A Note on Hybrid Signature Schemes

Nina Bindel[†] and Britta Hale[‡] *

[†]SandboxAQ, nina.bindel@sandboxaq.com
[‡]Naval Postgraduate School, britta.hale@nps.edu

Abstract This draft presents work-in-progress concerning hybrid/composite signature schemes. More concretely, we give several tailored combinations of Fiat-Shamir based signature schemes (such as Dilithium) or Falcon with RSA or DSA. We observe that there are a number of signature hybridization goals, few of which are not achieved through parallel signing or concatenation approaches. These include proof composability (that the post-quantum hybrid signature security can easily be linked to the component algorithms), weak separability, strong separability, backwards compatiblity, hybrid generality (i.e., hybrid compositions that can be instantiated with different algorithms once proven to be secure), and simultaneous verification. We do not consider backwards compatibility in this work, but aim in our constructions to show the feasibility of achieving all other properties. As a work-in-progress, the constructions are presented without the accompanying formal security analysis, to be included in an update.

1 Introduction

Post-quantum security is a focal point in current system design and modernization. Due to the risk imposed by quantum computers on common cryptographic algorithms, the National Institute of Standards and Technology (NIST) [8] has undertaken to standardize various post-quantum algorithms. The transition to post-quantum algorithms will be one of many challenges, ranging from algorithm functionality within protocol constraints to full system optimization on the trade-off between quantum resiliency and efficient operation.

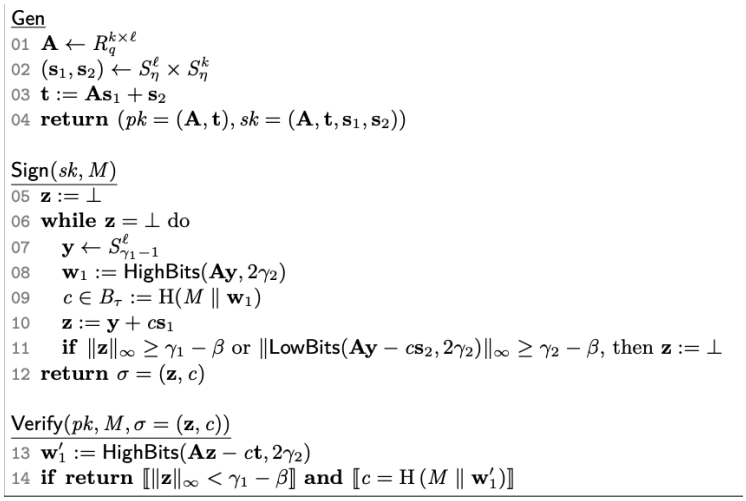
Within these transition discussions has risen the topic of *hybrid algorithms*, those that combine the respective security properties of standard and post-quantum algorithms. Hybridization has been looked at for key encapsulation [4], and in an initial sense for digital signatures [5]. Key encapsulation methods have looked at XOR-ing a post-quantum derived key with a standard key, with security ideally reducing to the randomness of either component. However, hybridization of digital signatures, where the verification tag must attest to both standard and post-quantum components is more subtle due to potential separability of the dual signatures and the risk of downgrading attacks.

* The views expressed in this document are those of the author and do not reflect the official policy or position of the DoD or the U.S. Government.

1.1 Motivation

Before diving into the design goals for hybrid digital signatures, it is worth taking a look on why hybrid digital signatures are desirable for some applications. This can be broken down into three motivation components.

Post-Quantum Complexity. For a cryptographer who has been working with post-quantum algorithms extensively, it may not immediately appear that these algorithms are particularly *complex*. Figure 1 shows the NIST finalist CRYSTALS-Dilithium algorithm – at least at a high level. It is supported by 38 pages of specification, including specification of multiple supporting algorithms. Standard algorithms such as RSA are often far more simple.



$\underline{\mathsf{Gen}}$
01 $\mathbf{A} \leftarrow R_q^{k \times \ell}$
02 $(\mathbf{s}_1, \mathbf{s}_2) \leftarrow S_\eta^\ell \times S_\eta^k$
03 $\mathbf{t} := \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$
04 **return** $(pk = (\mathbf{A}, \mathbf{t}), sk = (\mathbf{A}, \mathbf{t}, \mathbf{s}_1, \mathbf{s}_2))$

$\underline{\mathsf{Sign}(sk, M)}$
05 $\mathbf{z} := \perp$
06 **while** $\mathbf{z} = \perp$ do
07 $\quad \mathbf{y} \leftarrow S_{\gamma_1 - 1}^\ell$
08 $\quad \mathbf{w}_1 := \mathsf{HighBits}(\mathbf{A}\mathbf{y}, 2\gamma_2)$
09 $\quad c \in B_\tau := \mathrm{H}(M \parallel \mathbf{w}_1)$
10 $\quad \mathbf{z} := \mathbf{y} + c\mathbf{s}_1$
11 $\quad$ **if** $\|\mathbf{z}\|_\infty \geq \gamma_1 - \beta$ or $\|\mathsf{LowBits}(\mathbf{A}\mathbf{y} - c\mathbf{s}_2, 2\gamma_2)\|_\infty \geq \gamma_2 - \beta$, then $\mathbf{z} := \perp$
12 **return** $\sigma = (\mathbf{z}, c)$

$\underline{\mathsf{Verify}(pk, M, \sigma = (\mathbf{z}, c))}$
13 $\mathbf{w}_1' := \mathsf{HighBits}(\mathbf{A}\mathbf{z} - c\mathbf{t}, 2\gamma_2)$
14 **if return** $[\![\|\mathbf{z}\|_\infty < \gamma_1 - \beta]\!]$ **and** $[\![c = \mathrm{H}(M \parallel \mathbf{w}_1')]\!]$

Figure 1. Dilithium template, without key compression. Source: [2].

In addition are the mathematical assumption at play. RSA was built on the factoring hardness assumption that is well understood and reviewed. Many post-quantum algorithms are built on more complex assumptions that have not received such pervasive cryptanalysis approach investigation.

Even minor complexities may hide vulnerabilities. We can contrast this with RSA, a legacy standard algorithm that is simple enough to teach without field specific knowledge and whose core, "template" Sign and Verify algorithms can be written down in a single line each. Yet, as simple as RSA may seem, implementing it correctly has been non-trivial in wider practice over time (e.g., "Twenty Years of Attacks on the RSA Cryptosystem" [6] and "Forty years of attacks on the RSA cryptosystem: A brief survey" [15]). From adding padding to correct parameter

selection, it is clear that it can take time for understanding of finer grained security properties to mature.

If RSA, a simple algorithm based on a simple assumption, can still face a multitude of attacks and implementation aspects to fine-tune, it is logical to assume that the more complex post-quantum algorithms contain subtleties that have not yet been discovered. The case of SIKE [13,7], which made it to the second round of the NIST competition before being broken, is a warning that more cryptanalysis of many of the post-quantum algorithms may uncover unexpected subtleties.

Time. The above arguments could be taken as a license to delay deployment of cryptographic algorithms, giving time for yet further analysis and for the wheel of research to do its work. However, that would be an egregious error for many systems, were information security plays a vital role. Mosca's equation [14] very simply illustrates the risk of post-quantum transition delay:

$$l + d > q,$$

where l is the information life-span, d is the time for system transition to post-quantum algorithms, and q is the time before a quantum computer is ready to execute cryptanalysis.

In larger systems, including national systems, space systems, large healthcare support systems, and critical infrastructure, where acquisition and procurement time can be measured in years, this equation can have drastic implications. Figure 2 shows an expanded case for such a scenario, where d covers not only implementation, but system integration and testing, acquisition and procurement, and system updates (e.g., replacement for situations of hardware encoding). In short, many systems, including those that may handle data with long sensitivity lifespans, not only do not have time to delay a post-quantum transition, but are likely already behind. The need for quantum resistance now outranks a possible desire for more algorithm observation time.

This conundrum – the need to transition to post-quantum resistant algorithms now while simultaneously being aware of potential, hidden subtleties in their resistance to standard attacks – drives transition designs towards hybridization.

1.2 Hybrid Goals

Once we conclude that combining standard and post-quantum digital signature algorithms yields a potential advantage to security, the next step is how to design such hybrid algorithms. More particularly, what security properties should hybrid digital signatures provide?

Proof Composability. Proof composability is an inherent requirement in the very term *hybridization*. If the component algorithms are combined in such a way that it is not possible to prove reduction to the security properties of the respective algorithms, then an entire new proof of security is required. An implication of

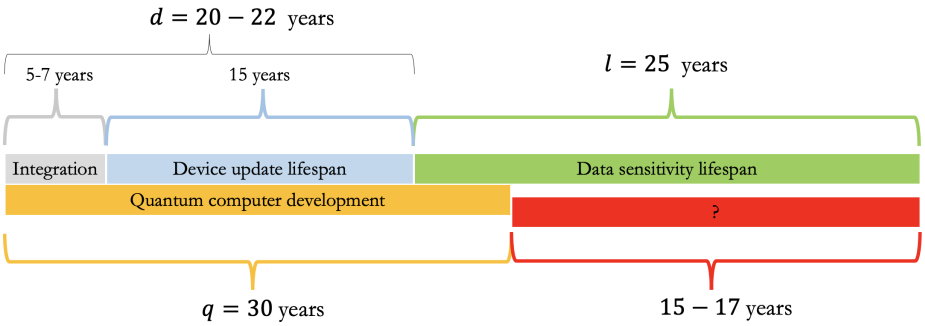


Figure 2. Expansion of Mosca's equation for a hypothetical system use case with vulnerability window shown in red. Source: [12].

this is that end users do not have the assurance that the combination builds on the standardization processes and analysis performed to date on component algorithms – the hybrid algorithm would be, in effect, an entirely new algorithm of its own. Thus, the security of the hybrid digital signatures should be clearly reducible to the standard and the post-quantum component, respectively.

Weak Non-Separability. Non-Separability was one of the earliest properties of hybrid digital signatures to be discussed [5]. It was defined as the guarantee that an adversary cannot simply "remove" either the standard or post-quantum digital signatures without evidence left behind; i.e., that there are artifacts that a carefully designed verifier may be able to identify, or that are identifiable in later audits. We term this *Weak Non-Separability* (WNS), and note that it is a digital signatures *verification* property in that it does not restrict an adversary from potentially creating a valid standard-only or post-quantum-only digital signatures from a hybrid one (a signature *stripping* attack), but rather implies that such a digital signatures will contain artifacts of the separation. Thus authentication is not simply provided by the sender to the receiver through correct verification of the digital signatures, but rather potentially through 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 standard and post-quantum authentication, that is then signed by e.g. a standard and then a post-quantum signature. If an adversary removes the outer signature but cannot forge the inner, then artifacts in the message itself point to the possible existence of a post-quantum signature such as a label stating "this message must be hybrid signed".[1] There is a limit, however, to what insight it can give on the missing signature.

[1] It should be noted in this analogy that an attacker removing the post-quantum is likely doing so due to that it can forge the standard signature, making it trivial to manipulate the message and remove the artifacts. Such considerations are another caveat to the weakness in WNS.

Strong Non-Separability. Non-Separability can be further strengthened from the weak version to fully restrict an adversary from separating the hybrid digital signatures. Under *Strong Non-Separability* (SNS), an adversary cannot take as input a hybrid digital signatures and output either a solely standard or a solely post-quantum digital signatures that will verify correctly. In other words, separation implies not only artifacts of the hybrid, but specifically that those artifacts are in the signature and that a separated signature will fail verification entirely. Thus, SNS restricts *creation* of valid partial digital signatures; authentication is provided by the sender to the receiver through correct verification of the digital signatures, as in traditional signature security experiments.

As an illustrative example distinguishing WNS from SNS, consider the case of component algorithms $\Sigma_1.\mathsf{Sign}$ and $\Sigma_2.\mathsf{Sign}$ where the hybrid signature is computed as a concatenation (σ_1, σ_2), where $\sigma_1 = \Sigma_1.\mathsf{Sign}(\text{hybridAlgID}, M)$ and $\sigma_2 = \Sigma_2.\mathsf{Sign}(\text{hybridAlgID}, M)$. In this case, separation and delivery of a new message $M^* = (\text{hybridAlgID}, M)$ along with signature σ_1 and $\Sigma_1.pk$ could allow for correct verification and the hybrid artifact is embedded in the message instead of the signature (identifiable through further investigation). Thus, this case shows WNS.

Some work [16] 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). External dependencies such as this may include artifacts and potentially even provide a degree of practical SNS based on dependencies at the system level. However, since those artifacts are outside the security definition scope for a digital signature, namely definitions such Existential Unforgeability under a Chosen Message Attack (EUF-CMA), we do not include them in the SNS category.

Backwards/Forwards Compatibility. Backwards compatibility refers to the property where a hybrid algorithm may also be used for a standard-only verification process i.e., a legacy receiver may take hybrid post-quantum digital signatures and verify them using only the standard verification algorithm. This necessarily implies that no post-quantum verification takes place. However, it does provide an option to transition various sender system attributes to post-quantum algorithms while still supporting select legacy receivers. Notably, this is a *verification* property; the sender has provided hybrid digital signatures, but the verifier, due to internal restrictions, only has the capacity to verify one signature.

Backwards compatibility may be further decomposed to subcategories where signature keys are either separate or hybrid (i.e., $\Sigma_1.pk$ and $\Sigma_2.pk$ or $\Sigma_h.pk = (\Sigma_1.pk, \Sigma_2.pk)$) such as to support implementations that cannot recognize hybrid schemes, or that a hybrid signature can be decomposed by a system that recognizes the hybrid signature but can only verify a component algorithm.

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

Forwards compatibility has also been a consideration in hybrid proposals [3]. Forward compatibility assumes that hybrid signatures will be used for some time, but that eventually all systems will transition to the post-quantum only components of such signatures. This is very similar to the backwards compatibility case and may imply separability of a hybrid algorithm; however it could also simply imply capability to support two separate signatures, one for hybrid and one for post-quantum. Thus the key distinction between backwards and forwards compatibility is that backwards compatibility may be needed for legacy systems that *cannot* use or potentially even recognize a hybrid algorithm (or potentially even a post-quantum algorithm at all), whereas in forwards compatibility the system has those capabilities and *can choose* what to support.

Simultaneous Verification. Simultaneous Verification is a further extension of SNS, requiring that not only both post-quantum and standard signature components be present to achieve a successful digital signatures verification, but that the verifier cannot "quit" the verification process before both components are verified.

Simultaneous Verification mimics traditional digital signatures guarantees. Authentication is a property that the sender provides to the receiver. Meanwhile the sender has no knowledge of whether or not the receiver correctly verified the digital signatures. What the sender is assured of is that one of two cases occurred: either 1) the receiver ignored the digital signatures or 2) the receiver initiated verification of the digital signatures (resulting in either successful or failed verification).

WNS complicates this situation, resulting in six cases instead of two: 1) the receiver ignored the digital signatures, 2) the receiver verified the full hybrid combination (with success or failure); 3) the receiver initiated verification of the hybrid digital signatures, but terminated once the standard component succeeded or failed; 4) the receiver initiated verification of the hybrid digital signatures, but terminated once the post-quantum component succeeded or failed; 5) the receiver initiated verification of the standard signature only (with success or failure), and 6) the receiver initiated verification of the post-quantum signature only (with success or failure). It may initially appear that (3) and (5) (resp. (4) and (6)) are similar, however (3) and (4) are precisely the cases eliminated by SNS, i.e. that the verifier does not take as in put the hybrid digital signatures, instead only attempting verification on one component.

SNS thus improves the situation to only four options. Still, the verifier can still terminate upon correctly checking only one component signature without actually verifying both parts. One could argue that a receiver who has checked the accuracy of their implementation should be assured that both components are verifying. This misconstrues the original intent though, which is to correctly mirror traditional digital signatures properties in hybrid digital signatures; ideally, the sender should be assured that there are only two options: 1) ignore the digital signatures or 2) verify the digital signatures (resulting in either failure or full verification).

Simultaneous Verification addresses this property. It ensures that both component algorithms verify simultaneously; alternatively phrased, it is not possible

to terminate hybrid verification with a success verification of one component algorithm without also knowing if the other component succeeded or failed.

Hybrid Generality. Hybrid digital signatures can be designed as a one-off instance – a combination of two digital signatures – in a process that must again be repeated for any other two signature combinations. Such processes are highly specific. Consequently, *hybrid generality* can be achieved through focusing design around inherent and common structures of component digital signatures. For instance, since multiple signature schemes use a Fiat-Shamir Transform, a hybrid scheme based on the transform can be made that is then generalizable to all such signatures. Such generality can also result in simplified constructions where as more tailored hybrid variants might be more efficient in terms of sizes and performance. It is key to find a balance of efficiency and generality, as more general constructions avoid design-specific attack vectors.

1.3 Contribution

In this work, we target the following properties: Proof Composablity, SNS, and Simultaneous Verification (a combination we term *true hybrid*). Simultaneous Verification is interesting in that it is a very strong property. While some developers may choose to forego it and achieve only SNS for a given use case, we note that user understanding of security properties is challenging enough without adding the subtle caveats present in e.g., WNS or SNS. Aligning hybrid properties to those traditionally expected from digital signatures ensures a more consistent understanding of the properties achieved by a digital signatures, which is particularly desirable in the unreliable world brought into effect by a potential quantum adversary. Furthermore, Simultaneous Verification has never before been achieved for hybrid digital signatures, which begs the question of whether that is possible. In this work, we show that Simultaneous Verification is achievable and provide it for a number of algorithm combinations. In addition, our hybrid combiners achieve shorter signatures than concatenation of the standard and post-quantum signatures in some cases, although the saving are little compared to the often large post-quantum signatures.

We aim for a generality of construction where possible, i.e., through use a Fiat-Shamir (FS) transform [10], such that any signature based on a FS transform may be "plugged" into the hybrid algorithm, with a plug-and-play design not only on hybrid operation but also with a provided proof. Thus, we present combinations such as RSA-FS and DSA-FS, that are applicable to the wide variety of FS-based digital signatures available. Moreover, we present combinations of RSA and DSA with Falcon (although DSA-Falcon does not satisfy all our goals).

1.4 Outline

Section 2 provides a high-level scale comparison of hybrid approaches, specifically focusing on a non-separability 'scale' of approaches and blackbox scale of how modular component algorithms can be combined. Section 3 describes preliminary

notation and basic definitions. Section 4 introduces true hybrid constructions, including FS-FS, FS-RSA, FS-DSA, Falcon-RSA, and FS-DSA.

2 Comparisons of Hybrids

In this section, we sketch out a scale comparison for hybrid signature schemes as there are various notions and meanings in the literature (as also differently discussed in [9]). These scales provide a comparative categorization of schemes – both the ones presented in this paper as well as basic approaches such as nested and concatenated signatures. We separate out two different categorisations (degree of hybridization and degree of blackbox modularity). These are related but do not correspond exactly to each other and are often two important points of discussion when security goals of hybrid signatures are discussed.

2.1 Non-separability

Non-separability is not a singular definition but rather is a scale, representing 'degrees' of separability hardness. Figure 4 shows the separability spectrum (in blue) with example constructions is given (in green).

On the very left of the spectrum are schemes in which one of the ingredient signatures can be stripped away with the verifier not being able to detect the change during verification. An example of this (depending on the exact implementation details) could be a simple concatenation of signatures. 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. Under a *weak hybrid failure*, such that the outer signature is stripped off, no artifact of the hybrid might remain (dependent on various implementation details).

Next on the spectrum are weakly non-separable signatures. Under weak non-separability, if one of the composite signatures of a hybrid is removed, artifacts of the hybrid will remain. This may enable the verifier to detect if an ingredient signature is stripped away from a hybrid signature, but that detectability depends highly on the type of artifact. For example, under a concatenated ECDSA-Dilithium hybrid where a certified ECDSA-Dilithium hybrid public key combination is required, the ECDSA component public key may be used to verify the ECDSA-only component signature. In this case, detection depends on the 'artifact' of the implementation requirement of a certified hybrid public key. This example illustrates that whether a hybrid signature is separable or (weakly) non-separable might also depend on the implementation on the protocol beyond the algorithmic level – i.e., on security aspects outside of standard definitions such as EUF-CMA. Nested signatures offer another example of this under a *strong hybrid failure*, where 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-tag is an artifact.

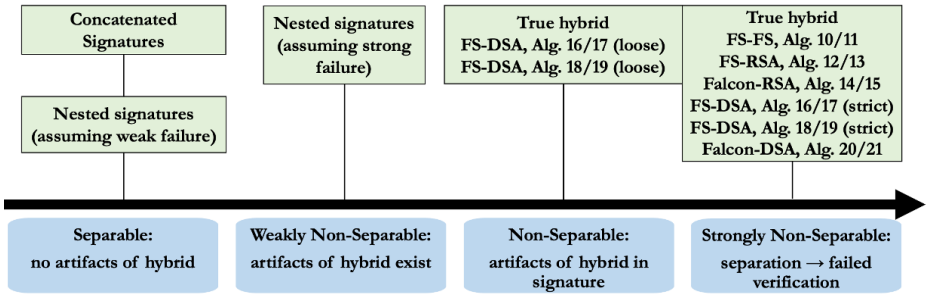


Figure 3. Spectrum (non-) separability of hybrid signature schemes

Third on the scale is a stronger 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 in 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 is enable to detect artifacts on an algorithmic level during verification. For example, the signature itself encodes the information that a hybrid signature scheme is used. Examples of this type may be found in our Alg. 16/17 and Alg. 18/19 examples in Section 4 if a 'loose' verification is applied if the verifier chooses to skip the $b = 1$ check.

For schemes of the strongest non-separability notion, verification fails not only when one of the ingredient signatures is missing but also if the verifier fails to verify both signatures. This non-separability 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 a the full signature or a failure if the signature is not valid. For hybrid signatures, a 'full signature' implies both component algorithms, and therefore the strongest non-separability notion enforces an all-or-nothing approach to verification. We focus this work on the viability of achieving such ultimate non-separability goals. Per NIST definition [1], this strongest non-separability notion also guarantees a hybrid verification, i.e. "verification of the [hybrid] signature requires all of the component signatures to be successfully verified." Thus, hybrid verification for other forms of non-separability notionally depend on that, when a signature if verified, no separation has taken place. More separable signatures that have been subject to a separation attack of any variety thus do not achieve hybrid verification.

2.2 Black Box Scale

On a practical notes, it is interesting to consider how entanglement of two component schemes for the goal of creating a hybrid digital signature affects approved use. This includes NIST guidance and FIPS approval considerations. NIST provides the following guidance (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. [1]

The emphasized text point to two things: 1) the signature scheme for one of the component hybrids must be approved and 2) that said algorithm must be properly implemented. This leaves some ambiguity as to whether only the algorithm must be approved and well implemented, or if that implementation must go through an approvals process as well. As such, there is a "black box scale" that developers may consider as to whether they are using at least one approved component algorithm ("black box algorithm"), or whether the implementation of that component algorithm has gone through an approvals review (thus making a "black box implementation"). The former 'black box algorithm' would suggest a straightforward path for FIPS-140 approvals based on the NIST guidelines; however, it is not inconceivable that using a 'black box implementation' could automate much of the certification review and therefore be attractive to developers.

We provide a scale for the different nuances of black box properties in Figure 4, together with respective example constructions. In addition the 'black box algorithm' options (represented as the 1 or 2 approved FIPS Algorithm box in blue) and the 'black box implementation' options (represented as the 1-approved and 2-approved fIPS implementation alternatives in blue), we also include a 0-approved hybrid approach for completeness of scale. This aligns to brand new signature algorithms that are reducible to hybrid hardness assumptions but require e.g., fresh NIST approval.

We align Alg. 10/11, 18/19, and 20/21 from this paper to the 'black box algorithm' interpretation. However, it is worth noting that the algorithmic modifications to either the standard or post-quantum component algorithms in these cases are minor (such as the inclusion if extra information in a hash). Since an addition, however minor, to an implementation implies that the implementation is not black box, we show these alignments conservatively. The remaining algorithms we provide in this paper align to either a full component black box implementation or event a both-component algorithm black box implementation approach.

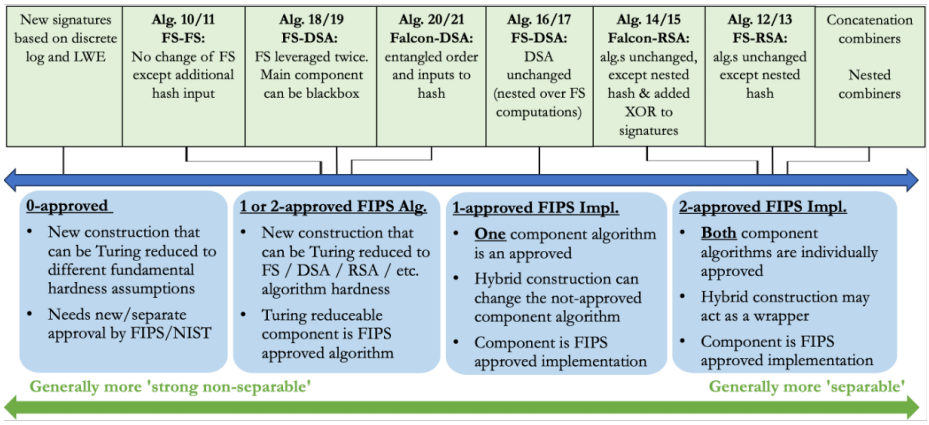


Figure 4. Spectrum of hybrids regarding whether ingredient signature schemes are use in a black-box way

3 Preliminaries

3.1 Digital signature schemes and unforgeability security definitions

We start be defining a signature scheme.

Definition 1 (Signature scheme). *A* digital signature scheme Σ *is a tuple* $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *of algorithms, with message space* $m \in \mathcal{M}$ *and signature space* $\sigma \in \mathcal{S}$*:*

- $\mathsf{KeyGen}() \leftarrow (sk, pk)$*: A probabilistic key generation algorithm that returns a secret signing key* sk *and public verification key* pk*.*
- $\mathsf{Sign}(sk, m) \leftarrow \sigma$*: A probabilistic signature generation algorithm that takes as input a signing key* sk *and a message* $m \in \mathcal{M}$*, and outputs a signature* $\sigma \in \mathcal{S}$*.*
- $\mathsf{Verify}(pk, m, \sigma) \leftarrow 0$ *or* 1*: A verification algorithm that takes as input a signature verification key* pk*, a message* $m \in \mathcal{M}$*, and a signature* $\sigma \in \mathcal{S}$*, and returns a bit* $b \in \{0, 1\}$ *indicating whether or not the signature verifies correctly under the public key. If* $b = 1$*, we say that the algorithm* accepts*, otherwise we say that the algorithm* rejects *the signature* σ *on the message* m*.*

We require that

$$\mathsf{Verify}(pk, m, \mathsf{Sign}(sk, m)) = 1$$

for any honestly generated pair $\mathsf{KeyGen}() \rightarrow (sk, pk)$*.*

If a proof for Σ *is being given in the random oracle model, we use* $\mathcal{H}_\Sigma$ *to denote the space of functions from which the random hash function is randomly sampled.*

$\underline{\text{Expt}^{\mathsf{EUF\text{-}CMA}}_{\Sigma}(\mathcal{A})}$:

0: $q_S \leftarrow 0$
1: $(sk, pk) \leftarrow\!\!\$\ \mathsf{KeyGen}()$
2: $((m^*_1, sig^*_1), \ldots, (m^*_{q_S+1}, sig^*_{q_S+1})) \ \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}_S(\cdot)}(pk)$
3: If $(\mathsf{Verify}(pk, m^*_i, sig^*_i) = 1\ \forall\ i \in [1, q_S + 1]) \wedge \left(m^*_i \neq m^*_j\ \forall\ i \neq j\right)$:
4: Return 1
5: Else return 0

$\underline{\text{Signing oracle } \mathcal{O}_S(m)}$:

8: $q_S \leftarrow q_S + 1$
9: $sig \leftarrow\!\!\$\ \mathsf{Sign}(sk, m)$
10: Return sig to $\mathcal{A}$

$\underline{\text{Verification oracle } \mathcal{O}_V(m, sig)}$:

11: $b \leftarrow \mathsf{Verify}(pk, m, sig)$
12: Return b to $\mathcal{A}$

Figure 5. EUF-CMA experiment in the random oracle models.

We say a signature scheme is secure if it is existentially unforgable against chosen-message attacks (EUF-CMA). More concretely, we say it is secure if the advantage

$$\text{Adv}^{\mathsf{EUF\text{-}CMA}}_{\Sigma}(\mathcal{A}) = \Pr\left[\text{Expt}^{\mathsf{EUF\text{-}CMA}}_{\Sigma}(\mathcal{A}) = 1\right]$$

against the unforgability game defined in Fig. 5 is negligible in the security parameter.

3.2 Relevant signature schemes from the literature

As we are presenting hybrid signature constructions tailored to different important schemes from the literature, we introduce these schemes with a common notation next. We do not present these schemes in detail e.g., inclusive of signing key generation, but rather the high-level sign and verify steps expected. We assume that these schemes are well known and details can be found in appropriate descriptions.

Signatures based on the Fiat-Shamir transform. The Fiat-Shamir transform can be used to construct a signature scheme from a Σ protocol between a prover and a verifier. Let H and D be hash functions, where D computes the digest of the message-to-be-signed. Moreover, we define P to be the function computing the prover's commitment (w) that the challenge (c) is computed over, and let f to be function computing the prover's response (z) in the Σ protocol. Furthermore, we define v to be the verifier's verification function and Rec the reconciliation functions that is used to compute the prover's commitment during

signature verification. For correctness of the signature scheme, it needs to hold that $P(sk, rand) = Rec(c, z, pk)$ for honestly generated $sk, pk, rand, c$ and z.

Algorithm 1 and Algorithm 2 show the basic sign and verify functions for a Fiat-Shamir based signature scheme.

Since Dilithium is a very prominent FS-based signature scheme, we do not give its description here. It is important to note that Dilithium is a 'FS with aborts signature scheme', which is irrelevant for our theoretic discussion of the hybrid signature scheme but of relevance for implementations.

Algorithm 1 Sign of FS$[H, D]$

Require: m, sk
Ensure: $sig = (c, z)$

1: $rand \leftarrow_{\$}$ Rand
2: $w \leftarrow P(sk, rand)$
3: $c \leftarrow H(w, D(m))$
4: $z \leftarrow f(c, rand, sk)$
5: **return** (c, z)

Algorithm 2 Verify of FS[H, D]

Require: m, (c, z), pk
Ensure: $\{1, 0\}_{\triangleright}$ accept, reject signature

1: $w \leftarrow Rec(c, z, pk)$
2: $b \leftarrow v(pk; c, z, D(m))$
3: **if** $b \wedge \big(c = H(w, D(m))\big)$ **then**
4: **return** 1
5: **end if**
6: **return** 0

RSA signature scheme. Our high-level description for the RSA signature generation and verification can be found in Algorithm 3 and Algorithm 4, respectively. We assume H to be a hash function, and *pad* to be appropriate padding for RSA.

Algorithm 3 Sign of RSA$[H]$

Require: m, sk
Ensure: $sig = s$

1: $c \leftarrow H(m)$
2: $s \leftarrow (c \| pad)^{sk} \bmod n$
3: **return** s

Algorithm 4 Verify of RSA[H]

Require: m, s, pk
Ensure: $\{1, 0\}_{\triangleright}$ accept, reject signature

1: $(c \| pad) \leftarrow (s)^{pk} \bmod n$
2: $c_{test} \leftarrow H(m)$
3: **if** $c = c_{test}$ **then**
4: **return** 1
5: **end if**
6: **return** 0

DSA signature schemes. We recall a high-level description of DSA's signature generation and verification in Algorithm 5 and Algorithm 6, respectively. Let $H : \{0,1\}^* \to \mathbb{Z}_q$ be a hash function with $\mathbb{Z}_q = \mathbb{Z}/q$ and $\mathbb{Z}_q^*$ the set of inverse elements in $\mathbb{Z}_q$. Moreover, let f be a function mapping from a group G to $\mathbb{Z}_q$. A DSA public key is defined as (G, q, g, y) for a group G with generator g, modulus q, and $y = g^x$ for secret key (G, q, g, x).

Algorithm 5 Sign of DSA$[H]$

Require: m, sk
Ensure: $sig = (r, s)$

1: $r \leftarrow 0, s \leftarrow 0$
2: **while** $r = 0$ or $s = 0$ **do**
3: $\quad k \leftarrow_\$ \mathbb{Z}_q^*$
4: $\quad r \leftarrow (f(g^k) \bmod p) \bmod q$
5: $\quad s \leftarrow k^{-1} \cdot (H(m) + xr) \bmod q$
6: **end while**
7: **return** (r, s)

Algorithm 6 Verify of DSA[H]

Require: m, $sig = (r, s)$, pk
Ensure: $\{1, 0\}$▷ accept, reject signature

1: **if** $r = \left(f(g^{H(m)\cdot s^{-1}} \cdot y^{r\cdot s^{-1}}) \bmod p\right) \bmod q$ **then**
2: $\quad$ **return** 1
3: **end if**
4: **return** 0

Falcon signature scheme. Falcon is based on the hash-and-sign paradigm for lattice-based cryptography that has been used int he (GPV) signature scheme by Gentry, Peikert, and Vaikuntanathan[11]. To achieve its higher performance and rather small size, Falcon uses a rather complicated trapdoor sampling algorithm. The details of this sampler are not of importance for the theoretical discussion of our constructions (although the details are important for implementations of such), and as such are omitted in our high-level explanation of the key generation, signature generation, and signature verification given in Algorithm 7, Algorithm 8, and Algorithm 9, respectively.

Let F be a hash function and f_1 be the trapdoor function that outputs elements of small Euclidean norm. In particular, the Euclidean norm should be smaller than an integer β as checked in the signature verification.

Algorithm 7 KeyGen of Falcon

Require: -
Ensure: sk, pk

1: $sk = g, -f, G, F$ polynomials with small coefficients
2: $pk = h = gf^{-1} \bmod q$
3: **return** sk, pk

Algorithm 8 Sign of Falcon

Require: sk, m
Ensure: (r, z_2)
1: $r \leftarrow_{\$} \text{Rand}$
2: $c \leftarrow F(r||m)$
3: $(z_1, z_2) \leftarrow f_1(c, sk)$ such that $z_1 + z_2 h = c \bmod q$
4: **return** (r, z_2)

Algorithm 9 Verify of Falcon

Require: $pk, m, (r, z_2)$
Ensure: accept, reject
1: $c \leftarrow F(r||m)$
2: $z_1 \leftarrow c - z_2 h \bmod q$
3: **if** $||(z_1, z_2)|| \leq \beta$ **then**
4: **return** accept
5: **end if**
6: **return** reject

4 True Hybrid / Parallel Schemes

Now we introduce a selection of true hybrid signature schemes, Σ_h. For the post-quantum sub-algorithm selection, we use 1) a generalized Fiat-Shamir based approach, for which Σ_h will hold for any Fiat-Shamir based post-quantum scheme, and 2) Falcon for contrast with a particular post-quantum scheme. For standard sub-algorithm selection, we provide for Fiat-Shamir based schemes, RSA and DSA. In particular, we provide a hybrid combination for DSA due to the ease in demonstrating the standard sub-components.

In what follows, let $\Sigma_1 = (\text{KeyGen}_1, \text{Sign}_1, \text{Verify}_1)$ and $\Sigma_2 = (\text{KeyGen}_2, \text{Sign}_2, \text{Verify}_2)$. Moreover, let $\Sigma_h = (\text{KeyGen}_\text{h}, \text{Sign}_\text{h}, \text{Verify}_\text{h})$, where KeyGen_h always returns $pk = (pk_1, pk_2)$ and $sk = (sk_1, sk_2)$ with $(pk_1, sk_1) \leftarrow \text{KeyGen}_1()$ and $(pk_2, sk_2) \leftarrow \text{KeyGen}_2()$.

4.1 FS-FS

A hybrid digital signature that combines two Fiat-Shamir based sub-algorithms can be realized elegantly, namely by selectively combining the commitment hash step in both schemes, see Algorithm 10 and Algorithm 11. This results in a signature size savings as only one commitment component must be transmitted in order to verify the signature, i.e, compared to the concatenation of two FS signatures, the signatures resulting from our FS-FS hybrid construction are smaller by the size of one hash output. Let $\Sigma_1 = \text{FS}_1[\text{H}, \text{D}]$ and $\Sigma_2 = \text{FS}_2[\text{H}, \text{D}]$ are both Fiat-Shamir signature schemes.

Correctness of our FS-FS construction follows directly from the correctness of the ingredient FS signature schemes.

Algorithm 10 Sign_h of $\Sigma_h = \text{FS-FS}[\Sigma_1, \Sigma_2, H, D]$

Require: m, sk
Ensure: $sig_h = (c, z_1, z_2)$

1: $rand_1 \leftarrow\!\!\$\ \text{Rand}$
2: $rand_2 \leftarrow\!\!\$\ \text{Rand}$
3: $w_1 \leftarrow P_1(sk_1, rand_1)$
4: $w_2 \leftarrow P_2(sk_2, rand_2)$
5: $c \leftarrow H((w_1, w_2), D(m))$
6: $z_1 \leftarrow f_1(c, rand_1, sk_1)$
7: $z_2 \leftarrow f_2(c, rand_2, sk_2)$
8: **return** (c, z_1, z_2)

Algorithm 11 Verify_h of $\Sigma_h = \text{FS-FS}[\Sigma_1, \Sigma_2, H, D]$

Require: m, $sig_h = (c, z_1, z_2)$, pk
Ensure: $\{1, 0\}$▷ accept, reject signature

1: $w_1 \leftarrow Rec_1(c, z_1, pk_1)$
2: $w_2 \leftarrow Rec_2(c, z_2, pk_2)$
3: $b_1 \leftarrow \text{v}_1(pk_1; c, z_1, D(m))$
4: $b_2 \leftarrow \text{v}_2(pk_2; c, z_2, D(m))$
5: **if** $b_1 \wedge b_2 \wedge \big(c = H(w_1, w_2, D(m))\big)$ **then**
6: **return** 1
7: **end if**
8: **return** 0

Theorem 1 (EUF-CMA security of FS-FS in the ROM). *Let either* $\Sigma_1 = \text{FS}[\text{H}, \text{D}]$ *or* $\Sigma_2 = \text{FS}[H, D]$ *be EUF-CMA secure in the random oracle model,* D *be a collision resistant hash function, and* H *be a second pre-image resistant hash function. Then* $\Sigma_h = \text{FS-FS}[\Sigma_1, \Sigma_2, H, D]$ *is EUF-CMA secure in the random oracle model.*

4.2 FS-RSA

Let $\Sigma_1 = \text{FS}[F, D]$ be a Fiat-Shamir signature scheme and $\Sigma_2 = \text{RSA}[F]$ be RSA. Here it is required that the hash algorithm used in RSA be the same as one of those used in the FS scheme. Compared to the concatenation of a FS signature and an RSA signature, the signatures resulting from our FS-RSA hybrid construction are smaller by the size of one hash output.

Algorithm 12 Sign_h of $\Sigma_h = \text{FS-RSA}[\Sigma_1, \Sigma_2, F, D]$

Require: m, sk
Ensure: $sig_h = (z, s)$

1: $rand \leftarrow_\$ \text{Rand}$
2: $w \leftarrow P(sk_1, rand)$
3: $c \leftarrow F(w, D(m))$
4: $z \leftarrow f(c, rand_1, sk_1)$
5: $s = (c||pad)^{sk_2} \bmod n$
6: **return** (z, s)

Algorithm 13 Verify_h of $\Sigma_h = \text{FS-RSA}[\Sigma_1, \Sigma_2, F, D]$

Require: m, $sig_h = (z, s)$, pk
Ensure: $\{1, 0\}$▷ accept, reject signature

1: $(c||pad) \leftarrow (s)^{pk_2} \bmod n$
2: $w \leftarrow Rec(c, z, pk_1)$
3: Check $\text{v}_1(pk_1; c, z, D(m))$
4: **if** $c = F(w, D(m))$ **then**
5: **return** 1
6: **end if**
7: **return** 0

4.3 Falcon-RSA

Very similarly, we can construct Falcon-RSA: Σ_1 is instantiated with Falcon (i.e., $\Sigma_1 = \text{Falcon}[F]$) and $\Sigma_2 = \text{RSA}[F]$ is RSA. Note that this requires that the same hash algorithm, F, be used in both schemes. For clarity, we sample random values as $r \leftarrow_\$ \text{Rand}$ in Falcon, vs. $rand \leftarrow_\$ \text{Rand}$ in FS.

Compared to the concatenation of an RSA signature and a Falcon signature, the signatures resulting from our Falcon-RSA hybrid construction are larger since we need to communicate two ring elements z_2 and z_3 (using an encoding of the randomness r) instead of one ring element and one random seed as in the original Falcon signature scheme, in addition to the RSA signature s.

Correctness follows directly by construction of z_3 and by the definition of the trapdoor sampling function f_1 that returns elements with small Euclidean norm.

Theorem 2 (EUF-CMA security of Falcon-RSA in the ROM). *Let either $\Sigma_1 = \text{Falcon}[F]$ or $\Sigma_2 = \text{RSA}[F]$ be EUF-CMA secure in the random oracle model, F be a collision resistant hash function. Then $\Sigma_h = \text{Falcon-RSA}[\Sigma_1, \Sigma_2, F]$ is EUF-CMA secure in the random oracle model.*

Algorithm 14 Sign_h of $\Sigma_h = \text{Falcon-RSA}[\Sigma_1, \Sigma_2, F]$

Require: m, $sk_1, sk_2, pk_1 = h$
Ensure: $sig_h = (z_2, z_3, s)$

1: $r \leftarrow_\$ \text{Rand}$
2: $c \leftarrow F(r||m)$
3: $(z_1, z_2) \leftarrow f_1(c, sk_1)$ such that $z_1 + z_2 h = c \bmod q$
4: $s = (c||pad)^{sk_2} \bmod n$
5: $z_3 \leftarrow z_1 \oplus r$
6: **return** (z_2, z_3, s)

Algorithm 15 Verify_h of $\Sigma_h = \text{Falcon-RSA}[\Sigma_1, \Sigma_2, F]$

Require: m, $sig_h = (z_2, z_3, s)$, pk_2, pk_1
Ensure: $\{1, 0\}$▷ accept, reject signature

1: $(c||pad) \leftarrow (s)^{pk_2} \bmod n$
2: $z_1 \leftarrow c - z_2 pk_1 \bmod q$
3: $r \leftarrow z_1 \oplus z_3$
4: **if** $||(z_1, z_2)|| \leq \beta \wedge c = F(r||m)$ **then**
5: **return** 1
6: **end if**
7: **return** 0

4.4 FS-DSA

Next we turn our attention to the FS-DSA combination, where $\Sigma_1 = \text{FS}[F, D]$, i.e., instantiated with a Fiat-Shamir scheme, and $\Sigma_2 = \text{DSA}[F]$ is DSA. Similarly to the Falcon-RSA combination in Section 4.3, the hash function F is used in both schemes.

Here we present two FS-DSA variants, illustrating that there may sometimes be a variety of approaches to hybridization that still achieve the intended hybridization goals. While it may not be readily apparent, these two constructions actually work in similar ways, i.e. by XOR-ing hash outputs so as to mask the digest of the message. What differs is whether the masking takes place internally to the DSA computation or internally to the FS computation. We find that such a masking approach helps to ensure simultaneous verification (one of our original hybridization goals). Note also, that this issue arises for DSA variants and was not present in FS-FS; this is due to the fact that the DSA signature verification algorithm conclusion step is not a comparison of digests, but rather a comparison of exponentiation values (r). Hybridization of digital signatures that conclude with a digest comparison allow for more straight forward means of mixing the two components to ensure simultaneous verification.

In contrast to our hybrid FS-FS and FS-RSA variants where the construction yields a signature smaller than the concatenation of two component signatures, or Falcon-RSA which results in a larger signature, our FS-DSA variants match directly to the total signature size of two component signatures, i.e., the signature size and format is the same as if two concatenated signatures were used. On one hand, this means that signature space is not conserved in the hybridization process. However, on the other hand, we note that there may be security benefits to sending a total hybrid signature that matches the format of component algorithm outputs: it is difficult for an adversary to know whether the signature sent is a true hybrid or simply a pair of concatenated signatures.

The correctness of our first FS-DSA construction in Algorithm 16 /Algorithm 17 follows from the correctness of the FS scheme (which gives $b = 1$ in line 2 and that $P(sk_1, rand_1) = Rec(c, z, pk_1)$) and the correctness of DSA to ensure $r = f(g^{F(w,D(m))s^{-1}}(pk_2)^{rs^{-1}}))$ mod p mod q. Similarly, the correctness follows for our second FS-DSA construction in Algorithm 18/Algorithm 19.

Algorithm 16 Sign_h of Σ_h = FS-DSA$[\Sigma_1, \Sigma_2, F, D]$

Require: m, sk
Ensure: $sig_h = (z, c, r, s)$

1: $r \leftarrow 0, s \leftarrow 0$
2: $rand_1 \leftarrow\!\!\$$ Rand
3: $w \leftarrow P(sk_1, rand_1)$
4: **while** $r = 0$ or $s = 0$ **do**
5: $k \leftarrow\!\!\$ \mathbb{Z}_q^*$
6: $r \leftarrow (f_2(g^k) \bmod p) \bmod q$
7: $s \leftarrow k^{-1}(F(w, D(m)) + (sk_2)r) \bmod q$
8: $c \leftarrow F(w, D(r, s) \oplus D(m))$
9: $z \leftarrow f_1(c, rand_1, sk_1)$
10: **end while**
11: **return** (z, c, r, s)

Algorithm 17 Verify_h of Σ_h = FS-DSA$[\Sigma_1, \Sigma_2, F, D]$

Require: m, $sig_h = (z, c, r, s)$, pk
Ensure: $\{1, 0\}\triangleright$ accept, reject signature

1: $w \leftarrow Rec(c, z, pk_1)$
2: $b \leftarrow \text{v}_1(pk_1; c, z, D(r, s) \oplus D(m))$
3: **if** $(b = 1) \wedge (r = f(g^{F(w,D(m))s^{-1}}(pk_2)^{rs^{-1}})) \bmod p \bmod q$ **then**
4: **return** 1
5: **end if**
6: **return** 0

Theorem 3 (EUF-CMA security of FS-DSA in the ROM). *Let either $\Sigma_1 = \text{FS}[F, D]$ or $\Sigma_2 = \text{DSA}[D]$ be EUF-CMA secure in the random oracle model, D be a collision resistant hash function, and F be a second-preimage resistant hash function. Then $\Sigma_h = \text{FS-DSA}[\Sigma_1, \Sigma_2, D, F]$ as given in Algorithm 16 and Algorithm 17 (resp., as given in Algorithm 18 and Algorithm 19) is EUF-CMA secure in the random oracle model.*

Note that it should be possible might to transform our FS-DSA constructions Algorithm 16/Algorithm 17 and Algorithm 18/Algorithm 17 to FS-ECDSA constructions.

Algorithm 18 Sign_h of $\Sigma_h = \text{FS-DSA}[\Sigma_1, \Sigma_2, F, D]$

Require: m, sk_1, sk_2
Ensure: $sig_h = (c, z, r, s)$,

$r \leftarrow 0, s \leftarrow 0$
$rand \leftarrow_\$ \text{Rand}$
$w \leftarrow P(sk_1, rand)$
$c \leftarrow F(w, D(m))$
while $r = 0$ or $s = 0$ **do**
    $k \leftarrow_\$ \mathbb{Z}_q^*$
    $r \leftarrow (f_2(g^k) \bmod p) \bmod q$
    $u \leftarrow F((w, r), D(m))$
    $z \leftarrow f_1(u, rand, sk_1)$
    $s \leftarrow k^{-1}(u \oplus c + (sk_2)r) \bmod q$
end while
return (u, z, r, s)

Algorithm 19 Verify_h of $\Sigma_h = \text{FS-DSA}[\Sigma_1, \Sigma_2, F, D]$

Require: m, $sig_h = (u, z, r, s)$, pk_2, h
Ensure: $\{1, 0\}\triangleright$ accept, reject signature

$w \leftarrow Rec(u, z, pk_1)$
$c \leftarrow F(w, D(m))$
$b \leftarrow \text{v}(pk_1; c, z, D(m))$
if $(b = 1) \wedge (r = f(g^{(F((w,r),D(m))\oplus c)\cdot s^{-1}} \cdot pk_2^{r\cdot s^{-1}})) \bmod p \bmod q$ **then**
    return 1
end if
return 0

4.5 Falcon-DSA

Finally, we construct Falcon-DSA, where Σ_1 is instantiated with the Falcon signature scheme and Σ_2 with DSA. As in Falcon-RSA, the schemes share the same hash algorithm, F.

The hybrid scheme signature size for Falcon-DSA are larger than in a concatenated approach.

It is important to note that our Falcon-DSA construction does not satisfy the goal of simultaneous verification. Such a construction is left for future work.

Theorem 4 (EUF-CMA security of Falcon-DSA in the ROM). *Let either $\Sigma_1 = \text{Falcon}[F]$ or $\Sigma_2 = \text{DSA}[F]$ be EUF-CMA secure in the random oracle model, F be a collision resistant hash function. Then $\Sigma_h = \text{Falcon-DSA}[\Sigma_1, \Sigma_2, F]$ is EUF-CMA secure in the random oracle model.*

Algorithm 20 Sign_h of Σ_h = Falcon-DSA$[\Sigma_1, \Sigma_2, F]$

Require: m, $sk_1, sk_2, pk_1 = h$
Ensure: $sig_h = (r_1, z_2, r_2, s)$

$r_2 \leftarrow 0, s \leftarrow 0$
$k \leftarrow_\$ \mathbb{Z}_q^*$
$r_1 \leftarrow_\$ \text{Rand}$
while $r_2 = 0$ or $s = 0$ **do**
    $r_2 \leftarrow (f_2(g^k) \bmod p) \bmod q$
    $c \leftarrow F((r_2, r_1) \| m)$
    $(z_1, z_2) \leftarrow f_1(c, sk_1)$ such that $z_1 + z_2 h = c \bmod q$
    $s \leftarrow k^{-1}(c + (sk_2) r_2) \bmod q$
end while
return (r_1, z_2, r_2, s)

Algorithm 21 Verify_h of Σ_h = Falcon-DSA$[\Sigma_1, \Sigma_2, F]$

Require: m, (r_1, z_2, r_2, s), $pk_1 = h, pk_2$
Ensure: $\{1, 0\}\triangleright$ accept, reject signature

$c \leftarrow F((r_2, r_1) \| m)$
$z_1 \leftarrow c - z_2 h \bmod q$
$b \leftarrow ||(z_1, z_2)||$
if $c = F\Big(\Big(f_2(g^{F((r_2,r_1),m)\cdot s^{-1}} (pk_2)^{r_2 \cdot s^{-1}}) \bmod p\Big) \bmod q, r_1), m\Big) \wedge (b < \beta)$ **then**
    return 1
end if
return 0

Acknowledgments

We thank Felix Günther, John Gray, Mike Ounsworth and Scott Fluhrer, for fruitful discussion in particular around the spectrum of hybrid constructions.

References

1. Post-quantum cryptography: Faqs. Tech. rep., National Institute of Standards (NIST), `https://csrc.nist.gov/Projects/post-quantum-cryptography/faqs`, created January 03, 2017, Updated July 17, 2023.
2. Bai, S., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., Stehlé, D.: Crystals-dilithium: Algorithm specifications and supporting documentation (version 3.1). Tech. rep. (2021), `https://pq-crystals.org/dilithium/data/dilithium-specification-round3-20210208.pdf`
3. Becker, A., Guthrie, R., Jenkins, M.: Non-Composite Hybrid Authentication in PKIX and Applications to Internet Protocols. Internet-Draft draft-becker-guthrie-noncomposite-hybrid-auth-00, Internet Engineering Task Force (Mar 2022), `https://datatracker.ietf.org/doc/draft-ietf-pquip-pqt-hybrid-terminology/00/`, work in Progress
4. Bindel, N., Brendel, J., Fischlin, M., Goncalves, B., Stebila, D.: Hybrid key encapsulation mechanisms and authenticated key exchange. In: IACR Cryptology ePrint Archive (2019)
5. Bindel, N., Herath, U., McKague, M., Stebila, D.: Transitioning to a quantum-resistant public key infrastructure. Cryptology ePrint Archive, Paper 2017/460 (2017), `https://eprint.iacr.org/2017/460`, `https://eprint.iacr.org/2017/460`
6. Boneh, D.: Twenty years of attacks on the rsa cryptosystem. Notices of the American Mathematical Society **46**, 203–212 (1999)

7. Castryck, W., Decru, T.: An efficient key recovery attack on sidh. Cryptology ePrint Archive, Paper 2022/975 (2022), https://eprint.iacr.org/2022/975, https://eprint.iacr.org/2022/975
8. Chen, L., Moody, D., Liu, Y.K.: Post-quantum cryptography. Tech. rep., National Institute of Standards (NIST) (2016), https://csrc.nist.gov/Projects/post-quantum-cryptography
9. D, F.: Terminology for Post-Quantum Traditional Hybrid Schemes. Internet-Draft draft-ietf-pquip-pqt-hybrid-terminology-00, Internet Engineering Task Force (May 2023), https://datatracker.ietf.org/doc/draft-ietf-pquip-pqt-hybrid-terminology/00/, work in Progress
10. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) Advances in Cryptology — CRYPTO' 86. pp. 186–194. Springer Berlin Heidelberg (1987)
11. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Ladner, R.E., Dwork, C. (eds.) 40th ACM STOC. pp. 197–206. ACM Press (May 2008). doi: 10.1145/1374376.1374407
12. Hale, B., Bindel, N., Van Bossuyt, D.L.: Handbook for management of threats, security and defense, resilience and optimal strategies: Quantum Computers – The Need for a New Cryptographic Strategy. To appear, Springer (2023)
13. Jao, D., Azarderakhsh, R., Campagna, M., Costello, C., De Feo, L., Hess, B., Hutchinson, A., Jalali, A., Karabina, K., Koziel, B., LaMacchia, B., Longa, P., Naehrig, M., Pereira, G., Renes, J., Soukharev, V., Urbanik, D.: Sike – supersingular isogeny key encapsulation. Tech. rep., https://sike.org/
14. Kaye, P., Laflamme, R., Mosca, M.: An Introduction to Quantum Computing. Oxford University Press, Inc., USA (2007)
15. Mumtaz, M., Ping, L.: Forty years of attacks on the rsa cryptosystem: A brief survey. Journal of Discrete Mathematical Sciences and Cryptography **22**, 29 – 9 (2019)
16. Ounsworth, M., Gray, J., Pala, M.: Composite Signatures For Use In Internet PKI. Internet-Draft draft-ounsworth-pq-composite-sigs-09, Internet Engineering Task Force (May 2023), https://datatracker.ietf.org/doc/draft-ietf-pquip-pqt-hybrid-terminology/00/, work in Progress