set of constructions that satisfy them. For a construction C , let $P(C)$ denote its SCAS profile. We say that construction A *structurally dominates* construction B , written $A \prec_S B$,

TABLE 1. SCAS DIMENSION SUMMARY FOR SYSTEM DESIGNERS.

Dimension	Core focus / question asked	Illustrative example from the case studies
Mathematical & cryptanalytic	Which hardness assumptions, algebraic structures, reductions, or statistical properties must remain resistant to useful exploitation?	RSA algebraic structure and number-field-sieve relationships used by the JNT attack path.
Key, entropy & state	Which keys, randomness sources, nonces, schedules, persistent states, or state transitions must remain correct and protected?	ML-KEM key-generation and encapsulation randomness, together with storage and lifecycle of decapsulation keys.
Compositional	How are primitives combined, ordered, bound, or domain-separated, and what additional assumptions arise from their interaction?	X25519MLKEM768 combines independently derived classical and post-quantum shared-secret contributions.
Implementation	Which security properties depend on timing, memory access, arithmetic representation, fault behaviour, encoding, or hardware characteristics?	Constant-time ML-KEM decapsulation and implementation behaviour across target platforms.
Protocol & interface	Which message formats, APIs, parsers, negotiation paths, error responses, or exposed operations create structural exposure?	Exposure of a raw RSA signing or decryption oracle enabling an attack path unavailable from the public key alone.
Lifecycle & replacement	How tightly is a dependency bound to hardware, firmware, certificates, protocols, standards, and operational processes, and how readily can it be replaced?	SHA-1 persistence in certificate, protocol, and legacy-compatibility ecosystems after confidence had already declined.

TABLE 2. QUALITATIVE PROPERTIES USED TO CHARACTERISE A SCAS ELEMENT.

Property	Question asked
Security role	Which claimed security property depends on this element?
Assumption	What must remain true for that role to hold?
Adversarial control	To what extent can an attacker influence the element or its inputs?
Observability	What information about the element or its behaviour is externally visible?
Coupling	Which other elements must interact with it for security or failure to emerge?
Independence	Does failure remain local, or invalidate assumptions elsewhere?
Failure propagation	How far can compromise or misuse spread through the construction?
Mutability	Can the element be altered without redesigning adjacent components?
Replaceability	Can it be retired while the wider system remains operational?
Assurance maturity	How directly has the relevant assumption or interaction been analysed and tested?

when both satisfy the same requirements and, under the same scope and assumptions, A is no worse on the SCAS properties material to the decision and is strictly preferable on at least one. A strict improvement may, for example, remove an unnecessary dependency, reduce coupling or adversarial exposure, constrain failure propagation, or improve mutability or replaceability without sacrificing a required security property.

A construction C^* is min-structure with respect to $\mathcal{R}$ when no alternative construction satisfying the same requirements structurally dominates it:

$$\nexists C' \in \mathcal{C}_{\mathcal{R}} \text{ such that } C' \prec_S C^*. \quad (6)$$

This is a partial-order concept rather than a global ranking. Several constructions may be min-structure while remaining mutually incomparable because one reduces coupling while another improves replaceability, or because their

security properties differ in ways that cannot be reduced to a common scale. min-structure therefore favours the least justified structural commitment under the stated requirements, which may differ substantially from the design with the fewest components, operations, or lines of code.

The analogy with min-entropy is intentionally limited. Both concepts ask a minimum question under explicit constraints, but min-structure is not an entropy measure and has no probabilistic or logarithmic interpretation. Its purpose is to expose *structural surplus*: dependencies, coupling, or lifecycle commitments for which no corresponding requirement can be identified, or which can be reduced by an otherwise equivalent design. Such surplus is a candidate for removal, isolation, or replacement, subject to revalidation of the required security properties.

The notation above makes requirements-relative comparisons explicit. Section 6.3 applies $\prec_S$ to a concrete post-quantum transition pair and shows why the same additional structure can be surplus under one requirement set and necessary under another.

Dominance first, policy second.. Incomparability is an intended outcome of the partial order. If one design improves replaceability while another reduces adversarial observability, and neither is no worse on all decision-relevant SCAS properties, the framework stops short of manufacturing a structural winner. The engineering decision then moves to an explicit policy layer outside $\prec_S$. Architects should declare mission priorities and hard constraints—for example assurance level, regulatory requirements, observability tolerance, replacement cost, performance, or certification burden—before using them to select among non-dominated constructions. This separates structural evidence from organisational preference: *dominance first, policy second*. Any requirements-relative tie-break is documented as a policy decision rather than presented as a cryptographic fact.

4. SCAS and Algorithm Longevity

Cryptographic algorithms do not possess predetermined expiration dates. Their practical security lifetime can change because of advances in cryptanalysis, changes in computational capability, implementation discoveries, revised threat models, standards decisions, or new operational requirements. SCAS therefore reframes longevity as an assurance and recovery problem: *what must continue to remain true for this construction to remain secure, and what is required if one of those conditions no longer holds?*

We use the term *confidence horizon* to describe the period over which the assumptions supporting a cryptographic dependency can reasonably be projected for a particular system and threat model. The confidence horizon is not a property of an algorithm alone. It is affected by the way the algorithm is implemented, composed, exposed, certified, and embedded in the surrounding architecture. A mature primitive behind a replaceable software boundary may therefore present a different long-term risk from the same primitive fixed in hardware or embedded in a protocol that cannot be migrated independently.

This distinction is particularly relevant to long-lived infrastructure. The objective is not to identify a primitive that can be assumed to remain secure indefinitely, but to ensure that the failure or retirement of one dependency does not force the failure or replacement of the entire system.

5. Evaluation Methodology

5.1. Comparative Procedure

To make the framework repeatable, we apply the same eight-stage procedure to each construction under study.

- 1) **Define scope and security properties.** State the construction, deployment assumptions, attacker model, and security properties being evaluated. This prevents a primitive from being credited or penalised for properties it was never intended to provide.
- 2) **Inventory structural dependencies.** Identify the distinct assumptions, states, interfaces, transformations, and lifecycle dependencies that satisfy the criteria in Section 3.2. Where available, a CBOM or equivalent machine-readable cryptographic inventory can seed this step, while contextual dependencies still require architectural analysis.
- 3) **Classify dimensions.** Map each dependency to one or more of the six SCAS dimensions in Section 3.3, using Table 1 as the quick-reference taxonomy.
- 4) **Characterise properties.** Record the qualitative properties in Table 2, together with the evidence source, stated failure condition, and required replacement or recovery action. Particular attention is given to adversarial control, observability, failure propagation, and replaceability.

- 5) **Identify coupling.** Record pairwise or higher-order interactions that create additional attack, failure, or migration paths not visible from isolated components.
- 6) **Analyse failure and recovery.** For each material dependency, describe what security property is lost if the dependency fails and what changes are required to restore assurance.
- 7) **Compare profiles.** Compare constructions by the location and nature of their structural dependencies. No aggregate score is calculated; differences are reported as qualitative architectural properties.
- 8) **Apply the min-structure test.** Where constructions satisfy the same stated requirements, ask whether one structurally dominates another under Section 3.7. For each dependency or coupling that differs, identify the requirement that justifies it. Incomparable profiles remain incomparable rather than being forced into an artificial ranking.

The procedure records exposure, coupling, and replacement properties because they affect the consequences of future cryptanalytic or operational change. Its partial-order step is used explicitly in Section 6.3.

5.2. Reproducibility and Inter-Rater Workflow

Dependency boundaries are qualitative, so a SCAS profile should be recorded in a form that permits independent reproduction and makes disagreement visible. For each candidate dependency, the evaluator records: (i) a canonical dependency name; (ii) the evidence or specification from which it was identified; (iii) the security property it supports; (iv) the assumption or invariant that must hold; (v) one or more SCAS dimensions; (vi) the qualitative properties in Table 2; (vii) a concrete failure condition; and (viii) the action required to recover or replace the dependency.

A practical inter-rater validation workflow is:

- 1) freeze the scope, requirement set, threat model, and source material;
- 2) have evaluators independently extract and profile dependencies using the same schema;
- 3) normalise only obvious naming differences, without merging substantively distinct failure conditions;
- 4) compare dependency sets, dimension assignments, and qualitative properties; and
- 5) adjudicate disagreements while retaining an audit trail of the original profiles and the reason for any merged or split dependency.

Agreement can be reported separately from security judgement. For example, Jaccard similarity between independently identified dependency sets can describe coverage convergence, while dimension-label and property agreement can be reported per matched dependency. Before strict set comparison, semantically equivalent labels should be manually or semantically normalised—for example, “randomness failure” and “entropy-source breakdown”—so that

synonymy does not artificially depress measured agreement. These are reproducibility measures rather than security scores. This paper specifies the workflow but does not claim an empirical inter-rater coefficient; a multi-assessor validation study is therefore an appropriate next test of the framework.

5.3. Retrospective Analysis

The retrospective analysis applies the framework to constructions whose security status or deployment guidance changed materially over time. Historical examples are useful because the eventual failure or retirement mechanism is known; however, they also create a risk of hindsight bias. To reduce that risk, the SCAS dimensions and qualitative properties are fixed before individual case studies are analysed. The case study is then asked whether the framework would have made the relevant dependency visible, not whether it would have predicted the eventual attack.

Candidate cases include algorithm retirement driven by changing computational feasibility, collision resistance, protocol composition, oracle exposure, and legacy compatibility mechanisms. The objective is to test whether SCAS captures structurally different reasons for loss of assurance without redefining the framework for each example.

5.4. Prospective Analysis

The prospective analysis applies the same procedure to contemporary constructions for which no comparable failure is known. This is the more important test of whether SCAS is a useful architectural framework rather than a retrospective explanation tool. Candidate classes include symmetric AEAD, public-key and post-quantum key establishment, hybrid KEM constructions, and stateful or ratcheted key-management architectures.

Prospective comparison maps materially different dependency profiles and asks whether those differences would affect the ability of a long-lived system to contain, replace, or recover from a future cryptographic change. It deliberately avoids forecasting which construction might fail first.

6. Case Studies

6.1. Case Study I: Historical Algorithm Retirement—SHA-1

SHA-1 provides a useful retrospective case because its loss of assurance was progressive rather than abrupt. As a 160-bit hash function, generic collision search suggests a birthday-bound cost of approximately 2^{80} evaluations. In 2005, Wang, Yin, and Yu published a full-round collision attack with estimated complexity below 2^{69} operations, demonstrating that the effective collision resistance of the construction was materially below the generic bound [11]. NIST subsequently began restricting SHA-1 for applications

that depended on collision resistance and encouraged migration to stronger hash functions [14].

The gap between theoretical erosion and operational break was substantial. In 2017, Stevens et al. produced the first public collision for full SHA-1. Their attack required work equivalent to approximately $2^{63.1}$ SHA-1 compression function evaluations and was reported as more than 100,000 times faster than a generic brute-force collision search [12]. In 2020, Leurent and Peyrin demonstrated the first practical chosen-prefix collision on SHA-1, reducing the cost of a more protocol-relevant collision class to approximately $2^{63.4}$ SHA-1 equivalents and showing a concrete PGP/GnuPG impersonation application [13]. NIST has since set a deadline to transition away from SHA-1 for all remaining applications by the end of 2030 [14].

The SCAS significance of this history lies in a more modest structural dependency than prediction of the specific differential attacks. Applications that required collision resistance depended on the internal compression-function and message-expansion structure continuing to behave sufficiently close to an ideal 160-bit hash. Once cryptanalysis reduced the cost of producing structured differentials, the nominal output length no longer described the complete security margin.

The lifecycle dimension is equally important. SHA-1 remained embedded in certificate, signature, version-control, and compatibility ecosystems after its collision-resistance margin had already been questioned. The resulting exposure was therefore a combination of mathematical erosion and deployment coupling. The primitive did not need to fail in every application for migration to become necessary; applications whose assurance depended specifically on collision resistance inherited the weakening structural assumption, while legacy uses that did not require that property followed different retirement timelines.

SHA-1 therefore illustrates a different SCAS failure path from the RSA oracle case in Section 6.2. The RSA example is dominated by interaction between algebraic structure and an exposed interface. SHA-1 is instead dominated by progressive cryptanalytic erosion of an internal security assumption, amplified by lifecycle coupling and delayed replacement. A single framework should make both profiles visible without treating them as the same kind of failure.

6.2. Case Study II: Structural Coupling and Oracle Exposure

A recent large-scale RSA experiment provides a useful example of the distinction between increased computing power and exploitation of structural attack surface. Shea et al. implemented the 2007 Joux–Naccache–Thomé (JNT) oracle-assisted RSA inversion method at the 1024-bit scale [5], [7]. The attack bypasses both factorisation of the RSA modulus and recovery of the private exponent. Temporary access to a raw, unpadded RSA signing or decryption oracle is combined with RSA’s algebraic structure and number-field-sieve relation finding to obtain a capability that sub-

sequently permits arbitrary RSA signatures to be forged offline.

The distinction from a simple increase in available computing power is concrete. The authors report that their RSA-1024 computation required approximately 1,380 CPU core-years and 2^{32} oracle queries over five calendar months [5]. Most of this work is a one-time precomputation. Once that precomputation is complete, the authors report that an arbitrary signature of choice can be forged offline at a marginal cost of approximately 180 core-years. They also demonstrated the attack using a hardware security module (HSM) as the signing oracle, thereby impersonating the HSM through black-box API interactions without exfiltrating the RSA private key [5]. The accompanying implementation artifact states that the reported computation used no GPUs [6]. The result therefore cannot be characterised simply as applying a larger GPU budget to the conventional RSA factoring problem.

The authors' own security interpretation is stronger than a comparison of raw core-year counts alone. Extrapolating their empirical measurements, they estimate that RSA security in the presence of the relevant signing oracle is approximately 15–30 bits lower than factoring-based security estimates across commonly used 1024–4096-bit RSA parameters, and that even 4096-bit RSA does not appear to reach a 128-bit security level in this attack model [5]. This model-dependent conclusion applies only where the relevant oracle exposure exists. Within that scope, it shows that a security estimate derived from generic factoring can materially overstate assurance when an additional structural path is exposed.

The reason is structural rather than computational in the narrow sense. In the conventional public-key-only model, the practical baseline remains generic factoring of N using GNFS. In the oracle-assisted model, the exposed raw RSA operation changes the mathematical structure available to the attacker, allowing the JNT method to approach special-number-field-sieve (SNFS) rather than GNFS complexity [5], [7]. Hardware determines how quickly an available computation can be executed; structural exposure determines which computation the attacker is able to perform.

The result applies to a specific oracle model rather than to RSA use in general. The attack depends on access to a raw RSA signing or decryption oracle, which ordinary RSA-PSS and PKCS#1 v1.5 signature interfaces do not normally expose. Its relevance to SCAS is narrower and more fundamental: the effective security of a cryptographic construction can depend on interactions between mathematical structure, interface exposure, and protocol use that are not represented by a security estimate derived from the dominant hardness assumption alone. In this case, the relevant attack path arises from the interaction of RSA's algebraic structure, temporary exposure of a raw inversion oracle, and NFS-based relation finding.

The underlying attack path was published in 2007; the 2026 contribution is the first public implementation and large-scale run at a cryptographically meaningful key size [6], [7]. The separation between the one-time precompu-

tation and the lower marginal cost of subsequent offline forgeries adds an important lifecycle dimension: once the precomputation has been completed, the attacker retains a durable capability without continued access to the HSM or private key. The structural opportunity therefore existed for almost two decades before the engineering effort demonstrated it at scale, and a successful exploitation can continue to matter after the original interface exposure has ended. This distinction between structural opportunity, engineering feasibility, and persistent exploitation is directly relevant to cryptographic longevity.

6.3. Case Study III: Post-Quantum Transition—ML-KEM-768 and X25519MLKEM768

The third case study applies SCAS prospectively and, unlike the preceding cases, uses the min-structure relation explicitly. We compare two key-establishment architectures at a common abstraction boundary: C_{PQ} , in which ML-KEM-768 supplies the key-establishment secret behind an otherwise stable interface, and C_H , which combines ML-KEM-768 with X25519 in the manner standardised for TLS 1.3 as X25519MLKEM768 in RFC 10024 [15], [29]. The first construction is an analytical single-KEM baseline rather than a claim that a pure ML-KEM TLS group is standardised. The comparison deliberately abstracts the surrounding authentication and record protection so that the structural difference under examination is the key-establishment mechanism itself.

ML-KEM-768 already carries a non-trivial structural profile. Its assurance rests on module-lattice hardness assumptions for the selected parameters and on the correctness of sampling, pseudorandom expansion, polynomial arithmetic, NTT operations, compression and encoding, decapsulation checks, implicit rejection, and implementation behaviour around secret intermediates and randomness [15], [16]. Those elements are not defects; they are the mechanisms by which the construction obtains its required security and performance properties.

C_H retains those dependencies and adds a second key-establishment family, X25519, together with the additional public and ephemeral values needed to execute it, combination of two shared secrets through the TLS 1.3 derivation path, larger key-exchange messages, and the negotiation and implementation logic needed to support the hybrid group. RFC 10024 specifies a 1216-byte client share for X25519MLKEM768: 1184 bytes for the ML-KEM encapsulation key and 32 bytes for the X25519 share [29]. RFC 9954 states the purpose of hybrid key exchange directly: combine multiple algorithms so that key-establishment security can remain when all but one component algorithm are defeated [28].

The min-structure result depends on the requirement set. Let $\mathcal{R}_{PQ}$ denote a requirement for post-quantum key establishment using ML-KEM-768, with no requirement for a transitional classical hedge. Under this deliberately narrow scope, both constructions can provide the required post-quantum key-establishment function, but C_H introduces

X25519, additional message material, a multi-secret combination path, and associated implementation and lifecycle dependencies for a property not required by $\mathcal{R}_{PQ}$. Assuming equivalent surrounding authentication and implementation quality, the SCAS profile therefore supports

$$C_{PQ} \prec_S C_H \quad \text{with respect to } \mathcal{R}_{PQ}. \quad (7)$$

The added hybrid structure is structural surplus relative to that requirement set.

Now let $\mathcal{R}_H$ denote the stronger requirement of PQ/T hybrid confidentiality: key-establishment security should remain when either component remains secure. This is the objective described for PQ/T hybrid key exchange in RFCs 9954 and 10024 [28], [29]. C_{PQ} does not satisfy $\mathcal{R}_H$ because it contains no independent traditional component. It is therefore not a member of $\mathcal{C}_{\mathcal{R}_H}$, and the dominance relation in Eq. 7 is no longer applicable. Within this two-construction comparison, the extra X25519 and combiner structure in C_H is necessary structure because it implements a required resilience property. No global claim that X25519MLKEM768 is min-structure follows; other hybrid constructions could satisfy the same requirement with different SCAS profiles.

This comparison produces two findings. First, min-structure is explicitly requirements-relative: “fewer mechanisms” is not the rule. A component that is surplus under one requirement can become necessary when the system is required to survive failure of a cryptographic family. Second, hybridisation illustrates the central agility–attack-surface trade-off. It deliberately increases mathematical, implementation, message-format, and lifecycle structure in order to reduce dependence on the continued security of any single key-establishment family. SCAS makes that exchange visible rather than treating either simplicity or hybridisation as intrinsically preferable.

7. Long-Lived Secure Systems

The principal architectural value of SCAS appears when the expected lifetime of the surrounding system is longer than the confidence horizon of one or more of its cryptographic dependencies. In that setting, cryptographic choice cannot be reduced to selecting the strongest currently approved primitive. The system must also preserve a credible path for replacing that primitive while keeping the rest of the infrastructure operational.

This changes the design objective from *algorithm selection* to *cryptographic survivability*. A long-lived system should assume that at least some cryptographic components will change during its service life and should make the boundaries of those changes explicit. SCAS provides a way to identify which dependencies are local, which are coupled across the system, and which would require coordinated hardware, protocol, certification, or operational change if they were retired.

7.1. Cryptography as a Lifecycle Dependency

Cryptographic replaceability exists on a spectrum. A primitive implemented in a maintained software library may be replaced by a controlled software update. A primitive implemented in signed firmware may additionally depend on update mechanisms, rollback policy, and vendor support. Programmable logic can introduce further qualification requirements, while a primitive embedded in fixed silicon may effectively remain immutable for the service life of the device. Above all of these sit protocol profiles, certificate formats, validation regimes, interoperability commitments, and operational procedures that can preserve a cryptographic dependency long after its source code has become replaceable.

For SCAS, these are security-relevant structures because they determine the recovery cost when an assumption changes. The same algorithm therefore has a different lifecycle profile when deployed in a cloud service, a smart meter, an aircraft line-replaceable unit, or a satellite payload. Cryptographic agility must consequently be assessed at the boundary at which change is actually possible, not merely at the API where an algorithm identifier appears.

This observation also creates a useful design rule: the least replaceable component should carry the fewest avoidable cryptographic commitments. Fixed hardware may need acceleration for arithmetic operations, memory protection, or entropy acquisition, but embedding unnecessary algorithm-specific policy or protocol semantics into that layer increases the cost of future migration.

7.2. Key Establishment, Key Lifecycle, and Traffic Protection

A particularly important form of structural minimisation is separation of cryptographic responsibilities. Three functions that are often coupled in practice should be distinguished conceptually: *key establishment*, which creates or transfers shared secret material; *key lifecycle management*, which derives, rotates, scopes, stores, refreshes, and retires working keys; and *traffic protection*, which applies encryption and authentication to data.

Separation provides useful containment only when each composition has an explicit security argument and appropriate domain separation, authentication, and key-derivation rules. If a key-establishment mechanism must be replaced, the key-lifecycle and traffic-protection layers need not necessarily change. If an AEAD primitive is retired, the mechanism that established the underlying secret need not necessarily be redesigned.

This separation is also compatible with heterogeneous sources of shared secret material. A system may obtain fresh secret material from a post-quantum KEM, a pre-positioned secret, a quantum-key-distribution service, or a future mechanism not yet standardised. Treating the source as an explicit interface rather than an implicit property of the whole protocol can reduce lifecycle coupling, while SCAS

still requires the security assumptions of each source and combiner to be recorded.

The goal is therefore not unlimited algorithm interchangeability. It is a small number of well-defined cryptographic boundaries at which substitution can occur without silently changing unrelated security properties.

7.3. Contemporary Infrastructure Examples

The long-lived-system problem is not unique to space infrastructure. Avionics, industrial control, energy networks, telecommunications, identity systems, and large fleets of embedded devices all combine long replacement cycles with interoperability and certification constraints. In each case, a cryptographic primitive can be inexpensive to replace in isolation while remaining expensive to replace in the deployed system because adjacent structures have adopted its key sizes, message formats, state model, hardware assumptions, or validation requirements.

IRIS² is a contemporary illustration of the scale at which this issue arises. The European Commission describes it as a secure multi-orbit connectivity system intended to provide encrypted governmental communications through a constellation of hundreds of satellites [3]. EuroQCI, which combines terrestrial and space-based quantum-communication infrastructure and includes QKD services, is explicitly described by the Commission as an integral part of IRIS² [17]. The resulting environment is therefore expected to combine long-lived space and ground assets with multiple cryptographic technologies and operational domains.

IRIS² is used here only to make the lifecycle question concrete; the paper draws no conclusion about its non-public architecture. A secure infrastructure expected to operate for many years must accommodate changes in standards, algorithms, key sources, and implementation guidance without assuming that every deployed physical component can be replaced at the same rate. The same reasoning applies to any system in which cryptographic change is faster than infrastructure replacement.

8. Discussion

The case studies illustrate why structural exposure and cryptographic weakness must remain separate concepts. SHA-1 lost collision-resistance margin through cryptanalysis of its internal structure. The RSA oracle result arises from an interaction between algebraic structure and an exposed operation. ML-KEM, by contrast, is analysed prospectively: its visible dependencies are not evidence of failure, but the conditions that an architecture must preserve, monitor, and be prepared to replace.

Across these cases, the common architectural concern is commitment. A system commits not only to a hardness assumption, but also to encodings, interfaces, state, randomness, composition rules, implementation behaviours, and migration paths. Some commitments are essential. Others are accidental consequences of how the system was assembled. SCAS is intended to distinguish the two so that unnecessary

commitments can be removed before they become long-term liabilities.

8.1. Architectural Response to an Uncertain Cryptanalytic Frontier

The preceding analysis changes the role of architecture. If the rate of future cryptanalytic discovery cannot be forecast with precision, if capable tools are becoming more widely accessible, and if public disclosure need not coincide with discovery, then long-lived assurance cannot depend solely on extrapolating the best attack known today. Conventional cryptanalysis remains foundational; SCAS addresses the residual architectural uncertainty that remains after that analysis is complete.

SCAS is a design discipline for reducing the architectural consequences of accelerating, increasingly accessible, and potentially non-public cryptanalytic capability; it complements rather than substitutes for cryptanalysis. Four complementary responses follow from the framework:

- 1) **Identify:** SCAS makes security-relevant structure and its dependencies explicit.
- 2) **Minimise:** min-structure removes or avoids structural commitments for which no required security, functional, or lifecycle property can be identified.
- 3) **Contain:** compartmentalisation and limited failure propagation reduce the scope of loss when one dependency is successfully exploited.
- 4) **Replace:** controlled crypto-agility provides a deliberate path to retire a dependency whose assumptions no longer deserve confidence.

This sequence is intentionally independent of any prediction that a specific algorithm will fail. Its value is greatest precisely when the form and timing of a future attack are unknown. Processing power influences when an available attack path becomes practical; structural properties influence which attack paths are available to be discovered and exploited. The purpose of SCAS is therefore not to eliminate future cryptanalysis, but to reduce unnecessary structure available for exploitation and to limit the architectural consequences when previously benign structure becomes security-relevant.

8.2. The Agility–Attack-Surface Trade-off

Crypto-agility is necessary precisely because future cryptographic change cannot be predicted with confidence. NIST defines agility broadly enough to include protocols, applications, software, hardware, firmware, and infrastructure, and explicitly treats its implementation as a problem involving operational trade-offs [2]. Recent systematic work likewise treats cryptographic agility as a broader engineering property than support for a large algorithm set, with complexity itself forming part of the trade-off [19].

SCAS exposes the tension directly. Supporting multiple algorithms can improve replaceability, but protocol negotiation can create downgrade paths; backwards compatibility can preserve obsolete dependencies; generic abstraction

layers can hide assumptions that are not actually interchangeable; and configuration freedom can create invalid combinations. Conversely, removing every alternative can minimise immediate protocol surface while making the selected primitive an irreversible architectural dependency.

The appropriate objective is therefore *controlled substitution*: enough replaceability to avoid lock-in without maximising algorithm multiplicity. A secure agile design constrains which substitutions are permitted, authenticates or fixes the negotiation context where appropriate, removes obsolete choices rather than preserving them indefinitely, and keeps the number of algorithm-dependent boundaries small. Under SCAS, agility is valuable when the lifecycle exposure it removes is greater than the structural exposure introduced by the agility mechanism itself.

8.3. Structural Minimisation as a Secure-by-Design Principle

A practical implication of SCAS is that *unnecessary* security-relevant structure should be treated as an architectural liability. Additional structure may be necessary to provide authentication, forward secrecy, compromise recovery, separation of duties, or crypto-agility. The governing design principle is therefore minimum necessary structure: every structural dependency should have an identified security or lifecycle purpose, and dependencies that provide no commensurate benefit should be removed, isolated, or made replaceable.

This principle is consistent with the long-standing “economy of mechanism” principle of Saltzer and Schroeder, which argues that protection mechanisms should be kept as simple and small as possible because unnecessary mechanism increases the opportunity for design and implementation flaws [8]. SCAS extends that reasoning specifically to cryptographic architecture by treating assumptions, state, interfaces, composition, and lifecycle coupling as forms of structure that require justification rather than as neutral implementation detail.

The same reasoning gives a concrete cryptographic interpretation to modern Secure by Design practice. The UK Government’s current cross-government approach requires security to be incorporated from the outset and maintained through the system lifecycle; Secure by Design principles are now mandatory for in-scope central-government digital services and technical infrastructure [9], [10]. Within that methodology, SCAS suggests that structural minimisation should be an explicit design objective: reduce the number of unnecessary cryptographic dependencies, minimise coupling between those that remain, and preserve deliberate replacement paths where long-term dependence cannot be avoided.

Architectural minimalism is a different objective: removing a necessary mechanism merely to reduce apparent attack surface can reduce security. min-structure, formalised in Section 3.7, instead seeks the least structural commitment that can be justified by the required security, functional, and lifecycle properties. A mechanism that adds necessary authentication, forward secrecy, compromise recovery, or

controlled substitution may move a design *toward* rather than away from min-structure if it removes a larger dependency or failure path. In long-lived systems, simplicity and replaceability must be designed together.

8.4. Limits of Structural Inference

SCAS identifies dependencies and possible avenues through which assurance may erode; exploitability and break timing remain questions for cryptanalysis. The framework also cannot establish the absence of unknown attack paths.

Within those limits, the framework still has an architectural implication. Where two designs provide the same required security properties and lifecycle capabilities, min-structure permits a structural preference only when one design actually dominates the other under the stated SCAS dimensions. Mutually incomparable designs remain outside a universal ordering. Likewise, the Cryptanalytic Visibility Gap formalises an information boundary: absence of public evidence cannot establish equality between public and total capability. Structural minimisation is therefore a design principle and partial-order test, subject to the limits established in Section 3.6.

8.5. Threats to Validity

SCAS is a qualitative framework and is therefore exposed to classification subjectivity. Two evaluators may choose different boundaries for what constitutes a distinct structural dependency, particularly in layered protocols. The dependency criteria in Section 3.2 and the profiling schema and independent-assessor workflow in Section 5.2 make those differences observable and reproducible, but inter-rater agreement remains an empirical question. A multi-assessor study should therefore report both dependency-set convergence and property-level disagreements rather than assuming a single canonical profile.

Historical case studies create a second risk: hindsight bias. Once an attack is known, it is easy to redescribe the affected mechanism as an obvious dependency. This paper mitigates that risk by fixing the dimensions and qualitative properties before applying them to the cases and by asking only whether the dependency would have been visible, not whether the eventual attack could have been predicted. A stronger future test would reconstruct historical assessments using only evidence available before the relevant cryptanalytic result.

Case selection also limits generality. SHA-1 and RSA represent only two classes of historical failure, while the ML-KEM/X25519MLKEM768 comparison represents one contemporary post-quantum transition design choice. Other classes—including authenticated encryption, stateful protocols, hardware side channels, certificate ecosystems, and hybrid signature schemes—may expose structural properties not adequately represented here. The six SCAS dimensions should therefore be treated as a proposed framework subject to extension, not as a proof that all relevant forms of structure have been enumerated.

Finally, structural profiles are threat-model and deployment dependent. An interface that is unreachable in one system may be adversarially controllable in another; an algorithm replaceable in cloud software may be effectively immutable in certified hardware. SCAS comparisons are meaningful only when scope, attacker capabilities, and lifecycle boundaries are stated explicitly. The framework is intended to improve those statements, not eliminate the judgement required to make them.

9. Related Work

SCAS combines several established lines of security engineering at a different level of abstraction. Saltzer and Schroeder’s economy-of-mechanism principle argues for simple protection mechanisms [8]; Manadhata and Wing formalise software attack surface in terms of resources and access exposed to adversaries [18]. SCAS extends that intuition to cryptographic assumptions, state, composition, interfaces, and lifecycle dependencies.

Cryptographic lifecycle management and crypto-agility address algorithm change over time. Mosca provides the established migration-timing model discussed in Section 1.2 [21]; NIST SP 800-57 supplies algorithm-lifetime and key-management concepts [4]; and current NIST guidance treats replacement across software, hardware, firmware, protocols, and infrastructure as an explicit objective [2]. A recent systematic review documents the diversity of agility definitions [19], while formal work studies preservation of security across updates in composable models [20].

CycloneDX’s Cryptography Bill of Materials (CBOM) provides a complementary inventory layer for algorithms, keys, certificates, protocols, and their relationships [30]. CBOM answers *what is present*; SCAS adds assumptions, exposure, coupling, failure propagation, and replaceability. The operational CBOM-to-SCAS enrichment schema is given in Appendix A.

Conventional reductions and compositional-security proofs remain the tools for establishing security under specified models; SCAS makes the architectural consequences of those models comparable across a lifecycle. Recent AI-assisted cryptanalysis supplies empirical context for the uncertainty considered here: HAWK and CryptanalysisBench show machine-assisted contribution to cryptanalytic research without establishing weakness in current deployed standards [23], [25].

10. Conclusion

Long-lived secure systems cannot rely on the assumption that every cryptographic choice made at deployment will remain suitable for the complete lifetime of the infrastructure. The post-quantum transition makes that problem unusually visible, but the underlying issue is broader: assurance can erode through cryptanalysis, implementation discovery, protocol composition, interface exposure, standards change, or simple inability to replace an embedded dependency. AI-assisted cryptanalysis adds a further uncertainty because

both the rate and accessibility of future discovery are becoming harder to project, while public disclosure need not coincide with the existence of a capability.

This paper has introduced Structural Cryptographic Attack Surface (SCAS) as a qualitative method for making those dependencies explicit: which assumptions, states, interfaces, transformations, and lifecycle commitments must continue to hold; how they interact; how failure may propagate; and what would be required to restore assurance if one dependency ceased to be acceptable.

The case studies exercise different parts of the framework. SHA-1 provides a retrospective example of progressive cryptanalytic erosion; the RSA oracle case shows a coupled structural attack path whose practical cost diverges sharply from the headline factoring assumption; and the ML-KEM/X25519MLKEM768 comparison applies the min-structure relation prospectively. The latter demonstrates a key property of the method: additional structure can be surplus under one requirement set and necessary under another. The common output is therefore a requirements-aware description of where long-term cryptographic commitment resides.

For Secure-by-Design systems, the architectural consequence is the min-structure principle. Security mechanisms that provide required properties should be retained and justified; unnecessary dependencies should be removed, coupling should be minimised, and unavoidable long-term dependencies should be placed behind deliberate replacement boundaries. Where two constructions meet the same requirements, SCAS asks whether one structurally dominates the other, as formalised in Section 3.7. Simplicity without replaceability creates rigidity, while agility without structural discipline can create new attack surface. Long-lived assurance requires both.

Future work should test the repeatability of SCAS across independent assessors, expand the case set, and examine whether structural profiles improve real design and migration decisions. If the framework is useful, its value will be in reducing unnecessary dependence on assumptions whose future is uncertain and ensuring that, when an assumption changes, the surrounding system was designed to contain and survive the change.

Acknowledgment of AI Assistance

This paper was developed with generative AI (OpenAI ChatGPT) used in three deliberately bounded supporting roles: **research assistant**, **adversarial reviewer**, and **editorial assistant**. As a research assistant, it supported literature discovery, exploratory calculations, code and experiment design, and organisation of technical material. As an adversarial reviewer, it was used to challenge proposed hypotheses and arguments, seek counterexamples and conventional objections, make assumptions explicit, and distinguish conjecture from evidence and proof. At several stages, the AI was deliberately instructed to assume that a proposed claim was wrong and construct the strongest case against it. As an editorial assistant, it supported manuscript structure, clarity,

consistency of terminology, and calibration of claims to the available evidence.

The conceptual direction, hypotheses, selection of evidence, interpretation of results, and final conclusions remain the responsibility of the human author. AI-generated suggestions were treated as provisional rather than authoritative. Substantive factual claims, references, calculations, and technical assertions were independently checked against primary sources or reproducible analysis before inclusion. The AI did not exercise authorship or final scientific judgement.

References

- [1] National Institute of Standards and Technology, “Post-Quantum Cryptography,” Cryptographic Technology Group, NIST Computer Security Resource Center, 2026. [Online]. Available: <https://csrc.nist.gov/projects/post-quantum-cryptography>
- [2] E. Barker, L. Chen, D. Cooper, D. Moody, A. Regenscheid, M. Souppaya, W. Newhouse, R. Housley, S. Turner, W. Barker, and K. Kent, “Considerations for Achieving Crypto Agility: Strategies and Practices,” National Institute of Standards and Technology, NIST CSWP 39upd1, 2026, doi: 10.6028/NIST.CSWP.39-upd1.
- [3] European Commission, Directorate-General for Defence Industry and Space, “IRIS² — Secure Connectivity,” 2026. [Online]. Available: https://defence-industry-space.ec.europa.eu/eu-space/iris2-secure-connectivity_en
- [4] E. Barker, “Recommendation for Key Management: Part 1 – General,” National Institute of Standards and Technology, NIST SP 800-57 Part 1 Rev. 5, 2020, doi: 10.6028/NIST.SP.800-57pt1r5.
- [5] L. Shea, M. Haller, A. Suhl, N. Heninger, and E. Thomé, “Forging 1024-bit RSA signatures in nearly SNFS time,” Cryptology ePrint Archive, Paper 2026/2131, Preprint, 2026. [Online]. Available: <https://eprint.iacr.org/2026/2131>
- [6] L. Shea, M. Haller, A. Suhl, N. Heninger, and E. Thomé, “Nearly SNFS-Speed Signature Forgery Sans Factoring N,” implementation artifact accompanying Cryptology ePrint Archive Paper 2026/2131, 2026. [Online]. Available: <https://github.com/ucsd-hacc/NSNFSFNSFN>
- [7] A. Joux, D. Naccache, and E. Thomé, “When e -th roots become easier than factoring,” in *Advances in Cryptology – ASIACRYPT 2007*, LNCS 4833, Springer, 2007, pp. 13–28, doi: 10.1007/978-3-540-76900-2_2.
- [8] J. H. Saltzer and M. D. Schroeder, “The protection of information in computer systems,” *Proceedings of the IEEE*, vol. 63, no. 9, pp. 1278–1308, 1975, doi: 10.1109/PROC.1975.9939.
- [9] UK Government Digital Service and Central Digital and Data Office, “Follow the Government Cyber Security Standard,” updated Feb. 7, 2025. [Online]. Available: <https://www.gov.uk/guidance/follow-the-government-cyber-security-standard>
- [10] UK Cabinet Office, “Government Cyber Security Strategy: 2022 to 2030,” 2022. [Online]. Available: <https://www.gov.uk/government/publications/government-cyber-security-strategy-2022-to-2030/government-cyber-security-strategy-2022-to-2030-html>
- [11] X. Wang, Y. L. Yin, and H. Yu, “Finding collisions in the full SHA-1,” in *Advances in Cryptology – CRYPTO 2005*, LNCS 3621, Springer, 2005, pp. 17–36, doi: 10.1007/11535218_2.
- [12] M. Stevens, E. Bursztein, P. Karpman, A. Albertini, and Y. Markov, “The first collision for full SHA-1,” in *Advances in Cryptology – CRYPTO 2017*, LNCS 10401, Springer, 2017, pp. 570–596, doi: 10.1007/978-3-319-63688-7_19.
- [13] G. Leurent and T. Peyrin, “SHA-1 is a shambles: First chosen-prefix collision on SHA-1 and application to the PGP Web of Trust,” in *29th USENIX Security Symposium*, 2020, pp. 1839–1856.
- [14] National Institute of Standards and Technology, “Transitioning away from SHA-1 for all applications,” Dec. 2022. [Online]. Available: <https://csrc.nist.gov/news/2022/nist-transitioning-away-from-sha-1-for-all-apps>
- [15] National Institute of Standards and Technology, “Module-Lattice-Based Key-Encapsulation Mechanism Standard,” FIPS 203, 2024, doi: 10.6028/NIST.FIPS.203.
- [16] G. Alagic, E. Barker, L. Chen, D. Moody, A. Robinson, H. Silberg, and N. Waller, “Recommendations for Key-Encapsulation Mechanisms,” National Institute of Standards and Technology, NIST SP 800-227, 2025, doi: 10.6028/NIST.SP.800-227.
- [17] European Commission, “European Quantum Communication Infrastructure – EuroQCI,” 2026. [Online]. Available: <https://digital-strategy.ec.europa.eu/en/policies/european-quantum-communication-infrastructure-euroqci>
- [18] P. K. Manadhata and J. M. Wing, “An attack surface metric,” *IEEE Transactions on Software Engineering*, vol. 37, no. 3, pp. 371–386, 2011, doi: 10.1109/TSE.2010.60.
- [19] C. Näther, D. Herzinger, J.-P. Steghöfer, S.-L. Gazdag, E. Hirsch, and D. Loebenberger, “Toward a common understanding of cryptographic agility: A systematic review,” *IEEE Access*, vol. 14, pp. 133743–133763, 2026, doi: 10.1109/ACCESS.2026.3728196.
- [20] C. Badertscher, M. Ciampi, and A. Kiayias, “Agile cryptography: A universally composable approach,” in *Theory of Cryptography Conference*, 2023.
- [21] M. Mosca, “Cybersecurity in an era with quantum computers: Will we be ready?” Cryptology ePrint Archive, Paper 2015/1075, 2015. [Online]. Available: <https://eprint.iacr.org/2015/1075>
- [22] National Institute of Standards and Technology, “What Is Post-Quantum Cryptography?” 2026. [Online]. Available: <https://www.nist.gov/cybersecurity-and-privacy/what-post-quantum-cryptography>
- [23] Anthropic, “Discovering cryptographic weaknesses with Claude,” July 28, 2026. [Online]. Available: <https://www.anthropic.com/research/discovering-cryptographic-weaknesses>
- [24] National Institute of Standards and Technology, “Post-quantum cryptography,” 2026, noting the July 2026 withdrawal of HAWK following disclosure of an AI-assisted mathematical vulnerability. [Online]. Available: <https://www.nist.gov/pqc>
- [25] L. Fluri, A. Shafran, N. Carlini, M. Jagielski, M. Nasr, O. Dunkelmann, E. Ronen, and F. Tramèr, “CryptanalysisBench: Can LLMs do cryptanalysis?” arXiv:2607.18538, 2026. [Online]. Available: <https://arxiv.org/abs/2607.18538>
- [26] UK AI Security Institute, “How fast is autonomous AI cyber capability advancing?” 2026. [Online]. Available: <https://www.aisi.gov.uk/blog/how-fast-is-autonomous-ai-cyber-capability-advancing>
- [27] GCHQ, “James Ellis,” historical profile describing the classified development of public-key cryptography within GCHQ. [Online]. Available: <https://www.gchq.gov.uk/person/james-ellis>
- [28] D. Stebila, S. Fluhrer, and S. Gueron, “Hybrid Key Exchange in TLS 1.3,” RFC 9954, July 2026. [Online]. Available: <https://www.rfc-editor.org/rfc/rfc9954.html>
- [29] K. Kwiatkowski, P. Kampanakis, B. E. Westerbaan, and D. Stebila, “Post-Quantum Traditional (PQ/T) Hybrid Key Agreement Mechanisms for TLS 1.3,” RFC 10024, Aug. 2026. [Online]. Available: <https://www.rfc-editor.org/rfc/rfc10024.html>
- [30] OWASP CycloneDX, “Cryptography Bill of Materials (CBOM),” 2026. [Online]. Available: <https://cyclonedx.org/capabilities/cbom/>

Appendix A. CBOM-to-SCAS Enrichment Schema

The following mapping illustrates how a machine-readable Cryptography Bill of Materials can seed a SCAS

assessment. Inventory data identifies candidate components and relations; contextual analysis supplies the structural properties that cannot be inferred from inventory alone.

TABLE 3. ILLUSTRATIVE CBOM-TO-SCAS ENRICHMENT MAPPING.

CBOM object / relation	SCAS enrichment
Algorithm / primitive	Mathematical assumptions; implementation dependencies; lifecycle and replacement constraints.
Key / secret material	Entropy source; generation, storage, reuse, state, rotation, and compromise propagation.
Protocol / cipher suite	Composition; negotiation; message or interface exposure; downgrade and binding dependencies.
Certificate / trust object	Protocol coupling; validation assumptions; certificate-profile and replacement dependencies.
Component relation	Candidate compositional coupling and failure-propagation path.
Implementation metadata	Platform-specific timing, memory, acceleration, firmware, certification, and replaceability constraints.

Operationally, the flow is *CBOM inventory* $\rightarrow$ *candidate SCAS dependencies* $\rightarrow$ *contextual structural enrichment* $\rightarrow$ *SCAS profile* $\rightarrow$ *min-structure and lifecycle decision*. Semantics such as adversarial exposure and practical replaceability remain deployment-dependent.