Stream: Internet Engineering Task Force (IETF)
RFC: 9955
Category: Informational
Published: April 2026
ISSN: 2070-1721
Authors:
N. Bindel, *SandboxAQ*
B. Hale, *Naval Postgraduate School*
D. Connolly, *SandboxAQ*
F. Driscoll, *UK National Cyber Security Centre*

RFC 9955
Hybrid Signature Spectrums

Abstract

This document describes classification of design goals and security considerations for hybrid digital signature schemes, including proof composability, non-separability of the component signatures given a hybrid signature, backwards and forwards compatibility, hybrid generality, and Simultaneous Verification (SV).

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are candidates for any level of Internet Standard; see Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9955.

Copyright Notice

Table of Contents

1. Introduction

Plans to transition protocols to post-quantum cryptography sometimes focus on confidentiality, given the potential risk of store and decrypt attacks, where data encrypted today using traditional algorithms could be decrypted in the future by an attacker with a sufficiently powerful quantum computer, also known as a Cryptographically Relevant Quantum Computer (CRQC).

It is important to also consider transitions to post-quantum authentication; delaying such transitions creates risks. For example, attackers may be able to carry out quantum attacks against RSA-2048 [RSA] years before the public is aware of these capabilities. Furthermore, there are applications where algorithm turnover is complex or takes a long time. For example, root certificates are often valid for about 20 years or longer. There are also applications where future checks on past authenticity play a role, such as long-lived digital signatures on legal documents.

Still, there have been successful attacks against algorithms using post-quantum cryptography, as well as implementations of such algorithms. Sometimes an attack exploits implementation issues, such as [KYBERSLASH], which exploits timing variations, or [HQC_CVE], which exploits implementation bugs. Sometimes an attack works for all implementations of the specified algorithm. Research indicates that implementation-independent attacks published in 2023 or earlier had broken 48% of the proposals in Round 1 of the NIST Post-Quantum Cryptography Standardization Project, 25% of the proposals not broken by the end of Round 1, and 36% of the proposals selected by NIST for Round 2 [QRCSP].

Such cryptanalysis and security concerns are one reason to consider 'hybrid' cryptographic algorithms, which combine both traditional and post-quantum (or more generally a combination of two or more) algorithms. A core objective of hybrid algorithms is to protect against quantum computers while at the same time making clear that the change is not reducing security. A security premise for these algorithms is that if at least one of the two component algorithms of the hybrid scheme holds, the confidentiality or authenticity offered by that scheme is maintained. It should be noted that the word 'hybrid' has many uses, but this document uses 'hybrid' only in this algorithm sense.

Whether or not hybridization is desired depends on the use case and security threat model. Users may recognize a need to start post-quantum transition, even while issues such as those described above are a concern. For this, hybridization can support transition. It should be noted that hybridization is not necessary for all systems; recommendations on system types or analysis methods for such determination are out of scope of this document. For cases where hybridization is determined to be advantageous, a decision on how to hybridize needs to be made. With many options available, this document is intended to provide context on some of the trade-offs and nuances to consider.

Hybridization of digital signatures, where the hybrid signature may be expected to attest to both standard and post-quantum components, is subtle to design and implement due to the potential separability of the hybrid/dual signatures and the risk of downgrade/stripping attacks. There are also a range of requirements and properties that may be required from hybrid signatures, which will be discussed in this document. Some of these are mutually exclusive, which highlights the importance of considering use-case-specific requirements.

This document focuses on explaining a spectrum of different hybrid signature scheme design categories and different security goals for them. It is intended as a resource for designers and implementers of hybrid signature schemes to help them decide what properties they do and do not require from their use case. In scope limitations, it does not attempt to give concrete recommendations for any use case. It also intentionally does not propose concrete hybrid signature combiners or instantiations thereof. As with the data authenticity guarantees provided by any digital signature, the security guarantees discussed in this document are reliant on correct provisioning and management of the keys involved, e.g., entity authentication, key revocation, etc. This document only considers scenarios with a single signer and a single verifier; constructions with multiple signers or verifiers are out of scope.

1.1. Terminology

We follow existing documents on hybrid terminology [RFC9794] and hybrid key encapsulation mechanisms (KEMs) [RFC9954] to enable settling on a consistent language. We will make clear when this is not possible. In particular, we follow the definitions of 'post-quantum algorithm', 'traditional algorithm', and 'combiner'. Moreover, we use the definition of 'certificate' to mean 'public-key certificate' as defined in [RFC4949].

Signature scheme: A signature scheme is defined via the following three algorithms:

`KeyGen() -> (pk, sk)`:
: A probabilistic key generation algorithm, which generates a public verifying key `pk` and a secret signing key `sk`.

`Sign(sk, m) -> (sig)`:
: A probabilistic signature generation, which takes as input a secret signing key `sk` and a message `m`, and outputs a signature `sig`. In this document, the secret signing key `sk` is assumed to be implicit for notational simplicity, and the following notation is used: `Sign(m) -> (sig)`. If the message `m` is comprised of multiple fields, `m1, m2, ..., mN`, this is notated as `Sign(m) = Sign (m1, m2, ... mN) -> (sig)`.

`Verify(pk, sig, m) -> b`:
: A verification algorithm, which takes as input a public verifying key `pk`, a signature `sig`, and a message `m`, and outputs a bit `b` indicating `accept (b=1)` or `reject (b=0)` of the signature for the message `m`.

Hybrid signature scheme:
: Following [RFC9794], we define a hybrid signature scheme to be "a multi-algorithm digital signature scheme made up of two or more component digital signature algorithms". While it often makes sense for security purposes to require that the security of the component schemes is based on the hardness of different cryptographic assumptions, in other cases, hybrid schemes might be motivated, e.g., by interoperability of variants on the same scheme, and as such, both component schemes are based on the same hardness assumption (e.g., both post-quantum assumptions or even both the same concrete assumption, such as Ring Learning With Errors (LWE)). We allow this explicitly. In particular, this means that in contrast to [RFC9794], we will use the more general term 'hybrid signature scheme' instead of requiring one post-quantum and one traditional algorithm (i.e., Post-Quantum Traditional (PQ/T) hybrid signature schemes) to allow also the combination of several post-quantum algorithms. The term 'composite scheme' has sometimes been used as a synonym for 'hybrid scheme'. This is different from [RFC9794] where the term is used as a specific instantiation of hybrid schemes such that "where multiple cryptographic algorithms are combined to form a single key or signature such that they can be treated as a single atomic object at the protocol level". To avoid confusion, we will avoid the term 'composite scheme'.

Hybrid signature:
: A hybrid signature is the output of the hybrid signature scheme's signature generation. As a synonym, we might use 'dual signature'. For example, NIST defines a dual signature as "two or more signatures on a common message" [NIST_PQC_FAQ]. For the same reason as above, we will avoid using the term 'composite signature', although it sometimes appears as a synonym for 'hybrid/dual signature'.

Component (signature) scheme:
: Component signature schemes are the cryptographic algorithms contributing to the hybrid signature scheme. This has a similar purpose as in [RFC9794]. 'Ingredient (signature) scheme' may be used as a synonym.

Next-generation algorithms:
: Following [RFC9954], we define next-generation algorithms to be "algorithms that are not yet widely deployed but may eventually be widely deployed". Hybrid signatures are mostly motivated by preparation for post-quantum transition or use in long-term post-quantum deployment, hence the reference to post-quantum algorithms in this document. However, the majority of the discussion in this document applies equally well to future transitions to other next-generation algorithms.

Policy:
: Throughout this document, we refer to 'policy' in the context of, e.g., a system policy requiring verification of two signatures, an RFC description, guidance documentation, etc. Similar terminology may include 'security configuration settings' or related phrasing. We treat these terms as equivalent for the purposes of this document.

Artifact: An artifact is evidence of the sender's intent to hybridize a signature that remains even if a component signature is removed. Artifacts can be, e.g., at the algorithmic level (e.g., within the digital signature), at the protocol level (e.g., within the certificate), or on the system policy level (e.g., within the message). Artifacts should be easily identifiable by the receiver in the case of signature stripping.

Stripping attack: A stripping attack refers to a case where an adversary takes a message and hybrid signature pair and attempts to submit (a potential modification of) the pair to a component algorithm verifier, meaning that the security is based only on the remaining component scheme. A common example of a stripping attack includes a message and hybrid signature, comprised of concatenated post-quantum and traditional signatures, where an adversary with a quantum computer simply removes the post-quantum component signature and submits the (potentially changed) message and traditional component signature to a traditional verifier. This could include an authentic traditional certificate authority signature on a certificate that was originally hybrid-signed. An attribute of this is that an honest signer would attest to generating the signature. Stripping attacks should not be confused with component message forgery attacks.

Component message forgery attacks: A forgery attack refers to a case where an adversary attempts to forge a (non-hybrid) signature on a message using the public key associated with a component algorithm. A common example of such an attack would be a quantum attacker compromising the key associated with a traditional component algorithm and forging a message and signature pair. Message forgery attacks may be formalized with experiments such as the Existential Unforgeability under Chosen Message Attack (EUF-CMA) [EUFCMA], while the difference introduced in component message forgery attacks is that the key is accepted for both hybrid and single algorithm use. Further discussion of this appears in Section 5.

1.2. Motivation for Use of Hybrid Signature Schemes

Before diving into the design goals for hybrid digital signatures, it is worth taking a look at motivations for them. As many of the arguments hold in general for hybrid algorithms, we again refer to [RFC9954], which summarizes these well. In addition, we explicate the motivation for hybrid signatures here.

1.2.1. Complexity

Next-generation algorithms and their underlying hardness assumptions are often more complex than traditional algorithms. For example, the signature scheme Module-Lattice-Based Digital Signature Algorithm (ML-DSA) [MLDSA] (also known as CRYSTALS-DILITHIUM) follows the well-known Fiat-Shamir transform [FS] to construct the signature scheme but also relies on rejection sampling that is known to give cache side channel information (although this does not lead to a known attack). Likewise, the signature scheme FALCON [FALCON] uses complex sampling during signature generation. Furthermore, attacks against the next-generation multivariate schemes Rainbow [RAINBOW] and Great Multivariate Short Signature (GeMSS) [GEMSS] might raise concerns for conservative adopters of other algorithms, which could hinder adoption.

As such, some next-generation algorithms carry a higher risk of implementation mistakes and revision of parameters compared to traditional algorithms, such as RSA. RSA is a relatively simple algorithm to understand and explain, yet during its existence and use, there have been multiple attacks and refinements, such as adding requirements to how padding and keys are chosen, and implementation issues, such as cross-protocol attacks (e.g., component algorithm forgeries). Thus, even in a relatively simple algorithm, subtleties and caveats on implementation and use can arise over time. Given the complexity of next-generation algorithms, the chance of such discoveries and caveats needs to be taken into account.

Of note, some next-generation algorithms have received considerable analysis, for example, following attention gathered during the NIST Post-Quantum Cryptography Standardization Process [NIST_PQC_FAQ]. However, if and when further information on caveats and implementation issues come to light, it is quite possible that vulnerabilities will represent a weakening of security rather than a full "break". Such weakening may also be offset if a hybrid approach has been used.

1.2.2. Time

In large systems, including national systems, space systems, large healthcare support systems, and critical infrastructure, acquisition and procurement time can be measured in years, and algorithm replacement may be difficult or even practically impossible. Long-term commitment creates further urgency for immediate post-quantum algorithm selection, for example, when generating root certificates with their long validity windows. Additionally, for some sectors, future checks on past authenticity plays a role (e.g., many legal, financial, auditing, and governmental systems). This means there is a need to transition some systems to post-quantum signature algorithms imminently. However, as described above, there is a need to remain aware of potential, hidden subtleties in next-generation algorithms' resistance to standard attacks, particularly in cases where it is difficult to replace algorithms. This combination of time pressure and complexity drives some transition designs towards hybridization.

1.3. Goals

There are various security and usability goals that can be achieved through hybridization. The following provides a summary of these goals while also noting where goals are in conflict, i.e., that achievement of one goal precludes another, such as backwards compatibility.

1.3.1. Hybrid Authentication

One goal of hybrid signature schemes is security. As defined in [RFC9794], ideally a hybrid signature scheme can achieve 'hybrid authentication', which is the property that (cryptographic) authentication is achieved by the hybrid signature scheme, provided that a least one component signature algorithm remains 'secure'. There might be, however, other goals in competition with this one, such as backwards compatibility -- referring to the property where a hybrid signature may be verified by only verifying one component signature (see description below). Hybrid authentication is an umbrella term that encompasses more specific concepts of hybrid signature security, such as 'hybrid unforgeability' described next.

1.3.1.1. Hybrid Unforgeability

Hybrid unforgeability is a specific type of hybrid authentication, where the security assumption for the scheme (e.g., EUF-CMA) is maintained as long as at least one of the component schemes maintains that security assumption. We call this notion 'hybrid unforgeability'; it is a specific type of hybrid authentication. For example, the concatenation combiner in [HYBRIDSIG] is 'hybrid unforgeable'. As mentioned above, this might be incompatible with backwards compatibility, where the EUF-CMA security of the hybrid signature relies solely on the security of one of the component schemes instead of relying on both, e.g., the dual message combiner using nesting in [HYBRIDSIG]. For more details, refer to the discussion below.

Use cases where a hybrid scheme is used with, e.g., EUF-CMA security assumed for only one component scheme generally use hybrid techniques for their 'functional transition' pathway support. For example, hybrid signatures may be used as a transition step for gradual post-quantum adoption while ensuring interoperability when a system includes both verifiers that only support traditional signatures and verifiers that have been upgraded to support post-quantum signatures.

In contrast, use cases where a hybrid scheme is used with, e.g., EUF-CMA security assumed for both component schemes without prioritization between them can use hybrid techniques for both functional transition and security transition (i.e., a transition to ensure security even if it may not be known which algorithm should be relied upon).

1.3.2. Proof Composability

Under proof composability, the component algorithms are combined in such a way that it is possible to prove a security reduction from the security properties of a hybrid signature scheme to the properties of the respective component signature schemes and, potentially, other building blocks such as hash functions, key derivation functions, etc. Otherwise, an entirely new proof of security is required, and there is a lack of assurance that the combination builds on the standardization processes and analysis performed to date on component algorithms. The resulting hybrid signature would be, in effect, an entirely new algorithm of its own. The more the component signature schemes are entangled, the more likely it is that an entirely new proof is required, thus not meeting proof composability.

1.3.3. Weak Non-Separability

Non-separability was one of the earliest properties of hybrid digital signatures to be discussed [HYBRIDSIG]. It was defined as the guarantee that an adversary cannot simply "remove" one of the component signatures without evidence left behind. For example, there are artifacts that a carefully designed verifier may be able to identify or that are identifiable in later audits. This was later termed Weak Non-Separability (WNS) [HYBRIDSIGDESIGN]. Note that WNS does not restrict an adversary from potentially creating a valid component digital signature from a hybrid one (a signature stripping attack) but rather implies that such a digital signature will contain artifacts of the separation. Thus, authentication that is normally assured under correct verification of digital signature(s) is now potentially also reliant on further investigation on the receiver side that may extend well beyond traditional signature verification behavior. For instance, this can intuitively be seen in cases of a message containing a context note on hybrid

authentication that is then signed by all component algorithms/the hybrid signature scheme. If an adversary removes one component signature but not the other, then artifacts in the message itself point to the possible existence of a hybrid signature such as a label stating "this message must be hybrid-signed". This might be a countermeasure against stripping attacks if the verifier expects a hybrid signature scheme to have this property. However, it places the responsibility of signature validity not only on the correct format of the message, as in a traditional signature security guarantee, but the precise content thereof.

1.3.4. Strong Non-Separability

Strong Non-Separability (SNS) is a stronger notion of WNS, which is introduced in [HYBRIDSIGDESIGN]. SNS guarantees that an adversary cannot take a hybrid signature (and message) as input and output a valid component signature (and potentially different message) that will verify correctly. In other words, separation of the hybrid signature into component signatures implies that the component signature will fail verification (of the component signature scheme) entirely. Therefore, authentication is provided by the sender to the receiver through correct verification of the digital signature(s), as in traditional signature security experiments. It is not dependent on other components, such as message content checking, or protocol-level aspects, such as public key provenance. As an illustrative example distinguishing WNS from SNS, consider the case of component algorithms `Sigma_1.Sign` and `Sigma_2.Sign` where the hybrid signature is computed as a concatenation `(sig_1, sig_2)`, where `sig_1 = Sigma_1.Sign(hybridAlgID, m)` and `sig_2 = Sigma_2.Sign(hybridAlgID, m)`. In this case, a new message `m' = (hybridAlgID, m)` along with signature `sig_1` and `Sigma_1.pk`, with the hybrid artifact embedded in the message instead of the signature, could be correctly verified. The separation would be identifiable through further investigation, but the signature verification itself would not fail. Thus, this case shows WNS (assuming the verification algorithm is defined accordingly) but not SNS.

Some work [COMP-MLDSA] has looked at reliance on the public key certificate chains to explicitly define hybrid use of the public key, namely that `Sigma_1.pk` cannot be used without `Sigma_2.pk`. This implies pushing the hybrid artifacts into the protocol and system level and a dependency on the security of other verification algorithms (namely those in the certificate chain). This further requires that security analysis of a hybrid digital signature requires analysis of the key provenance, i.e., not simply that a valid public key is used but how its hybridization and hybrid artifacts have been managed throughout the entire chain. External dependencies such as this may imply hybrid artifacts lie outside the scope of the signature algorithm itself. SNS may potentially be achievable based on dependencies at the system level.

1.3.5. Backwards and Forwards Compatibility

Backwards compatibility refers to the property where a hybrid signature may be verified by only verifying one component signature, allowing the scheme to be used by legacy receivers. In general, this means verifying the traditional component signature scheme, potentially ignoring the post-quantum signature entirely. This provides an option to transition sender systems to post-quantum algorithms while still supporting select legacy receivers. Notably, this is a verification property; the sender has provided a hybrid digital signature, but the verifier is allowed, due to internal policy and/or implementation, to only verify one component signature.

Forwards compatibility has also been a consideration in hybrid proposals [NC-HYBRID-AUTH]. Forwards compatibility assumes that hybrid signature schemes will be used for some time but that eventually all systems will transition to use only one (particularly, only one post-quantum) algorithm. As this is very similar to backwards compatibility, it also may imply separability of a hybrid algorithm; however, it could also simply imply capability to support separate component signatures. Thus, the key distinction between backwards and forwards compatibility is that backwards compatibility may be needed for legacy systems that cannot use and/or process hybrid or post-quantum signatures, whereas in forwards compatibility, the system has those capabilities and can choose what to support (e.g., for efficiency reasons).

Ideally, backwards and forwards compatibility is achieved using redundant information as little as possible, as noted in [RFC9954].

1.3.6. Simultaneous Verification

Simultaneous Verification (SV) builds on SNS and was first introduced in [HYBRIDSIGDESIGN]. SV requires that not only is the entire hybrid signature (e.g., all component signature elements) needed to achieve a successful verification present in the signature presented for verification but also that verification of both component algorithms occurs roughly simultaneously. Namely, "missing" information needs to be computed by the verifier so that a normally functioning verification algorithm cannot "quit" the verification process before the hybrid signature elements attesting for both component algorithms are verified. This may additionally cover some error-injection and similar attacks, where an adversary attempts to make an otherwise honest verifier skip component algorithm verification. SV mimics traditional digital signatures guarantees, essentially ensuring that the hybrid digital signature behaves as a single algorithm vs. two separate component stages. Alternatively phrased, under an SV guarantee, it is not possible for an otherwise honest verifier to initiate termination of the hybrid verification upon successful verification of one component algorithm without also knowing if the other component succeeded. Note that SV does not prevent dishonest verification, such as if a verifier maliciously implements a customized verification algorithm that is designed with intention to subvert the hybrid verification process or skips signature verification altogether.

1.3.7. Hybrid Generality

Hybrid generality means that a general signature combiner is defined based on inherent and common structures of component digital signatures "categories". For instance, since multiple signature schemes use a Fiat-Shamir transform, a hybrid scheme based on the transform can be made that is generalizable to all such signatures. Such generality can also result in simplified constructions, whereas more tailored hybrid variants might be more efficient in terms of sizes and performance.

1.3.8. High Performance

Similar to performance goals noted for hybridization of other cryptographic components [RFC9954], hybrid signature constructions are expected to be as performant as possible. For most hybrid signatures, this means that the computation time should only minimally exceed the sum of the component signature computation time. It is noted that performance of any variety may come at the cost of other properties, such as hybrid generality.

1.3.9. High Space Efficiency

Similar to space considerations in [RFC9954], hybrid signature constructions are expected to be as space performant as possible. This includes messages (as they might increase if artifacts are used), public keys, and the hybrid signature. For the hybrid signature, size should no more than minimally exceed the signature size of the two component signatures. In some cases, it may be possible for a hybrid signature to be smaller than the concatenation of the two component signatures.

1.3.10. Minimal Duplicate Information

Duplicated information should be avoided when possible, as a general point of efficiency. This might include repeated information in hybrid certificates or in the communication of component certificates in addition to hybrid certificates (for example, to achieve backwards and forwards compatibility) or sending multiple public keys or signatures of the same component algorithm.

2. Non-Separability Spectrum

Non-separability is not a singular definition but rather is a scale representing `degrees` of separability hardness, visualized in Figure 1.

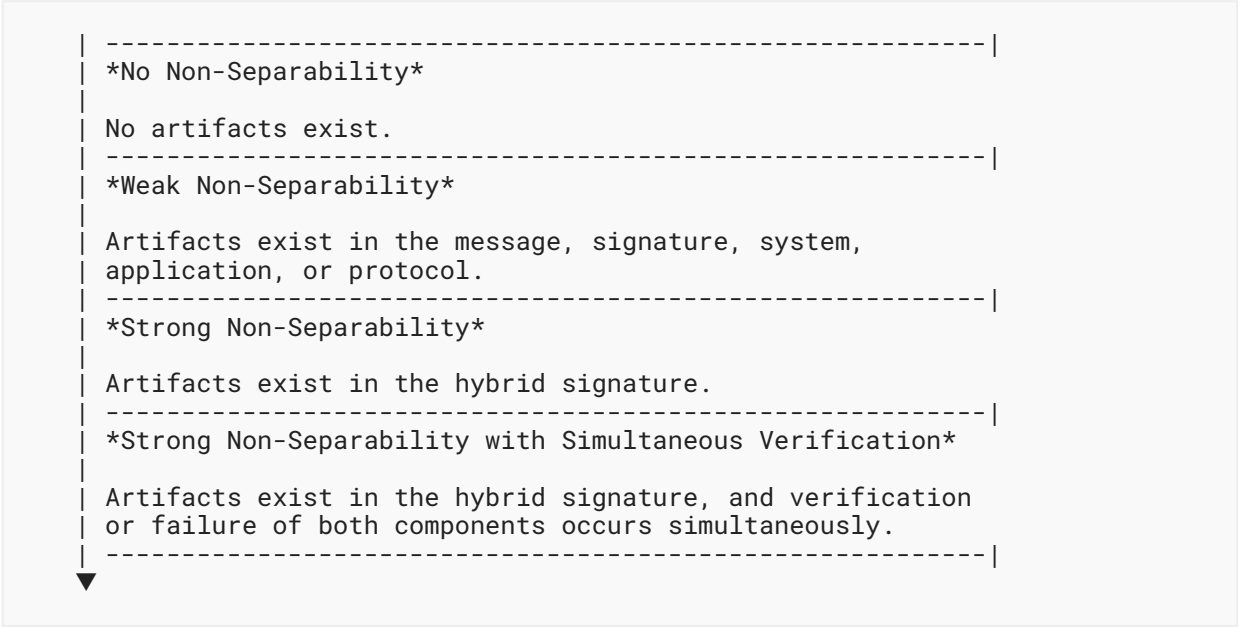

Figure 1: Spectrum of Non-Separability from Weakest to Strongest

At one end of the spectrum are schemes in which one of the component signatures can be stripped away with the verifier not being able to detect the change during verification. An example of this includes simple concatenation of signatures without any artifacts used. Nested

signatures (where a message is signed by one component algorithm and then the message-signature combination is signed by the second component algorithm) may also fall into this category, dependent on whether the inner or outer signature is stripped off without any artifacts remaining.

Next on the spectrum are weakly non-separable signatures. Under Weak Non-Separability, if one of the component signatures of a hybrid is removed, artifacts of the hybrid will remain (in the message, in the signature, at the protocol level, etc.). This may enable the verifier to detect if a component signature is stripped away from a hybrid signature, but that detectability depends highly on the type of artifact and permissions. For instance, if a message contains a label artifact "This message must be signed with a hybrid signature", then the system must be allowed to analyze the message contents for possible artifacts. Whether a hybrid signature offers (Weak/Strong) Non-Separability might also depend on the implementation and policy of the protocol or application the hybrid signature is used in on the verifier side. Such policies may be further ambiguous to the sender, meaning that the type of authenticity offered to the receiver is unclear. In another example, under nested signatures, the verifier could be tricked into interpreting a new message as the message/inner signature combination and verify only the outer signature. In this case, the inner signature is an artifact.

Third on the scale is the Strong Non-Separability notion, in which separability detection is dependent on artifacts in the signature itself. Unlike in Weak Non-Separability, where artifacts may be in the actual message, the certificate, or other non-signature components, this notion more closely ties to traditional algorithm security notions (such as EUF-CMA) where security is dependent on the internal construct of the signature algorithm and its verification. In this type, the verifier can detect artifacts on an algorithmic level during verification. For example, the signature itself may encode the information that a hybrid signature scheme is used. Examples of this type may be found in [HYBRIDSIGDESIGN].

For schemes achieving the most demanding security notion, Strong Non-Separability with Simultaneous Verification, verification succeeds only when both of the component signatures are present and the verifier has verified both signatures. Moreover, no information is leaked to the receiver during the verification process on the possible validity of the component signatures until both verify (or verification failure may or may not be attributable to a specific component algorithm). This construct most closely mirrors traditional digital signatures where, assuming that the verifier does verify a signature at all, the result is either a positive verification of the full signature or a failure if the signature is not valid. For fused hybrid signatures, a `full signature` implies the fusion of both component algorithms; therefore, this type of construction has the potential to achieve the strongest non-separability notion, which ensures an all-or-nothing approach to verification, regardless of adversarial action. Examples of algorithms providing this type of security can be found in [HYBRIDSIGDESIGN].

3. Artifacts

Hybridization benefits from the presence of artifacts as evidence of the sender's intent to decrease the risk of successful stripping attacks. This, however, depends strongly on where such evidence resides (e.g., in the message, in the signature, or somewhere on the protocol level

instead of at the algorithmic level). Even commonly discussed hybrid approaches, such as concatenation, are not inherently tied to one type of security (e.g., WNS or SNS). This can lead to ambiguities when comparing different approaches and assumptions about security or lack thereof. Thus, in this section, we cover artifact locations and also walk through a high-level comparison of a few hybrid categories to show how artifact location can differ within a given approach. Artifact location is tied to non-separability notions as described above; thus, the selection of a given security guarantee and general hybrid approach must also include a finer-grained selection of artifact placement.

3.1. Artifacts vs. Separability

Note that non-separability is a security notion of the signature scheme and not directly related to artifacts -- however, artifacts may be used for detection of separation. For instance, under strong non-separability, the scheme would fail verification if separation occurs, while for weak non-separability, some artifacts exist if separation occurs but verification would not necessarily fail. The verifier could indeed ignore the artifact, resulting in the scheme achieving only weak non-separability and not strong non-separability. It is rather that an artifact exists that could be identified if an investigation occurred, etc. Under weak non-separability, detection of separation may depend on non-cryptographic configurations or other dependencies. Also, strong non-separability and weak non-separability are properties of the signature scheme -- artifacts are not necessarily in the signature and may appear in the signed message, certificate, protocol, or policy (hence them not necessarily being related to the strong non-separability and weak non-separability security notions). Artifacts may still be useful (albeit dependent on system configurations) even if separable signatures are used.

3.2. Artifact Locations

There are a variety of artifact locations possible, ranging from within the message to the signature algorithm to the protocol level and even into policy, as shown in Table 1. For example, one artifact location could be in the message to be signed, e.g., containing a label artifact. Depending on the hybrid type, it might be possible to strip this away. For example, a quantum attacker could strip away the post-quantum signature of a concatenated dual signature, and, being able to forge the traditional signature, it could remove the label artifact from the message as well. So, for many applications and threat models, adding an artifact in the message might be insufficient under stripping attacks. Another artifact location could be in the public key certificates as described in [COMP-MLDSA]. In such a case, the artifacts are still present even if a stripping attack occurs. In yet another case, artifacts may be present through the fused hybrid method, thus making them part of the signature at the algorithmic level. Note that in this latter case, it is not possible for an adversary to strip one of the component signatures or use a component of the hybrid to create a forgery for a component algorithm. Such signatures provide SNS. Consequently, this also implies that the artifacts of hybridization are absolute in that verification failure would occur if an adversary tries to remove them.

Eventual security analysis may be a consideration in choosing between levels. For example, if the security of the hybrid scheme is dependent on system policy, then cryptographic analysis must necessarily be reliant on specific policies, and it may not be possible to describe a scheme's

security in a standalone sense. In this case, it is necessary to consider the configuration of a particular implementation or use to assess security, which could increase the risk of unknown and unanticipated vulnerabilities, regardless of the algorithms in use.

Location of Artifacts of Hybrid Intent	Level
Signature	Algorithm
Certificate	Protocol
Algorithm agreement / negotiation	Protocol
Message	Policy

Table 1: Artifact Placement Levels

3.3. Artifact Location Comparison Example

Here we provide a high-level example of how artifacts can appear in different locations even within a single, common approach. We look at the following categories of approaches: concatenation, nesting, and fusion. This is to illustrate that a given approach does not inherently imply a specific non-separability notion and that there are subtleties to the selection decision, since hybrid artifacts are related to non-separability guarantees. Additionally, this comparison highlights how artifact placement can be identical in two different hybrid approaches.

We briefly summarize the hybrid approach categories (concatenation, nesting, and fusion) for clarity in description before showing how each one may have artifacts in different locations in Table 2.

- Concatenation: Variants of hybridization where, for component algorithms `Sigma_1.Sign` and `Sigma_2.Sign`, the hybrid signature is calculated as a concatenation `(sig_1, sig_2)` such that `sig_1 = Sigma_1.Sign(hybridAlgID || m)` and `sig_2 = Sigma_2.Sign(hybridAlgID || m)`.
- Nesting: Variants of hybridization where, for component algorithms `Sigma_1.Sign` and `Sigma_2.Sign`, the hybrid signature is calculated in a layered approach as `(sig_1, sig_2)` such that, e.g., `sig_1 = Sigma_1.Sign(hybridAlgID || m)` and `sig_2 = Sigma_2.Sign(hybridAlgID || (m || sig_1))`.
- Fused hybrid: Variants of hybridization, where for component algorithms `Sigma_1.Sign` and `Sigma_2.Sign`, the hybrid signature is calculated to generate a single hybrid signature `sig_h` that cannot be cleanly separated to form one or more valid component constructs. For example, if both signature schemes are constructed through the Fiat-Shamir transform, the component signatures would include responses r_1 and r_2 and challenges c_1 and c_2, where c_1 and c_2 are hashes computed over the respective commitments comm_1 and comm_2 (and the message). A fused hybrid signature could consist of the component responses r_1 and r_2 and a challenge c that is computed as a hash over both commitments, i.e., c = Hash((comm_1 || comm_2) || Hash2(message)). As such, c does not belong to either of the component signatures but rather both, meaning that the signatures are 'entangled'.

#	Location of Artifacts of Hybrid Intent	Category
		Concatenated
1	None	No label in message, public keys are in separate certs
2	In message	Label in message, public keys are in separate certs
3	In cert	No label in message, public keys are in combined cert
4	In message and cert	Label in message, public keys are in combined cert
		Nested
5	In message	Label in message, public keys are in separate certs
6	In cert	No label in message, public keys are in combined cert
7	In message and cert	Label in message, public keys are in combined cert
		Fused
8	In signature	Public keys are in separate certs
9	In signature and message	Label in message, public keys are in separate certs
10	In signature and cert	Public keys are in combined cert
11	In signature and message and cert	Label in message, public keys are in combined cert

Table 2: Artifact Locations Depending on the Hybrid Signature Type

As can be seen, while concatenation may appear to refer to a single type of combiner, there are in fact several possible artifact locations depending on implementation choices. Artifacts help to support detection in the case of stripping attacks, which means that different artifact locations imply different overall system implementation considerations to be able to achieve such detection.

Case 1 provides the weakest guarantees of hybrid identification, as there are no prescribed artifacts and therefore non-separability is not achieved. However, as can be seen, this does not imply that every implementation using concatenation fails to achieve non-separability. Thus, it is advisable for implementers to be transparent about artifact locations.

In cases 2 and 5, the artifacts lie within the message. This is notable as the authenticity of the message relies on the validity of the signature, and the artifact location means that the signature in turn relies on the authentic content of the message (the artifact label). This creates a risk of circular dependency. Alternative approaches, such as cases 3, 4, 6, and 7, solve this circular dependency by provisioning keys in a combined certificate.

Another observation from this comparison is that artifact locations may be similar among some approaches. For instance, cases 3 and 6 both contain artifacts in the certificate. Naturally, these are high-level examples and further specification on concrete schemes in the categories are needed before prescribing non-separability guarantees to each, but this does indicate how there could be a strong similarity between such guarantees. Such comparisons allow for a systematic decision process, where security is compared and identified, and if schemes are similar in the desired security goal, then decisions between schemes can be based on performance and implementation ease.

A final observation that this type of comparison provides is how various combiners may change the security analysis assumptions in a system. For instance, cases 3, 4, 6, and 7 all push artifacts -- and therefore the signature validity -- into the certificate chain. Naturally, the entire chain must then also use a similar combiner if a straightforward security argument is to be made. Other cases, such as 8, 9, 10, and 11, put artifacts within the signature itself, meaning that these bear the closest resemblance to traditional schemes where message authenticity is dependent on signature validity.

4. Need for Approval Spectrum

In practice, use of hybrid digital signatures relies on standards where applicable. This is particularly relevant in the cases where use of FIPS-approved software modules is required but applies equally to any guidance or policy direction that specifies that at least one component algorithm of the hybrid has passed some certification type while not specifying requirements on the other component. NIST provides the following guidance in [NIST_PQC_FAQ] (emphasis added):

> Assume that in a [hybrid] signature, *one signature is generated with a NIST-approved signature scheme as specified in FIPS 186, while another signature(s) can be generated using different schemes*, e.g., ones that are not currently specified in NIST standards ... *`[hybrid] signatures` can be accommodated by current standards in `"FIPS mode"`, as defined in FIPS 140, provided at least one of the component methods is a properly implemented, NIST-approved signature algorithm.* For the purposes of FIPS 140 validation, any signature that is generated by a non-approved component scheme

> would not be considered a security function, since the NIST-approved component is regarded as assuring the validity of the `[hybrid]` signature.

This document does not define a formal interpretation of the NIST statement; however, we use it as motivation to highlight some points that implementers of hybrids may wish to consider when following any guidance documents that specify that 1) the signature scheme for one of the component algorithms must be approved and 2) the said algorithm must be a well-implemented or certified implementation. This type of `need for approval` (i.e., a requirement that an implementer is looking to follow regarding approval or certification of the software module implementation of a hybrid or its component algorithms) can drive some logistical decisions on what types of hybrids an implementer should consider.

In this respect, there is a `scale of approval` that developers may consider as to whether they are using at least one approved component algorithm implementation (`1-out-of-n approved software module`) or whether every component algorithm implementation is individually approved (`all approved software module`).

We provide a scale for the different nuances of `approval` of the hybrid combiners, where "approval" means that a software implementation of a component algorithm can be used unmodified for creation of the hybrid signature. This may be related to whether a hybrid combiner is likely to need dedicated certification.

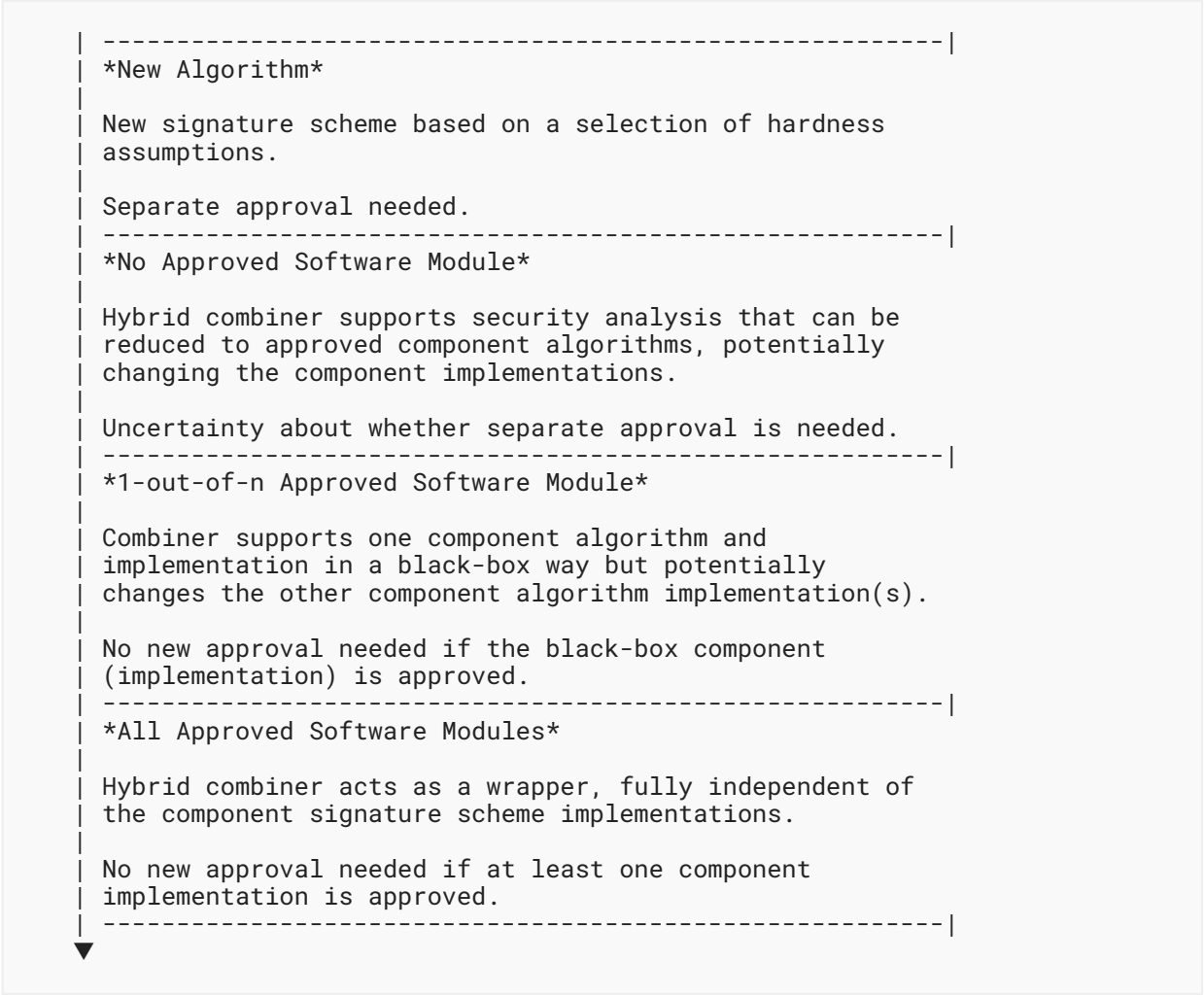

Figure 2: Generality / Need for Approval Spectrum

The first listed "combiner" would be a new construction with a security reduction to different hardness assumptions but not necessarily to approved (or even existing) signature schemes. Such a new, singular algorithm relies on both traditional and next-generation principles.

Next is a combiner that might take inspiration from existing/approved signature schemes such that its security can be reduced to the security of the approved algorithms. The combiner may, however, alter the implementations. As such, it is uncertain whether new approval would be needed as it might depend on the combiner and changes. Such a case may potentially imply a distinction between a need for fresh approval of the algorithm(s) and approval of the implementation(s).

The 1-out-of-n combiner uses at least one approved algorithm implementation in a black-box way (i.e., without modification to the software module implementation for that algorithm). It may potentially change the specifics of the other component algorithm implementations. If the premise is that no new approval is needed so long as at least one component is approved, then this is likely considered sufficient.

In an all-approved combiner, every algorithm implementation is used in a black-box way. A concatenation combiner is a simple example (where a signature is valid if all component signatures are valid). Thus, as all algorithm implementations are approved, a requirement that at least one of hybrid component algorithms is approved would be satisfied.

5. EUF-CMA Challenges

Unforgeability properties for hybrid signature schemes are more nuanced than for single-algorithm schemes.

Under the traditional EUF-CMA security assumption, an adversary requests signatures for messages of their choosing and succeeds if they are able to produce a valid signature for a message that was not part of an earlier request. EUF-CMA can be seen as applying to the hybrid signature scheme in the same way as single-algorithm schemes. Namely, the most straightforward extension of the traditional EUF-CMA security game would be that an adversary requests hybrid signatures for messages of their choosing and succeeds if they are able to produce a valid hybrid signature for a message that was not part of an earlier request. However, this has several layers of nuance under a hybrid construct.

Consider, for example, a simplistic hybrid approach using concatenated component algorithms. If the hybrid signature is stripped, such that a single component signature is submitted to a verification algorithm for that component along with the message that was signed by the hybrid, the result would be an EUF-CMA forgery for the component signature. This is because as the component signing algorithm was not previously called for the message, the hybrid signing algorithm was used to generate the signature. This is an example of a component algorithm forgery, a.k.a. a case of cross-algorithm attack or cross-protocol attack.

The component algorithm forgery verifier target does not need to be the intended recipient of the hybrid-signed message and may even be in an entirely different system. This vulnerability is particularly an issue among concatenated or nested hybrid signature schemes where individual component verification could be possible. It should be noted that policy enforcement of a hybrid verification does not mitigate the issue on the intended message recipient: The component forgery could occur on any system that accepts the component keys.

Thus, if EUF-CMA security for hybrids is informally defined in the straightforward way as that an adversary requests hybrid signatures for messages of their choosing and succeeds if they are able to produce a valid hybrid signature for a message that was not part of an earlier request, implicit requirements must hold in order to avoid real-world implications. Namely, either component algorithm forgeries, a.k.a. cross-protocol attacks, must be out of scope for the use case or the hybrid signature choice must be strongly non-separable. Otherwise, component

algorithm forgeries, which can be seen as a type of cross-protocol attack, affect the type of EUF-CMA properties offered and are a practical consideration that system designers and managers should be aware of when selecting among hybrid approaches for their use case.

There are a couple approaches to alleviating this issue, as noted above. One is on restricting key reuse. As described in [COMP-MLDSA], prohibiting hybrid algorithm and component algorithm signers and verifiers from using the same keys can help ensure that a component verifier cannot be tricked into verifying the hybrid signature. This would effectively put component forgeries out of scope for a use case. One means for restricting key reuse is through allowed key use descriptions in certificates. While prohibiting key reuse reduces the risk of such component forgeries, and is the mitigation described in [COMP-MLDSA], it is still a policy requirement and not a cryptographic assurance. Component forgery attacks may be possible if the policy is not followed or is followed inconsistently across all entities that might verify signatures using those keys. This needs to be accounted for in any security analysis. Since cryptographic provable security modeling has not historically accounted for key reuse in this way, it should not be assumed that systems with existing analyses are robust to this issue.

The other approach noted for alleviating the component forgery risk is through hybrid signature selection of a scheme that provides strong non-separability. Under this approach, the hybrid signature cannot be separated into component algorithm signatures that will verify correctly, thereby preventing the signature separation required for the component forgery attack to be successful.

It should be noted that weak non-separability is insufficient for mitigating risks of component forgeries. As noted in Section 9.3 of [COMP-MLDSA], in cases of hybrid algorithm selection that provide only weak non-separability, key reuse should be avoided, as mentioned above, to mitigate risks of introducing EUF-CMA vulnerabilities for component algorithms.

6. Discussion of Advantages and Disadvantages

The design (and hence security guarantees) of hybrid signature schemes depend heavily on the properties needed for the application or protocol using hybrid signatures. It seems that not all goals can be achieved simultaneously as exemplified below.

6.1. Backwards Compatibility vs. SNS

There is an inherent mutual exclusion between backwards compatibility and SNS. While WNS allows for a valid separation under leftover artifacts, SNS will ensure verification failure if a receiver attempts separation.

6.2. Backwards Compatibility vs. Hybrid Unforgeability

Similarly, there is an inherent mutual exclusion between backwards compatibility, when acted upon, and hybrid unforgeability, as briefly mentioned above. Since the goal of backwards compatibility is usually to allow legacy systems without any software change to be able to process hybrid signatures, all differences between the legacy signature format and the hybrid signature format must be allowed to be ignored, including skipping verification of signatures

additional to the classical signature. As such, if a system does skip a component signature, security does not rely on the security of all component signatures. Note that this mutual exclusion occurs at the verification stage, as a hybrid signature that is verified by a system that can process both component schemes can provide hybrid unforgeability even if another (legacy) system, processing the same hybrid signature, loses that property.

6.3. Simultaneous Verification vs. Low Need for Approval

Hybrid algorithms that achieve simultaneous verification tend to fuse (or 'entangle') the verification of component algorithms such that verification operations from the different component schemes depend on each other in some way. Consequently, there may be a natural connection between achieving simultaneous verification and a higher need for approval. As a contrasting example, NIST accommodates concatenation of a FIPS-approved signature and another (potentially non-FIPS approved) signature without any artifacts in FIPS 140 validation [NIST_PQC_FAQ]; however, as the component signatures are verified separately, it is not possible to enforce 'simultaneous verification'.

7. Security Considerations

This document discusses digital signature constructions that may be used in security protocols. It is an Informational document and does not directly affect any other documents. The security considerations for any specific implementation or incorporation of a hybrid scheme should be discussed in the relevant specification documents.

8. IANA Considerations

This document has no IANA actions.

9. References

9.1. Normative References

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2", FYI 36, RFC 4949, DOI 10.17487/RFC4949, August 2007, <https://www.rfc-editor.org/info/rfc4949>.

[RFC9794] Driscoll, F., Parsons, M., and B. Hale, "Terminology for Post-Quantum Traditional Hybrid Schemes", RFC 9794, DOI 10.17487/RFC9794, June 2025, <https://www.rfc-editor.org/info/rfc9794>.

[RFC9954] Stebila, D., Fluhrer, S., and S. Gueron, "Hybrid Key Exchange in TLS 1.3", RFC 9954, DOI 10.17487/RFC9954, April 2026, <https://www.rfc-editor.org/info/rfc9954>.

9.2. Informative References

[COMP-MLDSA]

Ounsworth, M., Gray, J., Pala, M., Klaußner, J., and S. Fluhrer, "Composite ML-DSA for use in X.509 Public Key Infrastructure", Work in Progress, Internet-Draft, draft-ietf-lamps-pq-composite-sigs-15, 24 February 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-lamps-pq-composite-sigs-15>.

[EUFCMA] Green, M., "EUF-CMA and SUF-CMA", <https://blog.cryptographyengineering.com/euf-cma-and-suf-cma/>.

[FALCON] Fouque, P., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T., Prest, T., Ricosset, T., Seiler, G., Whyte, W., and Z. Zhang, "FALCON: Fast-Fourier Lattice-based Compact Signatures over NTRU", Specification v1.2, 10 January 2020, <https://falcon-sign.info/falcon.pdf>.

[FS] Fiat, A. and A. Shamir, "How To Prove Yourself: Practical Solutions to Identification and Signature Problems", Advances in Cryptology -- CRYPTO' 86, Lecture Notes in Computer Science, vol. 263, pp. 186-194, DOI 10.1007/3-540-47721-7_12, 1986, <https://doi.org/10.1007%2F3-540-47721-7_12>.

[GEMSS] "GeMSS: A Great Multivariate Short Signature", 15 April 2020, <https://www-polsys.lip6.fr/Links/NIST/GeMSS_specification_round2_V2.pdf>.

[HQC_CVE] NIST, "CVE-2024-54137: liboqs has a correctness error in HQC decapsulation", 6 December 2024, <https://nvd.nist.gov/vuln/detail/CVE-2024-54137>.

[HYBRIDSIG] Bindel, N., Herath, U., McKague, M., and D. Stebila, "Transitioning to a Quantum-Resistant Public Key Infrastructure", Cryptology ePrint Archive, Paper 2017/460, May 2017, <https://eprint.iacr.org/2017/460>.

[HYBRIDSIGDESIGN] Bindel, N. and B. Hale, "A Note on Hybrid Signature Schemes", Cryptology ePrint Archive, Paper 2023/423, March 2023, <https://eprint.iacr.org/2023/423>.

[KYBERSLASH] Bernstein, D. J., Bhargavan, K., Bhasin, S., Chattopadhyay, A., Chia, T. K., Kannwischer, M. J., Kiefer, F., Ravi, P., and G. Tamvada, "KyberSlash: Exploiting secret-dependent division timings in Kyber implementations", Cryptology ePrint Archive, Paper 2024/1049, June 2024, <https://eprint.iacr.org/2024/1049>.

[MLDSA] NIST, "Module-Lattice-Based Digital Signature Standard", NIST FIPS 204, DOI 10.6028/NIST.FIPS.204, August 2024, <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.204.pdf>.

[NC-HYBRID-AUTH] Becker, A., Guthrie, R., and M. J. Jenkins, "Non-Composite Hybrid Authentication in PKIX and Applications to Internet Protocols", Work in Progress, Internet-Draft, draft-becker-guthrie-noncomposite-hybrid-auth-00, 22 March 2022, <https://datatracker.ietf.org/doc/html/draft-becker-guthrie-noncomposite-hybrid-auth-00>.

[NIST_PQC_FAQ] NIST, "Post-Quantum Cryptography FAQs", 5 July 2022, <https://csrc.nist.gov/Projects/post-quantum-cryptography/faqs>.

[QRCSP] Bernstein, D. J., "Quantifying risks in cryptographic selection processes", 24 November 2023, <https://cr.yp.to/papers/qrcsp-20231124.pdf>.

[RAINBOW] "PQC Rainbow", <https://www.pqcrainbow.org/>.

[RSA] Rivest, R. L., Shamir, A., and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", <https://people.csail.mit.edu/rivest/Rsapaper.pdf>.

Acknowledgements

This document is based on the template of [RFC9954].

We would like to acknowledge the following people in alphabetical order who have contributed to pushing this document forward, offered useful insights and perspectives, and/or stimulated work in the area: D.J. Bernstein, Scott Fluhrer, John Gray, Felix Günther, Serge Mister, Mike Ounsworth, Max Pala, Douglas Stebila, Falko Strenzke, and Brendan Zember

Authors' Addresses

Nina Bindel
SandboxAQ
Email: nina.bindel@sandboxaq.com

Britta Hale
Naval Postgraduate School
Email: britta.hale@nps.edu

Deirdre Connolly
SandboxAQ
Email: durumcrustulum@gmail.com

Florence Driscoll
UK National Cyber Security Centre
Email: flo.d@ncsc.gov.uk