NIST Cybersecurity White Paper
NIST CSWP 39

Considerations for Achieving Crypto Agility

Strategies and Practices

Elaine Barker*
Lily Chen
David Cooper*
Dustin Moody
Andrew Regenscheid
Murugiah Souppaya*
Computer Security Division
Information Technology Laboratory

Bill Newhouse
Applied Cybersecurity Division
Information Technology Laboratory

Russ Housley
Vigil Security

Sean Turner
sn3rd

William Barker†
Strativia LLC

Karen Kent
Trusted Cyber Annex

**Former employee; all work for this publication was done while at NIST.*

†Former employee; all work for this publication was done while at Strativia.

This publication is available free of charge from:
https://doi.org/10.6028/NIST.CSWP.39

December 19, 2025

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
U.S. DEPARTMENT OF COMMERCE

Publication History
Approved by the NIST Editorial Review Board on 2025-12-11

How to Cite this NIST Technical Series Publication:
Barker E, Chen L, Cooper D, Moody D, Regenscheid A, Souppaya M, Newhouse B, Housley R, Turner S, Barker W, Kent K (2025) Considerations for Achieving Crypto Agility: Strategies and Practices. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Cybersecurity White Paper (CSWP) NIST CSWP 39. https://doi.org/10.6028/NIST.CSWP.39

Author ORCID iDs
Elaine Barker: 0000-0003-0454-0461
Lily Chen: 0000-0003-2726-4279
David Cooper: 0000-0001-2410-5830
Dustin Moody: 0000-0002-4868-6684
Andrew Regenscheid: 0000-0002-3930-527x
Murugiah Souppaya: 0000-0002-8055-8527
Bill Newhouse: 0000-0002-4873-7648
Russ Housley: 0009-0001-6755-5962
Sean Turner: 0009-0005-9245-9752
William Barker: 0000-0002-4113-8861
Karen Kent: 0000-0001-6334-9486

Contact Information
crypto-agility@nist.gov

National Institute of Standards and Technology
Attn: Computer Security Division, Information Technology Laboratory
100 Bureau Drive (Mail Stop 8930) Gaithersburg, MD 20899-8930

Additional Information
Additional information about this publication is available at https://csrc.nist.gov/publications/cswp, including related content, potential updates, and document history.

Abstract

Cryptographic (crypto) agility refers to the capabilities needed to replace and adapt cryptographic algorithms in protocols, applications, software, hardware, firmware, and infrastructures while preserving security and ongoing operations. This white paper provides an in-depth survey of current approaches to achieving crypto agility. It discusses challenges and trade-offs and identifies approaches for providing operational mechanisms to achieve crypto agility. It also highlights critical working areas that require additional consideration.

Keywords

cryptographic agility; cryptographic algorithm; cryptographic application programming interface (API); cryptographic risk management; cryptographic transition.

Audience

Crypto agility is a cross-disciplinary topic with many stakeholders, including protocol designers, implementers, and operators; IT and cybersecurity architects; software and standards developers; hardware designers; and executives and policymakers. Achieving crypto agility requires cryptographic researchers to proactively address upcoming transitions and capture the attention of cryptographic application communities. Therefore, the intended audience also includes cryptographic researchers.

Acknowledgments

NIST appreciates the input and contributions from the collaborators of the National Cybersecurity Center of Excellence (NCCoE) Post-Quantum Cryptography (PQC) Migration project in the development of this white paper. The comments received during the public comment period and the discussions at the NIST Crypto Agility Workshop have greatly helped to improve the white paper. NIST sincerely acknowledges the commenters and workshop participants for their valuable contributions. A special note of thanks goes to Jim Foti and Isabel Van Wyk for reviewing and editing this document.

Table of Contents

Executive Summary....1

1. Introduction....2

2. Historic Transitions and Challenges....4

2.1. Long Period for a Transition....4

2.2. Backward Compatibility and Interoperability Challenges....4

2.3. Constant Needs of Transition....5

2.4. Resource and Performance Challenges....5

3. Crypto Agility for Security Protocols....7

3.1. Algorithm Identification....7

3.1.1. Mandatory-to-Implement Algorithms....8

3.1.2. Dependent Specifications....9

3.2. Algorithm Transitions....9

3.2.1. Preserving Protocol Interoperability....10

3.2.2. Providing Notices of Expected Changes....11

3.2.3. Integrity for Algorithm Negotiation....11

3.2.4. Hybrid Cryptographic Algorithms....12

3.3. Cryptographic Key Establishment....13

3.4. Balancing Security Strength and Protocol Complexity....14

3.4.1. Balancing the Security Strength of Algorithms in a Cipher Suite....14

3.4.2. Balancing Protocol Complexity....14

4. Crypto Agility in System Implementations....16

4.1. Using an API in a Crypto Library Application....16

4.2. Using APIs in the Operating System Kernel....17

4.3. Using Service Mesh in Cloud-Native Environments....18

4.4. Embedded Systems....18

4.5. Hardware....19

4.6. Using a Crypto Gateway for Legacy Systems....20

5. Crypto Agility Strategic Plan for Managing Organizations' Crypto Risks....22

5.1. Cryptographic Standards, Regulations, and Mandates....24

5.2. Crypto Security Policy Enforcement....24

5.3. Technology Supply Chains....25

5.4. Cryptographic Architecture....25

6. Considerations for Future Works....27

6.1. Resource Considerations....27

6.2. Agility-Aware Design 27
6.3. Complexity and Security 28
6.4. Crypto Agility in the Cloud 28
6.5. Maturity Assessment for Crypto Agility 29
6.6. Common Crypto API 31
7. Conclusion 33
References 34
Appendix A. List of Symbols, Abbreviations, and Acronyms 38
Appendix B. Definition of Crypto Agility in Other Literature 41

List of Figures

Fig. 1. Using a hybrid algorithm to transition to PQC 12
Fig. 2. Functional diagram of applications using a crypto API 17
Fig. 3. Crypto agility strategic plan for managing an organization's cryptographic risks 22

Executive Summary

Cryptographic algorithms have been relied upon for decades to protect every communication link and digital device. Advances in computing capabilities, cryptographic research, and cryptanalytic techniques sometimes necessitate replacing algorithms that no longer provide adequate security. A typical algorithm transition is costly, takes time, raises interoperability issues, and disrupts operations. *Cryptographic (crypto) agility* refers to the capabilities needed to replace and adapt cryptographic algorithms in protocols, applications, libraries, software, hardware, firmware, and infrastructures while preserving security and ongoing operations.

The threats posed by future cryptographically relevant quantum computers to public-key cryptography demand an urgent migration to quantum-resistant cryptographic algorithms. The impact of this transition will be much larger in scale than previous transitions because all public-key algorithms will need to be replaced rather than just a single algorithm. Also, this transition will certainly not be the last one required. Future cryptographic uses will demand new strategies and mechanisms to enable smooth transitions. As a result, crypto agility is a key practice that should be adopted at all levels, from algorithms to enterprise architectures.

This white paper provides an in-depth survey of current approaches for achieving crypto agility and discusses their challenges and trade-offs as an introduction for executives and policymakers. Sections 3, 4, and 6 present crypto agility considerations in technical detail and may be of interest to protocol designers, implementers, operators, IT and cybersecurity architects, software and standards developers, and hardware designers. Section 5 examines strategic planning for crypto agility, which should be beneficial for organizational risk management, governance, and policy professionals.

Executives can leverage the insights in this paper to develop a comprehensive strategic and tactical plan that integrates crypto agility into the organization’s overall risk management framework, ensuring that employees, business partners, and technology suppliers involved in cryptographic design, implementation, acquisition, deployment, and use consider and adopt these practices.

1. Introduction

Advances in computing capabilities, cryptographic research, and cryptanalytic techniques frequently create a critical need to replace algorithms that no longer provide adequate security for their use cases with algorithms that are considered secure. Historically, cryptographic transitions take place over several decades. For example, block ciphers transitioned from single DES to Triple DES and then to AES due to rapidly increasing computing power and more sophisticated cryptanalysis techniques. Each transition is costly, takes time, raises interoperability issues, and disrupts operations.

The threats posed by future cryptographically relevant quantum computers (CRQCs) to public-key cryptography demand an urgent migration to quantum-resistant cryptography. The impact of transitioning to post-quantum cryptography (PQC) will be much larger in scale than previous transitions because all public-key algorithms will need to be replaced rather than just a single algorithm. These algorithms have been used for decades to protect every communication link and digital device. With the rapid growth of computing power and cryptographic techniques, this PQC transition will certainly not be the last transition required. Future cryptographic applications will demand new strategies and mechanisms to enable smooth transitions.

Cryptographic (crypto) agility describes the capabilities needed to replace and adapt cryptographic algorithms in protocols, applications, software, hardware, firmware, and infrastructures while preserving security and ongoing operations. Many definitions and descriptions for crypto agility have been proposed, some of which are listed in Appendix B.

Crypto agility facilitates migrations between cryptographic algorithms without significant changes to the application that is using the algorithms. Its exact definition is highly dependent on specific organizational and technical contexts. For example:

- **Crypto Agility for a Computing System:** Cryptographic algorithms are implemented in software, hardware, firmware, and infrastructures to facilitate their use in applications. For example, replacing a cryptographic algorithm in applications often requires changes to application programming interfaces (APIs) and software libraries [1]. It may also necessitate the replacement of hardware to incorporate new hardware accelerators. *In a system, crypto agility is the ability to adopt new cryptographic algorithms and stop the use of vulnerable algorithms in applications without disrupting the running system*.
- **Crypto Agility for a Communication Protocol:** In a communication protocol, parties must agree on a common *cipher suite*: a common set of cryptographic algorithms used for key establishment, signature generation, hash function computation, encryption, and/or data authentication. Any update of algorithms must be reflected in the protocol specifications. *In a protocol, crypto agility is the ability to maintain interoperability when introducing new cryptographic algorithms and preventing the use of vulnerable algorithms*.
- **Crypto Agility for an Enterprise IT Architect:** Achieving crypto agility is not only a task for product designers, implementors, and operators but also for information technology

(IT) and cybersecurity architects, software and standards developers, hardware designers, and executives. *In an enterprise, crypto agility is a capacity to seamlessly and rapidly transition away from vulnerable cryptographic algorithms and adopt new, stronger ones without incurring significant changes to their infrastructure or experiencing unnecessary disruptions.*

Achieving crypto agility requires a systems approach to providing mechanisms that enable the transition to alternative algorithms and limit the use of vulnerable algorithms in a seamless way while maintaining security and acceptable operation. Significant effort has been made by the research and application community in approaching crypto agility. Some sector-specific guidelines and strategies were developed and are referred to in Appendix B. This white paper surveys crypto agility approaches in different implementation environments and proposes strategies for achieving the agility needs of varied applications. This paper also discusses crypto agility in different contexts and highlights the coordination needed among stakeholders.

The paper is structured as follows:

- Section 2 discusses the challenges faced in past transitions.
- Section 3 examines the challenges and existing practices in achieving crypto agility for security protocols.
- Section 4 addresses strategies for supporting crypto agility for applications — from an API to software libraries or hardware. Some of the strategies have been implemented in today's systems, and others will be considered in the future.
- Section 5 presents the use of a crypto agility strategic plan for managing an organization's cryptographic risks in an enterprise environment.
- Section 6 identifies important areas for consideration and future actions.
- Section 7 provides concluding thoughts.

2. Historic Transitions and Challenges

Most early cryptographic applications were deployed using algorithms that were expected to be used throughout the lifetime of the applications or systems. However, due to increases in computing power, advances in cryptanalytic techniques, and regulatory obsolescence, many algorithms require quick replacement within the lifetime of a system. The desire to make a replacement might be driven by concerns regarding the algorithm itself or related to a particular implementation of the algorithm. As a result, cryptographic transitions to replace algorithms or swap algorithm implementations should be an important part of security practices within an organization-wide risk management program.

In the past 50 years, applications involving cryptography have undergone multiple transitions. This section describes historical challenges, and the lessons learned to provide readers with necessary background and illustrate how systems that use cryptography should be designed to facilitate algorithm changes throughout their lifetimes.

2.1. Long Period for a Transition

Historically, decisions about the cryptographic algorithms used for applications were made without considering future transitions. Algorithms were sometimes implemented in a manner that was difficult to change, making maintenance and the addition of new algorithms hard to accomplish.

In 1977, the Data Encryption Standard (DES) became the first published encryption standard. The DES algorithm [2] had a 64-bit block size and a 56-bit key. Motivated by the threat of a practical brute-force attack against DES's 56-bit key, a variation of DES called Triple DES [3] (due to its capacity to perform DES operations with two or three 56-bit keys) was introduced in 1999 as a temporary solution before a stronger algorithm could be standardized and made available for use. This new and stronger algorithm, called the Advanced Encryption Standard (AES) [3] (with options for 128-, 192-, or 256-bit keys), was standardized in 2001. However, Triple DES continued to be used in many applications until it was finally disallowed in 2024. In all, the complete transition from Triple DES to AES took 23 years.

2.2. Backward Compatibility and Interoperability Challenges

The need for backward compatibility can also be a barrier to transition. For example, hash functions are used as a message digest in digital signatures, for the generation of message authentication codes (MACs), for key-derivation functions, and for random number generation. Cryptographic hash functions have also been used as a basic component in hash-based signatures. Cryptographic hash function requirements include collision resistance, pre-image resistance, and second pre-image resistance. SHA-1, a hash function with a 160-bit output length [4], was expected to provide 80 bits of collision resistance and 160 bits of pre-image resistance. Many use cases relied on these security properties. However, in 2005, SHA-1 was found to provide fewer than 80 bits of collision resistance [5]. In 2006, NIST responded by urging federal agencies to “stop relying on digital signatures that are generated using SHA-1 by the end of 2010.”

Because SHA-1 had been used in signatures for entity authentication in many existing secure protocols, interoperability and backwards compatibility had to be considered in the transition. In particular, using SHA-1 in digital signatures for entity authentication had to be allowed in certain circumstances for some protocols, such as Transport Layer Security (TLS) (Section 4.4.2.2 of [6]). Since 2005, more powerful and practical collision attacks on SHA-1 have been demonstrated by researchers [7], and NIST now recommends a complete transition away from SHA-1 by the end of 2030 [8]. This example shows that when some applications do not have crypto agility and cannot make timely transitions, a longer transition period may need to be allowed in order to facilitate backward compatibility.

2.3. Constant Needs of Transition

For a public-key cryptographic algorithm, security strength is determined by parameter selection. For example, one of the parameters for the RSA algorithm is the modulus size. When the use of RSA was first approved for digital signatures in 2000 as specified in Federal Information Processing Standards (FIPS) publication 186-2 [9], a minimum modulus size of 1024 bits was required to provide at least 80 bits of security strength. In 2013, the minimum modulus was increased to 2048 bits to provide a security strength of at least 112 bits due to progress in integer factorization and increases in computing power.

For many devices, the RSA key size (modulus) is fixed for the device. However, a transition to a larger RSA key size (modulus) may need to happen during a device’s lifetime. If a device is not designed to transition to a larger RSA key size (modulus) during its lifetime, the device will need to be replaced. Given the long-desired lifespan of many devices, it is generally more cost-effective to design the device for such transitions during its development.

When one transition is planned, another transition can appear for a different reason. Since 2005, NIST Special Publication (SP) 800-57 Part 1 [10] has projected the need to transition to a minimum of 128 bits of security strength by 2031. Because of the emerging need to transition to post-quantum cryptography, NIST Internal Report (IR) 8547 [11] stated that the 112-bit security strength for the current public-key algorithms would be disallowed in 2031 to facilitate a direct transition from the 112-bit security strength provided by current public-key schemes to post-quantum cryptography.

2.4. Resource and Performance Challenges

Transitions in general and transitions to post-quantum algorithms in particular present many challenges. Some quantum-resistant algorithms have larger sizes for public keys, signatures, and ciphertext than those for classic public-key algorithms. For example, an RSA key size (modulus) of 3072 bits provides roughly 128 bits of classical security strength. The transition to the post-quantum Module-Lattice-Based Digital Signature Algorithm (ML-DSA) specified in FIPS 204 will result in a signature of 2420 bytes (i.e., 19,360 bits) to provide a roughly equivalent classical security strength of 128 bits [12]. This shows that transitioning to new algorithms can challenge the capacity of a communication network and increase the time to transmit public keying material and message signatures.

3. Crypto Agility for Security Protocols

Many security protocols use cryptographic algorithms to provide confidentiality, integrity, authentication, and/or non-repudiation. Communicating peers must agree on a common set of cryptographic algorithms to provide these security services, which are collectively referred to as a *cipher suite*. The process of agreeing on the set of algorithms within a security protocol is called *cipher suite negotiation*. The cipher suite may include algorithms for integrity protection, authentication, key derivation, key establishment, encryption, and digital signatures to provide the needed security services. Crypto agility is achieved when an implementation of a security protocol can easily transition from one cipher suite to another, more desirable one. Each security protocol normally specifies a suite of mandatory-to-implement algorithms to ensure basic interoperability. However, a mandatory-to-implement algorithm may need to be replaced if a vulnerability is found.

To achieve crypto agility, security protocol implementations should be modular so they can easily accommodate the insertion of new algorithms or cipher suites. Implementations should also provide a way to determine when deployed implementations have shifted from the old algorithms to more desirable ones. The set of mandatory-to-implement algorithms is expected to change over time, and the cipher suite negotiation mechanism needs to accommodate the identification of yet-to-be specified algorithms in the future.

This section discusses challenges and existing practices in achieving crypto agility for security protocols.

3.1. Algorithm Identification

Security protocols include a mechanism to identify the algorithm or cipher suite in use. An algorithm identifier might be explicitly carried in the protocol, a management mechanism can be used to identify the algorithm, or the algorithm might be implied by the protocol version number. If a security protocol does not carry an explicit algorithm identifier, a new protocol version number is needed to identify the use of a new algorithm or cipher suite.

The version number of a protocol or an algorithm identifier is needed for an implementation to tell communicating peers which algorithm or cipher suite is being used. Changing the version number of a protocol usually requires significant effort by the standards developing organization (SDO). Thus, crypto agility is easier to achieve when security protocols include algorithm or cipher suite identifiers.

In some security protocols, a combination of the protocol version number and explicit algorithm or cipher suite identifiers is defined. For example, in TLS version 1.2 (TLSv1.2) [13] and TLS version 1.3 (TLSv1.3) [6], the version number specifies the hash function that is used as a binder for external pre-shared keys (PSKs).

Some security protocols carry one identifier for each algorithm that is used, while other security protocols carry one identifier for a cipher suite that specifies the use of multiple algorithms. For example, in the IPsec protocol suite, Internet Key Exchange Protocol version 2 (IKEv2) [14] most commonly negotiates algorithms with a separate identifier for each algorithm. In contrast, TLSv1.3 [6] negotiates algorithms with cipher suite identifiers. Both identification approaches are used successfully in security protocols, and both require the assignment of new identifiers to add support for new algorithms.

Designers are encouraged to pick one of these approaches and use it consistently throughout the protocol or family of protocols. Cipher suite identifiers make it easier for the protocol designer to avoid incomplete specifications because each cipher suite selects the algorithms for all cryptographic services. However, cipher suite identifiers inherently face a combinatoric explosion when all useful combinations of algorithms are specified. On the other hand, using multiple algorithm identifiers rather than cipher suites imposes a burden on implementations to determine which algorithm combinations are acceptable during session establishment. This determination is often made through a negotiation that is built into session establishment (sometimes called *security association establishment*). Local policy can limit the allowable combinations.

Regardless of the mechanism used, security protocols historically negotiate the symmetric cipher and cipher mode together to ensure that they are compatible. As a result, one algorithm identifier names the symmetric cipher, the key size, and the cipher mode.

In some protocols, the length of the key to be used is not specified by the algorithm or cipher suite identifier, thus allowing the key length to be flexible. For example, TLSv1.2 cipher suites include Diffie-Hellman key exchange without specifying a particular public-key length. When the algorithm identifier or suite identifier specifies a particular public-key length, migration to longer lengths would require the specification, implementation, and deployment of a new algorithm or cipher suite identifier. In contrast, a flexible public-key length in a cipher suite would make it easier to migrate away from short key lengths when the computational resources available to an attacker dictate the need to do so. However, the flexibility of asymmetric key lengths has led to interoperability problems when the key length is not firmly established. To avoid these interoperability problems in the future, any aspect of the algorithm not specified by the algorithm identifiers needs to be negotiated, including the key size and other parameters.

3.1.1. Mandatory-to-Implement Algorithms

For secure interoperability, communicating peers must agree on a common set of secure cryptographic algorithms. While many algorithms are often specified for a security protocol, an implementation may not support all possible algorithms. To ensure that interoperation is possible for all implementations, an SDO will often choose at least one set of algorithms with properly selected security strengths based on state-of-the-art cryptanalysis results as mandatory-to-implement (i.e., to be supported by all implementations). Of course, local policy may select an algorithm other than the mandatory-to-implement one.

However, SDOs need to change the set of mandatory-to-implement algorithms over time to keep up with advances in computing and cryptanalysis. For example, NIST has withdrawn approval for the DES and Triple DES algorithms. Each was a mandatory-to-implement algorithm in various security protocols at one time. It is highly desirable for SDOs to be able to revise mandatory-to-implement algorithms without modifying the base security protocol specification. To achieve this goal, some SDOs publish a base security protocol specification and a companion document that describes the supported algorithms, which allows for one document to be updated without necessarily modifying the other.

SDOs should specify the new algorithms before the current ones have weakened to the breaking point. For example, support for the AES algorithm was introduced in S/MIME v3.1 [15] in 2004, and the AES algorithm became mandatory to implement in S/MIME v3.2 [16] in 2010. This approach allows for a timely migration to the new algorithms while the old algorithms are still able to meet their security expectations. However, a failure of implementers and administrators to take prompt action to transition will increase the period of time that an old algorithm is used, perhaps dangerously so.

3.1.2. Dependent Specifications

Mandatory-to-implement algorithms are not specified for protocols that are embedded in other protocols. In these cases, the higher-level protocol specification identifies the mandatory-to-implement algorithms used in the embedded protocols. For example, S/MIME version 3.2 [16] (a higher-level protocol) makes use of (i.e., embeds) the cryptographic message syntax (CMS) [17]. Thus, S/MIME (not CMS) specifies the mandatory-to-implement algorithms. This approach allows various security protocols to use the CMS and make independent choices regarding which algorithms are mandatory to implement.

To add a new algorithm, the conventions for using that new algorithm are specified for the embedded security protocol (i.e., the CMS in the example above), and then at some future time, the higher-level protocol (i.e., S/MIME in the example above) might make that algorithm mandatory to implement.

3.2. Algorithm Transitions

Transitioning from a vulnerable algorithm can be complicated. It is relatively straightforward to specify how to use a new, better algorithm. However, developing, implementing, and deploying a security protocol specification using the new algorithm can often take years, especially if a new or additional infrastructure is required prior to deployment. The physical location of devices can add challenges to upgrades, especially for remote sensors and space systems. Overcoming these challenges takes time and increases cost. When the new algorithm is widely deployed, it should be used in lieu of the old algorithm. However, knowledge about the actual use and security of the new algorithm will always be imperfect, so one cannot be completely sure that it is safe to remove the old algorithm from an implementation.

A cryptographic key is associated with a particular algorithm, which means that key expiration and revocation are important tools for cryptographic algorithm transition. For example, the validity period on a certificate will ensure that the public key contained in the certificate is not used for authentication by a relying party beyond certificate expiration. Likewise, revoking the certificate indicates to a relying party that the public key should not be used, even if the certificate is not expired.

Algorithm transition is naturally facilitated as part of an algorithm selection or negotiation mechanism. During the negotiation phase, security protocols select the most suitable algorithm or cipher suite that is supported by all communicating peers and acceptable by their policies. In addition, a mechanism to determine whether a new algorithm has been deployed is often needed. For example, the SMIME Capabilities attribute [16] allows S/MIME mail user agents to share the list of algorithms that they are willing to use in order of preference. A secure email sender can tell that it is possible to use a new algorithm when all recipients include it in their SMIME Capabilities attribute. As another example, the Extension Mechanisms for DNS (EDNS(0)) [18] can be used in Domain Name System Security Extensions (DNSSEC) to signal the acceptance and use of new digital signature algorithms. In Resource Public Key Infrastructure (RPKI), all implementations must support the same digital signature algorithm. To ensure global acceptance of a digital signature, an approach to transition has been specified in which a new signature algorithm is introduced well before the original algorithm is phased out [19].

In the worst case, a deeply flawed algorithm may still be available and used in an implementation, which could permit an attacker to compromise data by downloading a simple attack script. Flawed security can also occur when a secure algorithm is used incorrectly or with poor key management. In such situations, it is not possible to provide notice to implementers (see Sec. 3.2.2), and the protection offered by the algorithm is severely compromised. Administrators may choose to stop using the vulnerable cipher suite that includes the algorithm well before the new cipher suite is widely deployed. This can happen by picking a date for a global switch to the new algorithm, or each installation can select a date on their own. In either case, interoperability will be sacrificed with any implementation that does not support the new crypto suite. Additionally, administrators can change a configuration setting to prohibit the use of a particular algorithm as a means of forcing algorithm transition.

3.2.1. Preserving Protocol Interoperability

Removing support for disallowed and obsolete cryptographic algorithms is challenging. Once an algorithm is determined to be vulnerable, it is difficult to eliminate all uses of that algorithm because many applications and environments rely on it. Since algorithm transitions can introduce interoperability problems, protocol designers and implementers may be inclined to delay the removal of support for vulnerable algorithms. As a result, flawed algorithms can be supported for far too long. The security impact of using legacy software that includes the flawed algorithm and having extended support periods can be reduced by making algorithm transitions easy. Social pressure is often needed to cause the transition to happen. For example, the RC4 stream cipher was supported in web browsers until Andrei Popov championed an effort to stop its use [20].

Implementers are often reluctant to remove disallowed [21] algorithms from server software, and server administrators are often reluctant to disable them over concerns that some party will no longer have the ability to connect to their server. Implementers and administrators want to improve security by using the strongest supported algorithms, but their actions are tempered by the desire to preserve backward compatibility. Some web browsers provide a visual warning when a disallowed algorithm is selected for use. These visual warnings provide an incentive for website operators to transition away from these algorithms.

Transitioning in the internet infrastructure and enterprise infrastructure is particularly difficult. The digital signature on a certification authority (CA) [22] certificate is often expected to last decades, which hinders transition away from a vulnerable signature algorithm. Once a long-lived certificate is issued with a particular signature algorithm, that algorithm is used by many relying parties to verify certificates signed by the CA, and none of the relying parties can stop supporting it without invalidating all of the certificates signed by that CA. Many certificates can be impacted by the decision to drop support for a vulnerable signature algorithm or an associated hash function since all certificates signed using that algorithm or hash function would need to be replaced.

Influential organizations like NIST and the Internet Engineering Task Force (IETF) can assist with overcoming the conflicting desire to preserve interoperability by coordinating the deprecation of an algorithm or cipher suite and simplifying the transition for users.

3.2.2. Providing Notices of Expected Changes

Cryptographic algorithm failures without warning are rare. Algorithm transitions are typically driven by advancements in computing capabilities, cryptographic research, and cryptanalytic techniques rather than unexpected failures. For example, the transition from DES to Triple DES to AES took place over decades, resulting in a shift in symmetric block cipher security strength from 56 bits to 112 bits to at least 128 bits. When possible, SDOs should provide notice to security protocol implementers about expected algorithm transitions.

Monitoring cryptographic research results provides a way to discover new attacks, assess impacts to existing security protocols, and foresee needed changes. In the worst case, a breakthrough cryptanalytic technique can indicate the need for an immediate algorithm transition. Crypto agility is needed to smoothly implement such a transition.

As part of their crypto agility efforts in the transition to PQC, security protocol designers need to plan for public keys, signatures, and key-encapsulation ciphertext to be much larger than those currently used. Public-key sizes and signature sizes directly impact the size of the certificates that contain those keys and signatures. To be safe, security protocol designers should plan for significant growth in the size of cryptographic data.

3.2.3. Integrity for Algorithm Negotiation

The mechanism that a security protocol uses to perform cryptographic algorithm negotiation should include integrity protection. If the integrity of algorithm selection during negotiation is not protected, the protocol will be subject to a downgrade attack in which an attacker influences the choice of cipher suite, and one with vulnerable algorithms is chosen. If a protocol specifies a single integrity algorithm to protect the negotiation without a way to negotiate an alternative integrity algorithm, significant problems will result in the event that the single algorithm is found to be vulnerable.

Extra care is needed when a mandatory-to-implement algorithm is used to provide integrity protection for the negotiation of other cryptographic algorithms. In this case, the integrity protection should be at least as strong as that provided by a future cipher suite, which can result in the need for several mandatory-to-implement algorithms to cover the various security strength requirements. Otherwise, a flaw in the mandatory-to-implement integrity algorithm may allow an attacker to influence the choices of the other algorithms.

Security protocols can negotiate a key-establishment mechanism, derive an initial cryptographic key, and then authenticate the negotiation. However, if the authentication fails, the only recourse is to start the negotiation over from the beginning. This is necessary for security but can lead to an unacceptable delay for the human user when authentication is unsuccessful.

3.2.4. Hybrid Cryptographic Algorithms

A hybrid cryptographic algorithm is a combination of two or more components that are themselves cryptographic algorithms. One early hybrid algorithm was a pseudorandom function (PRF) introduced in TLS 1.0 [23], which combined MD5 and SHA-1. One use case for hybrid public-key algorithms is to continue using the well-tested, traditional public-key algorithms while study of the new PQC algorithms continues and implementations mature. Choosing a hybrid algorithm may lead to a second transition when the traditional algorithm is disallowed.

However, if the overhead associated with the traditional algorithm is small, some security protocol implementations will avoid the second transition to only using the PQC algorithm by continuing to use the hybrid algorithm even when the traditional algorithm is no longer secure. Some SDOs are considering a hybrid of more than one PQC algorithm.[1] In this section, we examine the hybrid approach of combing a traditional algorithm with a PQC algorithm. Figure 1 represents the different transition paths when using a hybrid algorithm to transit to PQC.

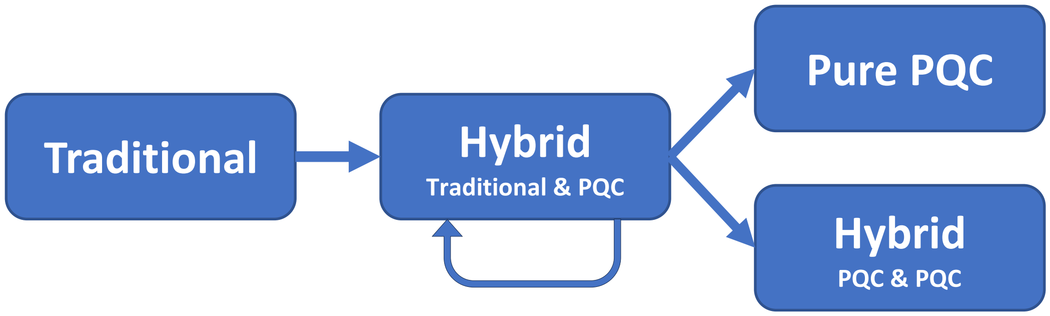

Fig. 1. Using a hybrid algorithm to transition to PQC

A hybrid signature algorithm combines a traditional signature algorithm and a PQC signature algorithm (e.g., combining Elliptic Curve Digital Signature Algorithm [ECDSA] and ML-DSA [12]). It requires two public keys to be certified: a public key for the traditional algorithm and a PQC public key. One option is to include the two public keys in a single certificate, where the public keys would always be used together. The PKI root of trust must be known to all of the relying

[1] Some of the hybrid algorithm specifications refer to "composite algorithms." At the level of the discussion in this section, the distinctions between "hybrid" and "composite" algorithms are unimportant. Thus, this section uses "hybrid" throughout.

parties to validate a certificate, and the cost of deploying a PKI root of trust is significant, so the expense associated with a transition to the use of a hybrid root of trust followed by a potential second transition to using only a PQC algorithm for a root of trust must be considered.

Another option is the deployment of a traditional root of trust and a PQC root of trust using separate certificates. In some cases, two certificates will be less expensive, but there are operational costs associated with validating two certification paths to establish a session key. A significant advantage of using separate roots of trust is that once the traditional PKI is no longer needed, one can simply stop issuing certificates under the traditional root of trust, while the PQC trust anchor continues to be used.

A hybrid key-establishment algorithm establishes a shared secret by combining the outputs of a traditional key-establishment algorithm and a PQC key encapsulation mechanism (KEM) (e.g., Elliptic Curve Diffie-Hellman (ECDH) [24] and Module-Lattice-Based Key Encapsulation Mechanism (ML-KEM) [25]). Examples of hybrid key-establishment algorithms are discussed in SP 800-227 [26]. The assumption is that at least one of the algorithms will remain strong over time. Security analysis for a hybrid key-establishment algorithm can be more complicated than analysis of either of the algorithms that are used in the hybrid algorithm.

In summary, hybrid signatures or key-establishment schemes can be a good strategy for preserving security in the face of uncertainty while transitioning from traditional public-key cryptography to PQC. However, hybrid schemes increase protocol complexity and exercise the capability to accommodate many cipher suites and highlight the crypto agility of a security protocol design.

3.3. Cryptographic Key Establishment

Some environments restrict key-establishment approaches by policy. Such policies tend to improve interoperability within a particular environment, but they cause problems for individuals who need to work in multiple incompatible environments. In addition, administrators need to be aware that multiple environments are being used, track the policies, and enable the algorithms or cipher suites for each of them.

Support for many key-establishment mechanisms in a security protocol offers more opportunities for crypto agility. Key establishment can include key-agreement mechanisms, key-transport mechanisms, and KEMs. Security protocol designers should perform an analysis to ensure that all security goals are achieved when each of the possible key-establishment mechanisms is used.

Traditionally, security protocol designers have avoided support for more than one mechanism for exchanges that establish cryptographic keys because such support would make the security analysis of the overall protocol more difficult. When frameworks like the Extensible Authentication Protocol (EAP) [27] are employed, the authentication mechanism often provides a session key in addition to performing authentication. As a result, key establishment is very flexible, but many of the cryptographic details are hidden from the application, which makes security analysis more difficult. Furthermore, this flexibility results in protocols that support multiple key-establishment mechanisms. In fact, the key-establishment mechanism itself is negotiable, which creates a design challenge to protect the negotiation of the key-establishment mechanism before it is used to produce cryptographic keys.

3.4. Balancing Security Strength and Protocol Complexity

When a protocol designer is specifying a cipher suite, the relative strength of each algorithm needs to be considered, and complexity in security protocols needs to be avoided. Each of these design goals is explored further in this section.

3.4.1. Balancing the Security Strength of Algorithms in a Cipher Suite

When selecting or negotiating a cipher suite, the relative strength of each algorithm needs to be considered. Generally, each of the algorithms in a cipher suite meet or exceed the minimum-security strength requirements. However, when the performance of a particular algorithm does not impact overall performance, using the strongest choice across all of the cipher suites can reduce complexity by reducing the number of algorithms that need to be supported. The security protections provided by each algorithm in a particular context need to be considered when making the selection. Algorithm strength needs to be considered when a security protocol is designed, implemented, deployed, and configured. Advice from experts about relative algorithm strengths is useful, but such advice is often unavailable to system administrators who are deploying a protocol implementation. For this reason, SDOs should provide clear guidance to implementers that leads to options with greater than or equal security strengths being available at the time of deployment.

Performance and security need to be balanced. Users will not employ security features if the application runs too slowly when they are used. Some algorithms offer flexibility in their strength by adjusting the key size, number of rounds, authentication tag size, parameter set, and so on. For example, AES-128 is more efficient than AES-256, but it also offers less security.

When security protocols support a single key-establishment mechanism, or a flexible framework is profiled to a single choice (e.g., EAP is used with a single authentication mechanism), the security analysis is much more straightforward. However, crypto agility is reduced.

3.4.2. Balancing Protocol Complexity

Security protocol design complexity can increase the opportunity for undiscovered, exploitable implementation bugs. Optional features can add complexity and lead to parts of an implementation rarely being exercised. A security protocol with fewer options means that there is a lower burden on implementation testing and a decreased attack surface, which provides fewer potential points of entry for attackers.

Security protocol designers need to balance agility with simplicity and anticipate changes to the supported set of cryptographic algorithms over time. Security protocol implementations should be as straightforward as possible to reduce vulnerability to attacks. While support for multiple algorithms is necessary to achieve crypto agility, gratuitous options should be avoided to keep implementations from becoming unnecessary large. For example, complex algorithm or cipher suite negotiation may provide opportunities for downgrade attacks. Support for many algorithm alternatives is also harmful because of the challenges in deciding which algorithms are acceptable in a particular environment and maintaining that list of algorithms over time. Furthermore, once an algorithm is disallowed, it should be removed from the protocol to reduce complexity. Implementers should periodically review the options that are actually being used and then remove unused options to keep the attack surface as small as possible.

4. Crypto Agility in System Implementations

A cryptographic application programming interface (crypto API) separates the implementation of applications that use cryptographic algorithms (e.g., email and web apps) from implementation of the cryptographic algorithms themselves. This separation allows the application to focus on high-level, application-specific details, while the cryptographic algorithms are implemented by a provider or a library to handle symmetric encryption and decryption, digital signature generation and verification, hashing, random number generation, and key establishment while also supporting the old and new algorithms during the transition.

For example, crypto APIs can separate AES-CCM [28] and AES-GCM [29], which are both authenticated encryption with associated data (AEAD) algorithms, from implementations by allowing an implementation to make the same crypto API calls to use either algorithm. Careful selection of default parameter values in the crypto API can make the interface to these two algorithms essentially identical, which facilitates future transition to a different AEAD algorithm.

Some crypto APIs offer implementations of security protocols like TLS or IPsec to further unburden the protocol implementation. These protocol implementations depend on other crypto APIs for cryptographic operations. The application provides the list of algorithms or cipher suites that are available and acceptable, and the algorithm negotiation capabilities for the protocol determine the algorithms that are used in the protocol.

To achieve crypto agility, system designers must introduce mechanisms that simplify the replacement of cryptographic algorithms in software, libraries, hardware, firmware, and infrastructures. These mechanisms will, at the same time, increase complexity. Therefore, system designers must make sure that the cryptographic interface is easy to use and well-documented to reduce the risk of errors. Additionally, clear guidance must be provided for practitioners to follow.

4.1. Using an API in a Crypto Library Application

A cryptographic service provider (CSP) is an implementation of one or more cryptographic algorithms that is accessible by applications through a crypto API (see Fig. 2).

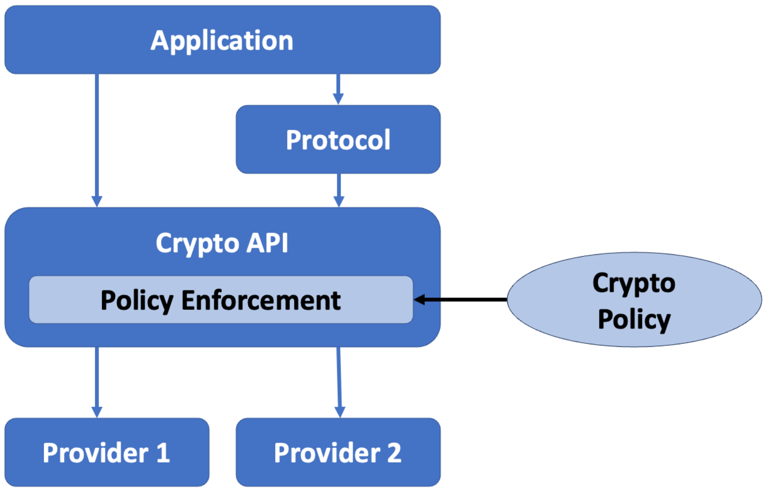

Fig. 2. Functional diagram of applications using a crypto API

CSPs are sometimes associated with protected key storage. For example, a CSP associated with a Trusted Platform Module (TPM) will also provide access to the asymmetric private keys that are stored on the TPM.

The cryptographic algorithm policy is set by the system administrator, which might be done to implement a policy set by the Chief Information Security Officer (CISO). The policy will indicate whether a particular algorithm is allowed. For example, if there is a provider for Triple DES, calls to encrypt with it will fail if the policy does not allow Triple DES. However, calls for Triple DES decryption might still be allowed so that stored files or email messages can be decrypted.

Some protocols are implemented in user space, which is an area in memory where applications are executed that is distinct from kernel space. Kernel space is a part of a computer memory that can only be accessed by the operating system. For example, application-chosen TLS crypto library applications operate in user space. In fact, most libraries run in user space, such as OpenSSL, BoringSSL, Bouncy Castle, Network Security Services (NSS), and OpenSSH. Application developers need to consider whether the API is provided via the command line interface (CLI) or by incorporating the cryptographic algorithm’s source code into the application (i.e., “compiling in” support).

For software libraries, it is important to facilitate efficient updates. Some standard mechanisms must be in place to avoid security pitfalls in library updates.

4.2. Using APIs in the Operating System Kernel

Some security protocols run in the operating system kernel, which is a computer program that is generally first loaded when the system is turned on and has complete control over all system resources accessible to all application programs in the system. For example, in the case of IPsec Encapsulating Security Payload (ESP), datagram encryption and authentication operate in the kernel. Similarly, disk encryption needs to run in the kernel.

To provide crypto agility in this case, the crypto API must also be accessible within the kernel. In some operating systems, only a subset of the overall capabilities of the crypto API are available from within the kernel. This subset is determined by the cryptographic operations required in the kernel. In many operating systems, the supported algorithms are established when the kernel is built, meaning that plugins to add algorithms are not available.

Some systems perform self-tests of the cryptographic functions as part of the operating system startup process. These tests ensure that the cryptographic operations are working as expected before the system is available to applications or users.

Where feasible, operating system designers should consider modular or updatable cryptographic components in the kernel. This might involve using loadable kernel modules or driver-like interfaces for cryptographic providers so that new algorithms can be added without rebuilding the entire kernel. At minimum, designing kernel crypto libraries to be easily replaced or patched (with proper code signing and testing) will significantly improve agility.

4.3. Using Service Mesh in Cloud-Native Environments

Emerging architectural patterns like service mesh and sidecar proxies in modern cloud-native environments can enhance crypto agility within distributed systems that are composed of numerous microservices. By abstracting cryptographic operations from application workloads and centralizing policy enforcement, the proxy acts as the cryptographic service provider and offers a uniform mechanism for negotiating, enforcing, and evolving cryptographic policies across all application components.

This approach enables organizations to seamlessly update cryptographic libraries, manage key rotation, and introduce new algorithms with minimal disruption to running services. It also supports algorithm transitions defined in the organization's crypto policies in response to evolving security requirements.

4.4. Embedded Systems

In embedded systems, some security protocols must run within the real-time operating system (RTOS) kernel or privileged system tasks. An RTOS is typically initialized during device startup and manages critical hardware resources, task scheduling, and inter-process communication for the entire embedded application. For example, secure communication protocols (e.g., TLS) offload encryption, authentication, and key management operations into system-level services within the RTOS to meet strict timing and security requirements.

To support cryptographic flexibility in this environment, the cryptographic API must be accessible to privileged code within the RTOS. However, in many embedded systems, only a minimal set of cryptographic primitives is included in the RTOS, and the set is selected based on the system memory requirements, real-time constraints, and security profile. In these cases, available algorithms are typically fixed at compile time, and the dynamic loading of new cryptographic modules after deployment is often not supported due to reliability and certification concerns. Designers should consider the need to update the cryptographic implementation together or separately from the rest of the system.

Similarly, secure boot mechanisms (i.e., starting a computer and loading its operating system) and firmware authentication routines operate as part of the system startup sequence before the main application tasks begin. Some embedded platforms also include cryptographic self-tests during the startup process to validate the integrity and correct functioning of the cryptographic operations, ensuring that any malfunction or tampering is detected before normal system execution.

4.5. Hardware

There are several aspects of the hardware implementation of cryptographic algorithms to consider with regard to crypto agility. For example, an entire integrated circuit chip might be dedicated to the implementation of one cryptographic algorithm, or a small portion of a chip might implement a particular building-block function in support of a single cryptographic algorithm. In either case, a low-level interface is needed that works well in a particular hardware environment. Firmware is often needed to manage memory and invoke various low-level functions in the proper order. The functions that are implemented in the integrated circuit cannot be changed. This makes them well-protected from attackers, but it also means that the chip will need to be replaced if it has design errors or if changes to the algorithms are needed.

Some chips are dedicated to supporting cryptographic operations, such as universal integrated circuit cards (UICCs) and TPMs. These chips are part of a larger computer system like a mobile phone, laptop computer, or server. The chips store private keys and support functionality to perform cryptographic operations that use the keys. For removable UICCs, the private keying material is never expected to leave the chip, except at the time of manufacture. These chips support a limited set of defined cryptographic algorithms, and changing supported algorithms is often accomplished by replacing the removable UICC or upgrading the device because non-removable UICCs and TPMs are purposefully difficult to replace. There are some cases in which changing the supported algorithms by upgrading the functionality running on the device is possible. In fact, some devices offer a slot to do so without opening the device. Field-programmable gate arrays (FPGAs) are an exception to this immutable property because this type of integrated circuit does support the ability to reconfigure the hardware after deployment.

Hardware security modules (HSMs) are special-purpose hardware devices that store private keys and perform cryptographic operations using those keys. An HSM might be a rack-mounted device for an organization, a high-value application key store, or a portable device that is easily locked in a safe when not in use. The private keying material never leaves the HSM. There are operations to securely back up the private keying material to another HSM.[2] HSMs offer tamper-detection capabilities to protect the private keying material stored in them. They often

[2] HSMs provide cryptographic services, but some also consume cryptographic services (e.g., leveraging APIs for cryptographic operations but still providing hardware-based protection for key storage).

include a microprocessor and one or more chips that are designed to accelerate different cryptographic algorithms or parts of the algorithms.

A personal portable cryptographic token (e.g., Personal Identity Verification [PIV] card, USB token) is a device that stores the private keys for an individual. The human user inserts the portable device into the computer they are currently using. These devices are essentially tiny HSMs that are intended to be used by one person, and the keying material never leaves the portable device.

Some central processing units (CPUs) have instructions that are designed to accelerate specific algorithms. A cryptographic algorithm implementation might detect whether such instructions are available and then take advantage of them if they are. For example, the Intel SHA Extensions paper [30] states that Intel-based CPUs offer features to make SHA hash computations faster. Scalable Vector Extension (SVE) and RISC-V Vector Extension (RVV) are also available that speed up SHA-3 [31][32][33][34].

Some hardware offers a good source of random numbers [35][36][37], which are vital to the generation of quality keying material. However, it is easier to provide multiple cryptographic algorithms to facilitate agility in library and application software than in hardware. Once a chip leaves the factory, additional algorithms may not be easily added to the chip. Other layers in an architecture fall on a spectrum between these two cases. The crypto API needs to be designed so that all points on this spectrum are accommodated. In some environments, especially HSMs and other cryptographic tokens, the data needs to move to the device where the key is stored for the data to be protected using that key.

For environments in which the update of cryptographic functions in hardware is not possible, the best and most conservative variants for each cryptographic function can be implemented to future-proof the implementation. A key element is the communication between cryptographers and developers to decide on a long-term plan based on the best estimate of the security needs during the lifetime of a specific hardware device. For example, secure booting requires the use of digital signature schemes. The public key and the program for verifying the signatures are included in the boot code and cannot be updated. In this case, to make sure that the platform is trustable during its lifetime, the signature schemes must be able to provide the required security during the lifetime of the device.

4.6. Using a Crypto Gateway for Legacy Systems

Legacy or monolithic mission-critical systems often have cryptographic functions tightly embedded within their components. Such systems were developed with obsolete technologies, and the original personnel are no longer available. The absence of abstraction between cryptographic operations and core system logic hampers the ability to support crypto agility.

To address risks posed by vulnerable or disallowed cryptography across such systems, an architectural solution known as a “crypto gateway” or “bump-in-the-wire” can be implemented. This approach introduces a dedicated cryptographic gateway to intercept and perform cryptographic functions externally to one or more system components. This often results in modern cryptographic solutions that encapsulate the vulnerable cryptography that was built into the legacy systems. By centralizing cryptographic operations within this gateway, organizations can enforce updated cryptographic policies, rotate keys, and deploy new algorithms uniformly across the system without modifying or redeploying individual legacy components. This approach can help provide crypto agility for mission-critical systems if direct modifications to embedded cryptographic code and functions are not feasible.

5. Crypto Agility Strategic Plan for Managing Organizations' Crypto Risks

Numerous strategies and frameworks (e.g., the Crypto Agility Risk Assessment Framework [CARAF] [38]) have been developed to address the risks that arise from insufficient crypto agility at the organization level. This paper presents a proposed approach that has been introduced as part of the NIST NCCoE Migration to Post-Quantum Cryptography project [39].

Organizations need to transition or migrate their cryptographic algorithms multiple times throughout the lifetimes of their IT systems. By incorporating crypto agility into their cryptographic policies during the design and development of the systems and technology acquisitions or scheduled replacements, updates, or modernization efforts, organizations can proactively address emerging threats, technological advances, system weaknesses, and evolving business requirements, standards, regulations, and mandates. A crypto agility strategic plan combines key functions to inform the migration/transition of cryptography at different technology levels, including governance [32], cryptographic and data assets, risk management, and automated tooling (see Fig. 3).

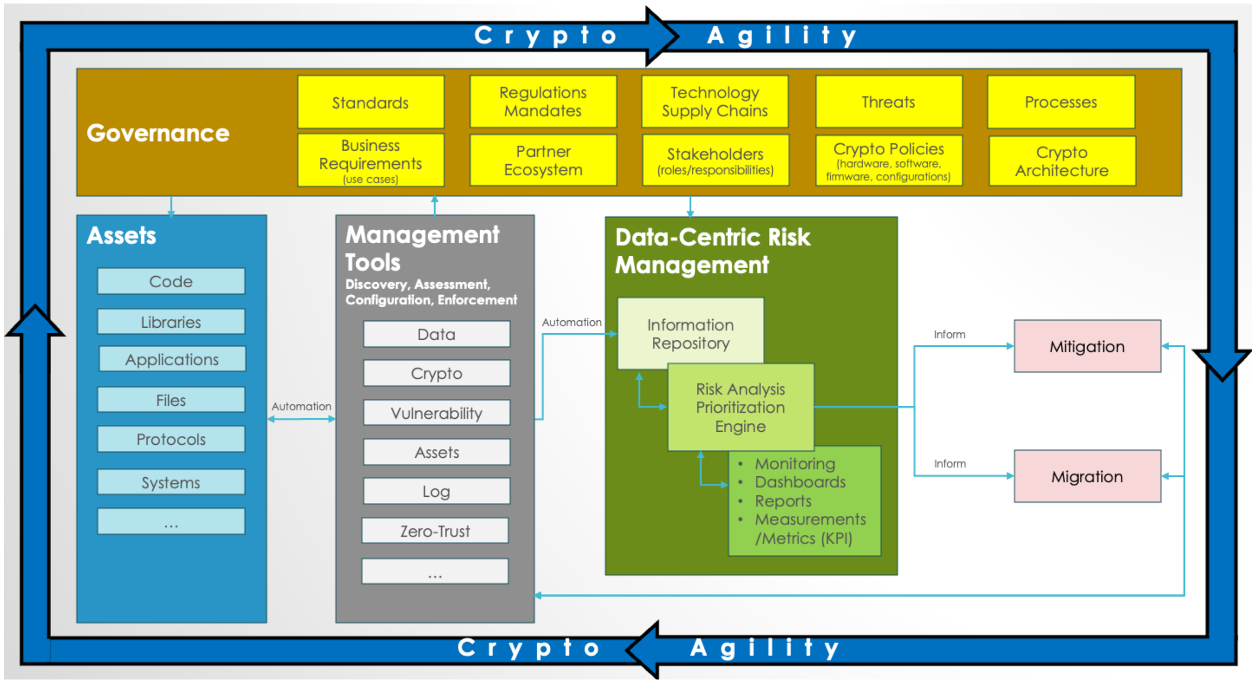

Fig. 3. Crypto agility strategic plan for managing an organization's cryptographic risks

The plan may include several key activities:

- Integrate crypto agility into the organization's existing governance function to establish, communicate, and monitor the cybersecurity risk management strategy, expectations, and policies related to cryptography. This includes understanding cryptographic standards, regulations, and mandates and communicating these requirements to data owners, IT and development teams, business partners, and technology supply chain vendors prioritized by the criticality of the data for the primary use cases. Identify key stakeholders, including a responsible official to lead the effort and monitor the progress of the activities.

- Inventory the use of cryptography for data protection across the organization by adopting an assets-centric approach informed by the criticality of the data to identify the organization’s use cases and most valuable assets, such as application codes, libraries, software, hardware, firmware, user-generated content, communication protocols, enterprise services, and systems.
- Identify gaps in enterprise management tools for managing assets, configurations, vulnerabilities, and logs. These tools should support cryptographic risk management and data protection functions by automating the identification, assessment, characterization, enforcement, and monitoring of cryptography across assets. If necessary, enhance the tools with automated data and cryptographic discovery capabilities, including algorithms and key lengths. For instance, vulnerability management and software/hardware development tools can help ensure comprehensive visibility and an inventory of assets, such as code, libraries, applications, and associated cryptographic algorithms.
- Develop a prioritized list of assets to be mitigated based on the data collected in the previous steps. A cryptographic policy-informed risk assessment engine should be used to analyze this data, form an implementation strategy, and recommend actions to reduce risks. Based on the organization’s defined cryptographic policy, the engine will continuously measure, monitor, and report on the state of cryptography within the enterprise and focus on crypto agility key performance indicators (KPIs) for the level of effort needed to effectively and efficiently adapt and migrate. This may include the time, cost, and ease to migrate and mitigate in accordance with technology refresh cycles.
- Implement the strategy and actions based on the prioritized list of assets. The level of crypto agility will determine whether assets can be smoothly migrated or compensatory mitigation measures must be implemented to reduce risks. Organizations can use existing automated enterprise management tools when feasible to inventory, assess, and migrate assets (e.g., code, applications, software, hardware, and communication protocols) or implement compensating security controls as part of a zero-trust approach [40] if the assets are not agile enough to support the cryptographic policy.

These steps are continuously repeated to mitigate evolving cryptographic risks and progressively strengthen the crypto agility posture within organizations over time.

Cryptography governance is an important function of a crypto agility strategic plan. The following subsections discuss some components of governance that are crucial for organizations to drive cryptographic practices and compliance in support of managing the cryptographic risks among all stakeholders, from the organization’s board to system implementers.

5.1. Cryptographic Standards, Regulations, and Mandates

Any crypto agility effort must consider the effects of standards, regulations, and mandates on transition requirements for cryptographic algorithms. Movements to achieve crypto agility involve coordination between protocol designers, software and hardware vendors, application and standards developers, policymakers, and IT administrators. Government standards and regulations can mandate transition when an algorithm is found to be vulnerable. SP 800-131A guides algorithm and security strength transitions by setting transition schedules for implementers to terminate certain algorithms or security strengths based on a common understanding of the computing power available to attackers and the latest research results. For example, SP 800-131Ar2 (Revision 2) [41] set the end of 2023 as the date to disallow three-key Triple DES for applying cryptographic protection. These transition guidelines are informed by various stakeholders, including cryptographic researchers and designers, cryptanalysts, policymakers, regulators, SDOs, and technology providers.

Industry standards play an important role in compliance with security requirements for cryptographic algorithm use in different application environments. The standards for internet protocols, communications, and applications update the supported cipher suites to eliminate vulnerable algorithms and ciphers. Security protocols often define mandatory-to-implement cipher suites to reflect state-of-the-art cryptography and support interoperability.

The NIST Cryptographic Algorithm Validation Program (CAVP) provides validation testing for FIPS-approved and NIST-recommended cryptographic algorithms. Cryptographic algorithm validation is a prerequisite for cryptographic module validation. The approved algorithms and relevant parameter sets are updated based on transition requirements [41].

From a practitioner's perspective, certain policies, laws, and mechanisms must be established to enhance crypto agility practices, facilitate transitions, and provide proper security during the transition. These laws and policies are coupled with industry-specific requirements. It is very important to handle assets in a secure way during a transition. For example, for the encrypted storage of data at rest, a mechanism must be established to handle encrypted user data when the encryption algorithm is to be replaced by a stronger one. Similarly, when a digital signature algorithm must be replaced, a mechanism to handle already-signed documents using the algorithm to be replaced is required.

5.2. Crypto Security Policy Enforcement

The crypto agility assessment must consider cryptographic security policy establishment and enforcement for each protocol, system, and application. One of the most challenging aspects of crypto agility is replacing vulnerable algorithms in a timely manner without interrupting the operation of the system. For security protocols, a cryptographic security policy can be enforced by specifying mandatory-to-implement algorithms and disallowing the use of vulnerable algorithms in a timely fashion. For an application, a security policy can be enforced by using an API, as represented in Fig. 2.

Enforcing a cryptographic security policy requires communication among cryptographers, developers, practitioners, implementers, and policymakers. Each decision on deprecating a cryptographic algorithm must be synchronized among all of the stakeholders so that the security policy can be updated quickly and translated into a technology-specific, machine-consumable configuration profile that represents a crypto policy that can be deployed with automated tools.

5.3. Technology Supply Chains

Technology supply chains play a critical role in the governance function of a crypto agility strategic plan. They guide decisions about whether to migrate to new cryptographic systems or employ mitigation techniques when cryptographic changes are necessary. This involves examining the impacts of the supply chain on the entire cryptographic architecture, including hardware, firmware, software modules, and communication protocols.

A resilient technology supply chain enables modular updates that allow cryptographic algorithms to be replaced or upgraded seamlessly due to emerging vulnerabilities of the crypto algorithms without overhauling the entire system. This approach minimizes disruptions and ensures continuous security.

Crypto agility requires all system components in a supply chain to work in harmony during updates. The supply chains have dependencies on the maturity of standards, protocols, and the cryptographic validation program before the products and services can be delivered to the implementer.

Technology producers can help an organization by providing automated mechanisms that have visibility into products, services, and protocols to include a comprehensive list of cryptographic components [35], such as algorithms, protocols, libraries, applications, certificates, and related crypto materials. This will inform whether the cryptographic components can be updated without a complete system overhaul or need to be replaced if vulnerabilities are discovered or new cryptographic algorithms are introduced due to emerging threats.

When systems reach their end-of-life stage and their cryptographic components no longer comply with organizational cryptographic policies, the organization implements the defined mitigation strategies and begins the replacement process as outlined in its policy.

5.4. Cryptographic Architecture

Network architecture deals with data flows across the interfaces, protocols, and physical and logical communication components of an enterprise architecture. Another closely related concept is the cryptographic architecture, which focuses on how assets (including data) are protected and ensures data integrity and authentication for data at rest, in transit, and in use. It defines the design, management, and deployment of cryptographic controls, such as encryption algorithms, key management, digital signatures, and secure protocols.

The cryptographic architecture in a crypto agility strategic plan provides the technical foundation upon which governance functions are built. The cryptographic architecture creates a structured framework by defining standardized processes, protocols, and key management practices that govern how cryptographic updates are implemented and maintained throughout an organization. In essence, a cryptographic architecture is part of the organization governance function for capturing how an organization integrates and manages cryptographic functions to secure its assets and communications. The architecture establishes the design principles, standards, and processes for implementing and maintaining cryptographic services, key management, and related security mechanisms in libraries, software, hardware, and firmware. It captures how cryptographic components interact with each other and with other parts of the network architecture.

Organizations can include crypto agility characteristics as part of the cryptographic architecture to capture cryptographic standards, policies, algorithms, protocols, key management practices, and security components. This helps an organization decide whether to replace or upgrade cryptographic algorithms in a timely manner when new vulnerabilities or threats emerge or in support of an organization-defined cryptographic policy.

6. Considerations for Future Works

Achieving crypto agility demands collaboration and communication among cryptographic researchers, software and standards developers, protocol designers and implementers, hardware designers, and practitioners to manage the risks of using cryptography to secure data. To be actionable, crypto agility requirements must be specific for each implementation and application environment. This section discusses some trade-offs, and each subsection highlights important areas for consideration by the relevant stakeholders.

6.1. Resource Considerations

Resource limitation is the most difficult challenge for achieving crypto agility. This section discusses resource considerations for protocol designers, hardware implementers, and cryptographers.

Crypto agility requires support for multiple cryptographic algorithms in a protocol. Some algorithms have much larger public keys, signatures, or ciphertext than the algorithms being replaced. Experience has shown that large sizes challenge the limits of existing protocols. It is important for protocol designers to consider resource demands to plan for future transitions and to distinguish intrinsic limitations from shortsighted design decisions.

Hardware implementation is limited by capacity. It may not be possible to implement many algorithms in one hardware platform. Some optimization efforts (e.g., accelerator reuse) have been considered. At present, further research is needed in this area to address the transition from traditional public-key cryptography to PQC.

Future cryptographic algorithm designs must consider resource limitations. Historically, each design has focused on the resource requirements of a single algorithm for an application without considering the other algorithms used by the application. For example, the design may use a specific primitive or a subroutine (e.g., a hash function) that is not commonly used by other applications. To save hardware resources, it is desirable for different algorithms to share the same primitive or subroutine.

Cryptographers have considered algorithms based on diverse assumptions so that when one assumption is determined to be incorrect, an alternative algorithm based on a different assumption is in place for the same purpose. Achieving crypto agility within resource limitations requires cryptographers to prioritize security-related diversities.

Considering the combination of algorithms used by different applications in a device is a new area of research to optimize resource use. It must take a different approach from that of traditional approaches where the resource needed for an algorithm is viewed in isolation from the need for other algorithms to be used by applications.

6.2. Agility-Aware Design

This section discusses crypto agility design considerations for applications, platforms, and protocol designs.

To achieve cryptographic agility, a product or system must be designed to accommodate new algorithms and parameters. This requires a user interface (UI) and API that can support varying key and parameter sizes for underlying cryptographic libraries and hardware accelerators. The design should avoid making assumptions based on specific algorithms to ensure that buffers, memory, and storage can handle diverse cryptographic data.

Some well-deployed security protocols (e.g., TLS) facilitate authenticated cipher suite negotiation to allow adding new algorithms to and discontinuing the use of vulnerable algorithms from the available cipher suites. It should be a common practice in any protocol design to facilitate secure transition and avoid inflexible implementations.

It is also important to include crypto agility as an evaluation perspective for project proposals, security architects, protection profiles, protocol specifications, and application designs. For example, in most of the IETF Requests for Comment (RFCs), there is a section called "Security Considerations." It may be beneficial to include a discussion about "Crypto Agility Considerations" in the standards to provide a rationale for the design choices to allow crypto agility.

6.3. Complexity and Security

Accommodating crypto agility introduces complexity into protocols and systems that protocol designers and system architects and implementers should take into consideration. It can also increase attack surfaces. For example, if cipher suite negotiation integrity is not properly protected, a downgrade attack can lead to a vulnerable cipher suite than would otherwise be agreed upon. For software libraries and APIs, a larger number of options may increase the chance to introduce security vulnerabilities or attack vectors. For enterprise IT administrators, it is important to make sure that the configuration is updated to reflect current security requirements.

Crypto agility requires sound mechanisms to ensure a secure and smooth transition. Currently, most security analyses and evaluations of a protocol or system configuration are based on selected cryptographic algorithms without considering transition mechanisms. For a protocol, a cryptographic transition mechanism will facilitate the communication parties securely agreeing upon a cipher suite that satisfies updated security requirements. For a system configuration, a cryptographic transition mechanism enables applications to securely switch from a vulnerable algorithm to a secure algorithm.

6.4. Crypto Agility in the Cloud

This section discusses crypto agility considerations for cloud computing service architects, developers, operators, and cryptographers.

The main security model used in the cloud is the shared responsibility model, which clearly divides security duties between the cloud provider and the customer. The cloud provider secures the underlying infrastructure, including physical facilities, hardware, networking, and virtualization. The customer manages the security of their data, applications, and configurations. To ensure comprehensive and compliant cloud security, the shared responsibilities vary by service model, such as infrastructure as a service (IaaS), platform as a service (PaaS), and software as a service (SaaS).

Cloud providers are responsible for ensuring the agility of the cryptographic hardware, such as HSM, hardware root of trust, hardware-enabled security functions like encrypted memory, and services like secure runtime environment, attestation service, crypto library, container services and images, secure communication, data protection, authentication, key management. All of these capabilities are made accessible to customers through well-designed and robust APIs.

Cloud providers demonstrate their adherence to security best practices by undergoing third-party assessments, such as FedRAMP [42] certification. Customers, in turn, can leverage this crypto agility within cloud platforms to enhance application resiliency and potentially lower maintenance and support costs by decoupling cryptographic functions from their core application logic. Cloud providers offer APIs to make their cryptographic functions and configurations transparent to customers.

While the range of cryptographic hardware, features, and services that a provider supports may limit application portability across different platforms, customers still maintain complete control over keys that are managed within cloud-based HSMs or secure runtime environments. Although cloud providers cannot access customers' keys, they can manage cryptographic resource use by controlling the underlying infrastructure. Cloud providers bear the capital and operational costs of these services, balancing them with diverse and often conflicting national or industry-specific compliance cryptographic policies. In the IaaS model, customers have more control, but they are responsible for the agility of their cryptographic functions when they choose to manage their own hardware (e.g., HSMs) or cryptographic applications within the cloud or when integrating cloud-based applications and services that employ on-premises cryptographic services.

6.5. Maturity Assessment for Crypto Agility

This section introduces the concept of a crypto agility maturity model to help organizations continuously measure their progress in adopting crypto agility across their environments and achieve resilience against evolving changes in crypto requirements.

Since organizations vary significantly in their mission, size, sector, country, regulatory regime, and risk tolerance, there is not a single crypto maturity model that effectively serves all needs. Instead, organizations should adapt their existing, mature risk management frameworks to include crypto agility. Enterprise risk management frameworks often incorporate a maturity model concept, and these can be adapted to assess and report on crypto agility. This approach utilizes a shared vocabulary for effective communication within the organization, with external partners, and with suppliers. It integrates directly with the organization's current risk management processes and streamlines the evaluation of cryptographic agility readiness.

The maturity model described in this paper is derived from the NIST Cybersecurity Framework (CSF) [43], which is a voluntary set of guidelines based on standards and best practices designed for managing and reducing cybersecurity risks. The CSF has four tiers that show increasing sophistication in cybersecurity risk management:

- Tier 1 – Partial: An initial, informal, and reactive approach
- Tier 2 – Risk-Informed: Management-approved but not fully integrated practices
- Tier 3 – Repeatable: Standardized and consistently updated processes
- Tier 4 – Adaptive: Proactive, continuously improved, and dynamic approach to cybersecurity based on evolving risks

Adapting the CSF tiers and drawing upon insights from various industry initiatives [44][45][46][47], a crypto agility maturity model might include:

- Tier 1 – Partial
    - Crypto agility practices are unstructured and unplanned.
    - Each group or team within the organization implements its own cryptography on a case-by-case basis.
    - The selection of cryptographic algorithms, schemes, libraries, and cryptographic products and services is not informed by current cryptographic-based exploits and evolving threat landscapes.
    - The organization is unaware of potential cryptographic risks from partners, suppliers, and acquired products and services.
    - Organizational and external awareness (including partners and suppliers) of cryptography usage is limited.
    - A formal cryptographic policy or architecture is lacking, hindering internal and external communication.
    - Discussions about the organization's crypto agility are infrequent and inconsistent.
- Tier 2 – Risk-Informed
    - The crypto agility strategy and plan include a cryptographic policy that has been defined and approved by management but has not been adopted consistently as an organization-wide policy.
    - The cryptographic policy is shaped by established standards, approved or validated technologies, business requirements, existing processes and procedures, and stakeholder input.
    - Crypto agility prioritization is determined and refined through risk assessments.
    - A cryptographic architecture is being developed and informed by inventories of cryptographic assets, data, and external dependencies.

- Tier 3 – Repeatable
  - Crypto agility is formally integrated into the organization's risk management plan and is guided by a well-defined cryptographic policy.
  - The cryptographic policy, processes, procedures, roles, and responsibilities are defined, implemented, reviewed, and assessed.
  - Crypto agility practices and cryptographic architectures are regularly updated based on changes in business and mission requirements, threats, and technological evolution.
  - Crypto agility is part of the organizational awareness and training curriculum to ensure that personnel, particularly developers, have the appropriate knowledge and skills.
  - Integrated and automated cryptographic discovery and remediation tools are used to prioritize and continuously mitigate cryptographic risks.
  - Crypto-agility practices are continuously tested to measure their impact on organizational processes in support of the mission and to ensure that the organization is adequately prepared to activate them when necessary.
- Tier 4 – Adaptive
  - Crypto agility is a fundamental element of organizational risk management with defined objectives.
  - Crypto agility is monitored, measured, and reported to executives as part of the risk register and linked to financial, business, and mission objectives.
  - Crypto agility is considered in all changes to business objectives at the executive level and in every line of code by developers.

Crypto agility policies, processes, and procedures are continuously adapted, monitored, and communicated in near real-time in response to changes in the environment, such as standards, regulations, supply chains, partners' ecosystems, mission and business requirements, and threat and technological landscapes.

6.6. Common Crypto API

One of the needs identified during the NIST Crypto Agility Workshop was a common cryptographic API — a universal interface that bridges established cryptographic API frameworks by abstracting complex cryptographic operations to support crypto agility. An effective solution must balance generality with specificity by including the essential functions required for operational use, interoperability, and smooth transitions to different algorithms without being hindered by the specific characteristics of individual cryptographic implementations.

NIST collaborates with the cryptographic community to develop standards and guidelines, and industry-led SDOs define mechanisms for supporting the cryptographic standards (e.g., for software, hardware, firmware, protocols). Industry partners — in collaboration with government experts, SDOs, and academic researchers — are in the best position to research and initiate the development of a common cryptographic API. This can be done by a consortium of practitioners through a series of iterative discussions, workshops, and prototype implementations to define operational use cases and the associated minimum set of requirements for a common API that can be backward compatible with existing widely deployed cryptographic APIs and support emerging cryptographic functions and algorithms.

7. Conclusion

Crypto agility is a future-proofing strategy to address changes in cryptographic algorithms. It demands communication among cryptographers, protocol designers, developers, implementers, and practitioners to accommodate evolving security, performance, and interoperability challenges. The pursuit of crypto agility capabilities involves the exploration of new technologies and management schemes, and new crypto agility requirements must be developed for each environment. The security analysis and evaluation of protocols, systems, and applications must include mechanisms for transitions. When transition mechanisms are not available, organizations should have a plan to implement compensating controls to mitigate cryptographic vulnerabilities and evolving threats. Although crypto agility is now being considered in security practices to facilitate transitions, achieving measurable maturity in this area requires ongoing and significant effort.

References

[1] OpenSSL 3.5.0-alpha1. Available at https://github.com/openssl/openssl/releases/tag/openssl-3.5.0-alpha1

[2] National Institute of Standards and Technology (1999) Data Encryption Standard (DES). (U.S. Department of Commerce, Washington, DC), Federal Information Processing Standards Publication (FIPS) NIST FIPS 46-3. Withdrawn May 19, 2005. Available at https://csrc.nist.gov/pubs/fips/46-3/final

[3] National Institute of Standards and Technology (2001) Advanced Encryption Standard (AES). (U.S. Department of Commerce, Washington, DC), Federal Information Processing Standards Publication (FIPS) NIST FIPS 197-upd1, updated May 9, 2023. https://doi.org/10.6028/NIST.FIPS.197-upd1

[4] National Institute of Standards and Technology (2015) Secure Hash Standard (SHS). (U.S. Department of Commerce, Washington, DC), Federal Information Processing Standards Publication (FIPS) NIST FIPS 180-4. https://doi.org/10.6028/NIST.FIPS.180-4

[5] Wang X, Yin YL, Yu H (2005) Finding Collisions in the Full SHA-1. Advances in Cryptology — CRYPTO 2005. Lecture Notes in Computer Science, vol. 3621. (Springer, Berlin, Heidelberg). https://doi.org/10.1007/11535218_2

[6] Rescorla E (2018) The Transport Layer Security (TLS) Protocol Version 1.3. (Internet Engineering Task Force (IETF)), IETF Request for Comments (RFC) RFC 8446. https://doi.org/10.17487/RFC8446

[7] Leurent G, Peyrin T (2020). SHA-1 is a Shambles - First Chosen-Prefix Collision on SHA-1 and Application to the PGP Web of Trust. SEC'20: Proceedings of the 29th USENIX Conference on Security Symposium. Available at https://eprint.iacr.org/2020/014

[8] National Institute of Standards and Technology (2022) NIST Transitioning Away from SHA-1 for All Applications. Available at https://csrc.nist.gov/news/2022/nist-transitioning-away-from-sha-1-for-all-apps

[9] National Institute of Standards and Technology (2000) Digital Signature Standard (DSS). (U.S. Department of Commerce, Washington, DC), Federal Information Processing Standards Publication (FIPS) NIST FIPS 186-2. Withdrawn October 5, 2001. Available at https://csrc.nist.gov/pubs/fips/186-2/final

[10] Barker E (2020) Recommendation for Key Management: Part 1 – General. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) NIST SP 800-57pt1r5. https://doi.org/10.6028/NIST.SP.800-57pt1r5

[11] Moody D, Perlner R, Regenscheid A, Robinson A, Cooper D (2024) Transition to Post-Quantum Cryptography Standards. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Internal Report (IR) NIST IR 8547 ipd. https://doi.org/10.6028/NIST.IR.8547.ipd

[12] National Institute of Standards and Technology (2024) Module-Lattice-Based Digital Signature Standard. (U.S. Department of Commerce, Washington, DC), Federal Information Processing Standards Publication (FIPS) NIST FIPS 204. https://doi.org/10.6028/NIST.FIPS.204

[13] Dierks T, Rescorla E (2008) The Transport Layer Security (TLS) Protocol Version 1.2. (Internet Engineering Task Force (IETF)), IETF Request for Comments (RFC) RFC 5246. https://doi.org/10.17487/RFC5246

[14] Kaufman C, Hoffman P, Nir Y, Eronen P, Kivinen T (2014) Internet Key Exchange Protocol Version 2 (IKEv2). Internet Engineering Task Force (IETF). Request for Comments (RFC) RFC 7296. https://doi.org/10.17487/RFC7296

[15] Ramsdell B (2004) Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification. (Internet Engineering Task Force (IETF)), IETF Request for Comments (RFC) RFC 3851. https://doi.org/10.17487/RFC3851

[16] Ramsdell B, Turner S (2010) Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2. (Internet Engineering Task Force (IETF)), IETF Request for Comments (RFC) RFC 5751. https://doi.org/10.17487/RFC5751

[17] Housley R (2009) Cryptographic Message Syntax (CMS). (Internet Engineering Task Force (IETF)), IETF Request for Comments (RFC) RFC 5652. https://doi.org/10.17487/RFC5652

[18] Crocker S, Rose S (2013) Signaling Cryptographic Algorithm Understanding in DNS Security Extensions (DNSSEC). (Internet Engineering Task Force (IETF)), IETF Request for Comments (RFC) RFC 6975. https://doi.org/10.17487/RFC6975

[19] Gagliano R, Kent S, Turner S (2013) Algorithm Agility Procedure for the Resource Public Key Infrastructure (RPKI). (Internet Engineering Task Force (IETF)), IETF Request for Comments (RFC) RFC 6916. https://doi.org/10.17487/RFC6916

[20] Popov A (2015) Prohibiting RC4 Cipher Suites. (Internet Engineering Task Force (IETF)), IETF Request for Comments (RFC) RFC 7465. https://doi.org/10.17487/RFC7465

[21] Barker E, Roginsky A (2019) Transitioning the Use of Cryptographic Algorithms and Key Lengths. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) NIST SP 800-131Ar2. https://doi.org/10.6028/NIST.SP.800-131Ar2

[22] Cooper D, Santesson S, Farrell S, Boeyen S, Housley R, Polk W (2008) Internet X.509 Public Key Infrastructure Certification and Certificate Revocation List (CRL) Profile. (Internet Engineering Task Force (IETF)), IETF Request for Comments (RFC) RFC 5280. https://doi.org/10.17487/RFC5280

[23] Dierks T, Allen C (1999) The TLS Protocol Version 1.0. (Internet Engineering Task Force (IETF)), IETF Request for Comments (RFC) RFC 2246. https://doi.org/10.17487/RFC2246

[24] Barker EB, Chen L, Roginsky AL, Vassilev A, Davis R (2018) Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) NIST SP 800-56Ar3. https://doi.org/10.6028/NIST.SP.800-56Ar3

[25] National Institute of Standards and Technology (2024) Module-Lattice-Based Key-Encapsulation Mechanism Standard. (U.S. Department of Commerce, Washington, DC), Federal Information Processing Standards Publication (FIPS) NIST FIPS 203. https://doi.org/10.6028/NIST.FIPS.203

[26] Alagic G, Barker EB, Chen L, Moody D, Robinson A, Silberg H, Waller N (2025) Recommendations for Key-Encapsulation Mechanisms. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) NIST SP 800-227. https://doi.org/10.6028/NIST.SP.800-227

[27] Aboba B, Blunk L, Vollbrecht J, Carlson J, Levkowetz H (2004) Extensible Authentication Protocol (EAP). (Internet Engineering Task Force (IETF)), IETF Request for Comments (RFC) RFC 3748. https://doi.org/10.17487/RFC3748

[28] Dworkin MJ (2004) Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) NIST SP 800-38C, Includes updates as of July 20, 2007. https://doi.org/10.6028/NIST.SP.800-38C

[29] Dworkin MJ (2007) Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) NIST SP 800-38D. https://doi.org/10.6028/NIST.SP.800-38D

[30] Gulley S, Gopal V, Yap K, Feghali W, Guilford J, Wolrich G (2013) Intel SHA Extensions: New Instructions Supporting the Secure Hash Algorithm on Intel Architecture Processors. (Intel Corporation.) Available at https://www.intel.com/content/dam/develop/external/us/en/documents/intel-sha-extensions-white-paper-402097.pdf

[31] Rott JK (2012) Intel Advanced Encryption Standard Instructions (AES-NI). Available at https://www.intel.com/content/www/us/en/developer/articles/technical/advanced-encryption-standard-instructions-aes-ni.html

[32] Arm (2025) Crypto plug-in. Available at https://developer.arm.com/documentation/100964/1129/Plug-ins-for-Fast-Models/Crypto

[33] Arm (2025) Cryptographic extensions. Available at https://developer.arm.com/documentation/101754/0624/armclang-Reference/Other-Compiler-specific-Features/Supported-architecture-features/Cryptographic-extensions

[34] RISC-V (2021) RISC-V Cryptography Extensions Task Group Announces Public Review of the Scalar Cryptography Extensions. Available at https://riscv.org/blog/2021/09/risc-v-cryptography-extensions-task-group-announces-public-review-of-the-scalar-cryptography-extensions/

[35] ISO Central Secretary (2011) ISO/IEC 18031:2011 Information technology — Security techniques — Random bit generation (International Organization for Standardization, Geneva, CH), Standard ISO/IEC 18031:2011. Available at https://www.iso.org/standard/54945.html

[36] Barker EB, Kelsey JM (2015) Recommendation for Random Number Generation Using Deterministic Random Bit Generators. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) NIST SP 800-90Ar1. https://doi.org/10.6028/NIST.SP.800-90Ar1

[37] Sönmez Turan M, Barker EB, Kelsey JM, McKay KA, Baish ML, Boyle M (2018) Recommendation for the Entropy Sources Used for Random Bit Generation. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) NIST SP 800-90B. https://doi.org/10.6028/NIST.SP.800-90B

[38] CARAF: Crypto Agility Risk Assessment Framework. Available at https://doi.org/10.1093/cybsec/tyab013

[39] NIST NCCoE Migration to Post-Quantum Cryptography project. Available at https://www.nccoe.nist.gov/crypto-agility-considerations-migrating-post-quantum-cryptographic-algorithms
[40] Rose SW, Borchert O, Mitchell S, Connelly S (2020) Zero Trust Architecture. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) NIST SP 800-207. https://doi.org/10.6028/NIST.SP.800-207
[41] Barker EB, Roginsky AL (2019) Transitioning the Use of Cryptographic Algorithms and Key Lengths. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) NIST SP 800-131Ar2. https://doi.org/10.6028/NIST.SP.800-131Ar2
[42] FedRAMP. Available at https://www.fedramp.gov/
[43] National Institute of Standards and Technology (2024) The NIST Cybersecurity Framework (CSF) 2.0. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Cybersecurity White Paper (CSWP) NIST CSWP 29. https://doi.org/10.6028/NIST.CSWP.29
[44] FS-ISAC (2024) Building Cryptographic Agility in the Financial Sector. Available at https://www.fsisac.com/hubfs/Knowledge/PQC/BuildingCryptographicAgilityInTheFinancialSector.pdf
[45] Hohm J, Heinemann A, Wiesmaier A (2022) Towards a Maturity Model for Crypto-Agility Assessment. Available at https://arxiv.org/abs/2202.07645
[46] ATIS (2024) Strategic Framework for Crypto Agility and Quantum Risk Assessment. Available at https://atis.org/resources/strategic-framework-for-crypto-agility-and-quantum-risk-assessment/
[47] Deloitte (2025) Cryptographic Resilience: A Cyber Security Framework (CSF) 2.0 Community Profile. Available at https://www.deloitte.com/content/dam/assets-shared/docs/services/consulting/2025/deloitte-cryptographic-resilience-community-profile-april-2025.pdf
[48] National Academies of Sciences, Engineering, and Medicine (2016) Cryptographic Agility and Interoperability: Proceedings of a Workshop. Forum on Cyber Resilience Workshop Series. (The National Academies Press, Washington, DC). https://doi.org/10.17226/24636
[49] ETSI TR 103 619 V1.1.1 Cyber; Migration strategies and recommendations to Quantum Safe schemes. Available at https://www.etsi.org/deliver/etsi_tr/103600_103699/103619/01.01.01_60/tr_103619v010101p.pdf

Appendix A. List of Symbols, Abbreviations, and Acronyms

AEAD
Authenticated Encryption with Associated Data

AES
Advanced Encryption Standard

AES-CCM
Advanced Encryption Standard – Counter with CBC-MAC

AES-GCM
Advanced Encryption Standard – Galois/Counter Mode

API
Application Programming Interface

CA
Certification Authority

CAVP
Cryptographic Algorithm Validation Program

CISO
Chief Information Security Officer

CLI
Command Line Interface

CMS
Cryptographic Message Syntax

CPU
Central Processing Unit

CRQC
Cryptographically Relevant Quantum Computer

CSF
Cybersecurity Framework

CSP
Cryptographic Service Provider

DES
Data Encryption Standard

DNSSEC
Domain Name System Security Extensions

EAP
Extensible Authentication Protocol

ECDH
Elliptic Curve Diffie-Hellman

ECDSA
Elliptic Curve Digital Signature Algorithm

EDNS
Extension Mechanisms for Domain Name System

ESP
Encapsulating Security Payload

FIPS
Federal Information Processing Standards

HSM
Hardware Security Module

IaaS
Infrastructure as a Service

IETF
Internet Engineering Task Force

IKE
Internet Key Exchange

IPsec
Internet Protocol Security

IR
Internal Report

KEM
Key-Encapsulation Mechanism

KPI
Key Performance Indicator

MAC
Message Authentication Code

ML-DSA
Module-Lattice-Based Digital Signature Algorithm

ML-KEM
Module-Lattice-Based Key Encapsulation Mechanism

PaaS
Platform as a Service

PIV
Personal Identity Verification

PKI
Public Key Infrastructure

PQC
Post-Quantum Cryptography

PRF	Pseudorandom Function
PSK	Pre-Shared Key
RFC	Request for Comment
RPKI	Resource Public Key Infrastructure
RSA	Rivest-Shamir-Adelman
RTOS	Real-Time Operating System
RVV	RISC-V Vector Extension
SaaS	Software as a Service
SDO	Standards Developing Organization
SHA	Secure Hash Algorithm
SIM	Subscriber Identity Module
S/MIME	Secure Multipurpose Internet Mail Extensions
SP	Special Publication
SVE	Scalable Vector Extension
TLS	Transport Layer Security
TPM	Trusted Platform Module
UI	User Interface
UICC	Universal Integrated Circuit Card
USB	Universal Serial Bus

Appendix B. Definition of Crypto Agility in Other Literature

A 2016 NIST presentation [48] described crypto agility as:

- The ability for implementations to select from the available security algorithms in real time and based on their combined security functions;
- The ability to add new cryptographic features or algorithms to existing hardware or software, resulting in new, stronger security features; and
- The ability to gracefully retire cryptographic systems that have become either vulnerable or obsolete.

In [32], cryptographic agility for the financial sector is defined as:

> …a measure of an organization’s ability to adapt cryptographic solutions or algorithms (including their parameters and keys) quickly and efficiently in response to developments in cryptanalysis, emerging threats, technological advances, and/or vulnerabilities...a design principle for implementing, updating, replacing, running, and adapting cryptography and related business processes and policies with no significant architectural changes, minimal disruption to business operations, and short transition time.

In [46], the Alliance for Telecommunications Industry Solutions (ATIS) described crypto agility as “the ability of a system or organization to adapt and switch to different cryptographic primitives, algorithms, or protocols easily and efficiently with limited impact on operations and with low overhead.”

In [49], ETSI defines crypto agility as a “property that permits changing or upgrading cryptographic algorithms or parameters.”