NIST IR 8477-Based Set Theory Relationship Mapping (STRM)
Reference Document: Secure Controls Framework (SCF) version 2026.2
STRM Guidance: https://securecontrolsframework.com/start-here/set-theory-relationship-mapping-strm/

Focal Document: NIST CSWP 39 - Considerations for Achieving Crypto Agility
Focal Document URL: https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.39.pdf
Published STRM URL: https://content.securecontrolsframework.com/strm/scf-strm-general-nist-cswp-39.pdf

FDE #	FDE Name	Focal Document Element (FDE) Description	STRM Rationale	STRM Relationship	SCF Control	SCF #	Secure Controls Framework (SCF) Control Description	Strength of Relationship	Notes
1	Introduction	See FDE for details	Functional	No Relationship	N/A	N/A	N/A	0	
2	Historic Transitions and Challenges	See FDE for details	Functional	No Relationship	N/A	N/A	N/A	0	
2.1	Long Period for a Transition	See FDE for details	Functional	No Relationship	N/A	N/A	N/A	0	
2.2	Backward Compatibility and Interoperability Challenges	See FDE for details	Functional	No Relationship	N/A	N/A	N/A	0	
2.3	Constant Needs of Transition	See FDE for details	Functional	No Relationship	N/A	N/A	N/A	0	
2.4	Resource and Performance Challenges	See FDE for details	Functional	No Relationship	N/A	N/A	N/A	0	
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.	Functional	Intersects With	Post-Quantum Cryptography Agility Plan (PSCAP)	QTS-03	Mechanisms exist to develop a risk-prioritized Post-Quantum Cryptography Agility Plan (PQCAP) that: (1) Enables cryptographic agility; (2) Aligns with evolving security standards; and (3) Defines the approach for selecting and implementing PQC algorithms.	5	
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	Functional	Intersects With	Cryptographic Cipher Suites and Protocols Inventory	CRY-01.5	Mechanisms exist to identify, document and review deployed cryptographic cipher suites and protocols to proactively respond to industry trends regarding the continued viability of utilized cryptographic cipher suites and protocols.	5	
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.	Functional	Intersects With	Statutory, Regulatory & Contractual Compliance	CPL-01	Mechanisms exist to facilitate the identification and implementation of relevant statutory, regulatory and contractual controls.	5	
3.1.2	Dependent Specification	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-toimplement 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.	Functional	Intersects With	Statutory, Regulatory & Contractual Compliance	CPL-01	Mechanisms exist to facilitate the identification and implementation of relevant statutory, regulatory and contractual controls.	5	

NIST IR 8477-Based Set Theory Relationship Mapping (STRM)
Reference Document: Secure Controls Framework (SCF) version 2026.2
STRM Guidance: https://securecontrolsframework.com/start-here/set-theory-relationship-mapping-strm/

Focal Document: NIST CSWP 39 - Considerations for Achieving Crypto Agility
Focal Document URL: https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.39.pdf
Published STRM URL: https://content.securecontrolsframework.com/strm/scf-strm-general-nist-cswp-39.pdf

FDE #	FDE Name	Focal Document Element (FDE) Description	STRM Rationale	STRM Relationship	SCF Control	SCF #	Secure Controls Framework (SCF) Control Description	Strength of Relationship	Notes
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	Functional	No Relationship	N/A	N/A	N/A	0	
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 longlived 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	Functional	No Relationship	N/A	N/A	N/A	0	
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.	Functional	No Relationship	N/A	N/A	N/A	0	
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.	Functional	No Relationship	N/A	N/A	N/A	0	

NIST IR 8477-Based Set Theory Relationship Mapping (STRM)
Reference Document: Secure Controls Framework (SCF) version 2026.2
STRM Guidance: https://securecontrolsframework.com/start-here/set-theory-relationship-mapping-strm/

Focal Document: NIST CSWP 39 - Considerations for Achieving Crypto Agility
Focal Document URL: https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.39.pdf
Published STRM URL: https://content.securecontrolsframework.com/strm/scf-strm-general-nist-cswp-39.pdf

FDE #	FDE Name	Focal Document Element (FDE) Description	STRM Rationale	STRM Relationship	SCF Control	SCF #	Secure Controls Framework (SCF) Control Description	Strength of Relationship	Notes
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 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. In this section, we examine the hybrid approach of combining a traditional algorithm with a PQC algorithm. Figure 1 represents the different transition paths when using a hybrid algorithm to transit 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 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	Functional	No Relationship	N/A	N/A	N/A	0	
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 keyestablishment mechanism before it is used to produce cryptographic keys.	Functional	Intersects With	Cryptographic Key Management	CRY-09	Mechanisms exist to facilitate cryptographic key management controls to protect the confidentiality, integrity and availability of keys.	5	
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.	Functional	No Relationship	N/A	N/A	N/A	0	
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 minimumsecurity 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.	Functional	No Relationship	N/A	N/A	N/A	0	
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.	Functional	No Relationship	N/A	N/A	N/A	0	

NIST IR 8477-Based Set Theory Relationship Mapping (STRM)
Reference Document: Secure Controls Framework (SCF) version 2026.2
STRM Guidance: https://securecontrolsframework.com/start-here/set-theory-relationship-mapping-strm/

Focal Document: NIST CSWP 39 - Considerations for Achieving Crypto Agility
Focal Document URL: https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.39.pdf
Published STRM URL: https://content.securecontrolsframework.com/strm/scf-strm-general-nist-cswp-39.pdf

FDE #	FDE Name	Focal Document Element (FDE) Description	STRM Rationale	STRM Relationship	SCF Control	SCF #	Secure Controls Framework (SCF) Control Description	Strength of Relationship	Notes
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 welldocumented to reduce the risk of errors. Additionally, clear guidance must be provided for practitioners to follow.	Functional	Intersects With	Crypto-Agility Architecture	QTS-06	Mechanisms exist to validate design-level cryptographic agility across protocols, libraries, kernels and hardware to ensure Technology Assets, Applications and/or Services (TAAS) can support larger Post-Quantum Cryptography (PQC) key, signature and ciphertext sizes.	5	
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). 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.	Functional	No Relationship	N/A	N/A	N/A	0	
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.	Functional	No Relationship	N/A	N/A	N/A	0	
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.	Functional	No Relationship	N/A	N/A	N/A	0	
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 selftests 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.	Functional	Intersects With	Real-Time Operating System (RTOS) Security	EMB-18	Mechanisms exist to ensure embedded technologies utilize a securely configured Real-Time Operating System (RTOS).	5	

NIST IR 8477-Based Set Theory Relationship Mapping (STRM)
Reference Document: Secure Controls Framework (SCF) version 2026.2
STRM Guidance: https://securecontrolsframework.com/start-here/set-theory-relationship-mapping-strm/

Focal Document: NIST CSWP 39 - Considerations for Achieving Crypto Agility
Focal Document URL: https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.39.pdf
Published STRM URL: https://content.securecontrolsframework.com/strm/scf-strm-general-nist-cswp-39.pdf

FDE #	FDE Name	Focal Document Element (FDE) Description	STRM Rationale	STRM Relationship	SCF Control	SCF #	Secure Controls Framework (SCF) Control Description	Strength of Relationship	Notes
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 lowlevel 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 nonremovable 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. Fieldprogrammable 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-	Functional	No Relationship	N/A	N/A	N/A	0	
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.	Functional	No Relationship	N/A	N/A	N/A	0	
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). 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	Functional	Intersects With	Cryptographic Agility Risk Assessment (CARA)	QTS-02	Mechanisms exist to perform a Cryptographic Agility Risk Assessment (CARA) that maps Technology Assets, Applications, Services and Data (TAASD) to: (1) Identify TAASD most vulnerable to quantum-enabled cryptanalytic threats; and (2) Prioritize TAASD based on potential business impact.	5	
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). 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	Functional	Intersects With	Post-Quantum Cryptography Agility Plan (PSCAP)	QTS-03	Mechanisms exist to develop a risk-prioritized Post-Quantum Cryptography Agility Plan (PQCAP) that: (1) Enables cryptographic agility; (2) Aligns with evolving security standards; and (3) Defines the approach for selecting and implementing PQC algorithms.	5	

NIST IR 8477-Based Set Theory Relationship Mapping (STRM)
Reference Document: Secure Controls Framework (SCF) version 2026.2
STRM Guidance: https://securecontrolsframework.com/start-here/set-theory-relationship-mapping-strm/

Focal Document: NIST CSWP 39 - Considerations for Achieving Crypto Agility
Focal Document URL: https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.39.pdf
Published STRM URL: https://content.securecontrolsframework.com/strm/scf-strm-general-nist-cswp-39.pdf

FDE #	FDE Name	Focal Document Element (FDE) Description	STRM Rationale	STRM Relationship	SCF Control	SCF #	Secure Controls Framework (SCF) Control Description	Strength of Relationship	Notes
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). 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	Functional	Intersects With	Post-Quantum Cryptography (PQC) Discovery & Visibility	QTS-04	Mechanisms exist to gain situational awareness into the organization's current cryptographic landscape through a formal discovery process that uses a combination of: (1) Automated tools; and (2) Manual techniques.	5	
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 threekey 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	Functional	Intersects With	Statutory, Regulatory & Contractual Compliance	CPL-01	Mechanisms exist to facilitate the identification and implementation of relevant statutory, regulatory and contractual controls.	5	
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, machineconsumable configuration profile that represents a crypto policy that can be deployed with automated tools.	Functional	Intersects With	Deprecated Cryptographic Algorithms	QTS-06.5	Mechanisms exist to identify and disallow quantum-vulnerable and otherwise deprecated cryptographic algorithms.	5	
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.	Functional	Intersects With	Post-Quantum Cryptography Agility Plan (PSCAP)	QTS-03	Mechanisms exist to develop a risk-prioritized Post-Quantum Cryptography Agility Plan (PQCAP) that: (1) Enables cryptographic agility; (2) Aligns with evolving security standards; and (3) Defines the approach for selecting and implementing PQC algorithms.	5	

NIST IR 8477-Based Set Theory Relationship Mapping (STRM)
Reference Document: Secure Controls Framework (SCF) version 2026.2
STRM Guidance: https://securecontrolsframework.com/start-here/set-theory-relationship-mapping-strm/

Focal Document: NIST CSWP 39 - Considerations for Achieving Crypto Agility
Focal Document URL: https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.39.pdf
Published STRM URL: https://content.securecontrolsframework.com/strm/scf-strm-general-nist-cswp-39.pdf

FDE #	FDE Name	Focal Document Element (FDE) Description	STRM Rationale	STRM Relationship	SCF Control	SCF #	Secure Controls Framework (SCF) Control Description	Strength of Relationship	Notes
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.	Functional	Intersects With	Crypto-Agility Architecture	QTS-06	Mechanisms exist to validate design-level cryptographic agility across protocols, libraries, kernels and hardware to ensure Technology Assets, Applications and/or Services (TAAS) can support larger Post-Quantum Cryptography (PQC) key, signature and ciphertext sizes.	5	
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.	Functional	Intersects With	Post-Quantum Cryptography Agility Plan (PSCAP)	QTS-03	Mechanisms exist to develop a risk-prioritized Post-Quantum Cryptography Agility Plan (PQCAP) that: (1) Enables cryptographic agility; (2) Aligns with evolving security standards; and (3) Defines the approach for selecting and implementing PQC algorithms.	5	
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.	Functional	Intersects With	Security, Compliance & Resilience Protection Portfolio Management	PRM-01	Mechanisms exist to facilitate the implementation of resource planning controls that provide a portfolio management approach to achieve security, compliance and resilience objectives.	5	
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.	Functional	Intersects With	Security, Compliance & Resilience Resource Management	PRM-02	Mechanisms exist to address all capital planning and investment requests, including the resources needed to implement the Security, Compliance & Resilience Program (SCRP) and document all exceptions to this requirement.	5	

NIST IR 8477-Based Set Theory Relationship Mapping (STRM)
Reference Document: Secure Controls Framework (SCF) version 2026.2
STRM Guidance: https://securecontrolsframework.com/start-here/set-theory-relationship-mapping-strm/

Focal Document: NIST CSWP 39 - Considerations for Achieving Crypto Agility
Focal Document URL: https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.39.pdf
Published STRM URL: https://content.securecontrolsframework.com/strm/scf-strm-general-nist-cswp-39.pdf

FDE #	FDE Name	Focal Document Element (FDE) Description	STRM Rationale	STRM Relationship	SCF Control	SCF #	Secure Controls Framework (SCF) Control Description	Strength of Relationship	Notes
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.	Functional	Intersects With	Prioritization To Address Evolving Risks & Threats	PRM-02.1	Mechanisms exist to integrate foundational cybersecurity practices with advanced technologies to maintain situation awareness of and minimize the organization's exposure to evolving risks and threats.	5	
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.	Functional	Intersects With	Crypto-Agility Architecture	QTS-06	Mechanisms exist to validate design-level cryptographic agility across protocols, libraries, kernels and hardware to ensure Technology Assets, Applications and/or Services (TAAS) can support larger Post-Quantum Cryptography (PQC) key, signature and ciphertext sizes.	5	
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.	Functional	Intersects With	Crypto-Agility Architecture	QTS-06	Mechanisms exist to validate design-level cryptographic agility across protocols, libraries, kernels and hardware to ensure Technology Assets, Applications and/or Services (TAAS) can support larger Post-Quantum Cryptography (PQC) key, signature and ciphertext sizes.	5	
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 thirdparty 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	Functional	Intersects With	Cloud Security Architecture	CLD-02	Mechanisms exist to ensure the cloud security architecture supports the organization's technology strategy to securely design, configure and maintain cloud employments.	5	

NIST IR 8477-Based Set Theory Relationship Mapping (STRM)
Reference Document: Secure Controls Framework (SCF) version 2026.2
STRM Guidance: https://securecontrolsframework.com/start-here/set-theory-relationship-mapping-strm/

Focal Document: NIST CSWP 39 - Considerations for Achieving Crypto Agility
Focal Document URL: https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.39.pdf
Published STRM URL: https://content.securecontrolsframework.com/strm/scf-strm-general-nist-cswp-39.pdf

FDE #	FDE Name	Focal Document Element (FDE) Description	STRM Rationale	STRM Relationship	SCF Control	SCF #	Secure Controls Framework (SCF) Control Description	Strength of Relationship	Notes
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 thirdparty 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	Functional	Intersects With	Crypto-Agility Architecture	QTS-06	Mechanisms exist to validate design-level cryptographic agility across protocols, libraries, kernels and hardware to ensure Technology Assets, Applications and/or Services (TAAS) can support larger Post-Quantum Cryptography (PQC) key, signature and ciphertext sizes.	5	
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 o Crypto agility practices are unstructured and unplanned. o Each group or team within the organization implements its own cryptography on a case-by-case basis. o The selection of cryptographic algorithms, schemes, libraries, and cryptographic products and services is not informed by current	Functional	Subset Of	Targeted Capability Maturity Levels	PRM-01.2	Mechanisms exist to define and identify targeted capability maturity levels.	10	
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 o Crypto agility practices are unstructured and unplanned. o Each group or team within the organization implements its own cryptography on a case-by-case basis. o The selection of cryptographic algorithms, schemes, libraries, and cryptographic products and services is not informed by current	Functional	Subset Of	Crypto Agility Maturity Assessment	QTS-02.3	Mechanisms exist to measure progress in adopting cryptographic agility using defined maturity criteria to support resilience against evolving Post-Quantum Cryptography (PQC) requirements and threats.	10	
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.	Functional	No Relationship	N/A	N/A	N/A	0	

NIST IR 8477-Based Set Theory Relationship Mapping (STRM)
Reference Document: Secure Controls Framework (SCF) version 2026.2
STRM Guidance: https://securecontrolsframework.com/start-here/set-theory-relationship-mapping-strm/

Focal Document: NIST CSWP 39 - Considerations for Achieving Crypto Agility
Focal Document URL: https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.39.pdf
Published STRM URL: https://content.securecontrolsframework.com/strm/scf-strm-general-nist-cswp-39.pdf

FDE #	FDE Name	Focal Document Element (FDE) Description	STRM Rationale	STRM Relationship	SCF Control	SCF #	Secure Controls Framework (SCF) Control Description	Strength of Relationship	Notes
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.	Functional	No Relationship	N/A	N/A	N/A	0	