Recommendations for Quantum-safe Signature Solutions

HAPKIDO Report D5.3

Serge Fehr Julia Kastner

June 30, 2026

Abstract

This report aims at offering advice in finding suitable post-quantum secure digital signature solutions. As the choice of a suitable solution is dependent on the intended application, it is not possible to make a universal recommendation; rather, we offer guidance by means of information.

To this end, we provide an overview of the available post-quantum signature schemes that may be considered for replacing current non-quantum-secure solutions. We elaborate on their respective characteristics and their respective advantages and disadvantages. Furthermore, we discuss approaches for hybrid solutions, which combine post-quantum and conventional signature schemes via cryptographic combiners and so offer security even if one of the two component schemes gets broken. We point out the advantages and disadvantages of using such a hybrid solution, and we elaborate on certain features that in the literature are sometimes promoted as being desirable.

1 Introduction

1.1 Background

Digital Signatures. Since their introduction in the seminal work by Diffie and Hellman [DH76], where the idea of public-key cryptography was first articulated as a fundamentally new paradigm, and subsequently by Rivest, Shamir, and Adleman [RSA78], who provided the first concrete instantiation through the RSA scheme, *digital signature schemes* have played an indispensable role in cryptography, both in the theory as well as in practical applications.

Digital signatures are widely used for *identification* and *authentication*. The former refers to the ability to verify that an entity (such as a user, server, or organization) is who it claims to be, e.g., so that you can verify that you are indeed talking to your bank and not to someone pretending to be your bank. Authentication, on the other hand, ensures the integrity and origin of a message, i.e., it guarantees that the message was indeed created by the claimed sender and has not been altered during transmission (or storage). For example, you want to be sure that the software update you are about to install is legitimate and was indeed rolled out by the corresponding software company. Crucially, by means of digital signatures, these guarantees are achieved without requiring the involved parties to have any pre-shared secret information, distinguishing digital signatures from traditional symmetric solutions, like passwords or message authentication codes. This makes digital signatures particular suitable in open environments, where parties may have no prior relationship.

One of the most significant advantages of digital signatures over symmetric-key approaches is that they provide *non-repudiation*. This property ensures that once a party has signed a message, it cannot plausibly deny having done so at a later time. Unlike symmetric-key- or password-based systems, where the shared secrets make it difficult to attribute actions to a single party, digital signatures are produced with the help of a secret key that is known only to the signer, and so any valid signature can be uniquely linked to the signer. Therefore, similar to a handwritten signature on a piece of paper, a digital signature on a digital document provides legal assurance of the origin of the document and

undeniably confirms the approval by the party who has issued the signature, making the signer thus accountable for the message that (s)he has signed. Due to these properties, digital signatures are used in applications such as secure web communication, electronic contracts, software distribution, blockchain systems, digital identity systems, etc.

Post-quantum Security. Due to the discovery of Shor's quantum algorithm [Sho94] in 1994 for factoring integers (and computing discrete logarithms in any group), it is known that the public-key cryptography that has been used in applications like above over the last few of decades will become insecure once a scalable and universally computable quantum computer becomes available, which can run Shor's algorithm. In response to this threat, the *National Institute of Standards and Technology (NIST)* initiated the *Post-Quantum Cryptography (PQC) Standardization Process* in 2016, with the goal of identifying and standardizing suitable key encapsulation and digital signature schemes that are believed to be secure against quantum attacks, i.e., against adversaries that may run quantum algorithms. After several rounds of public evaluation and cryptanalysis, NIST selected a first set of such schemes. The selected digital signature scheme are now referred to as *ML-DSA*, *FN-DSA* and *SLH-DSA*. At the same time, NIST opened an additional call for post-quantum signature schemes in order to diversify the portfolio of standardized schemes, in particular with respect to the underlying hardness assumptions and the implementation characteristics. With originally 40 candidate schemes in round 1, this additional call has just proceeded to the 3rd round in May 2026, leaving 9 candidates remaining in the process.

Selecting a set of schemes to be standardized is an important first step; the next big step then is to integrate the new post-quantum standards into the current infrastructure. This migration is expected to be a gradual and complex process that requires careful integration into existing protocols, software ecosystems, and hardware platforms, while maintaining interoperability and possibly backward compatibility during the transition period.

An additional complication is that, in contrast to the relatively uniform landscape of classical public-key cryptography, there is no all-purpose replacement in the post-quantum case. The post-quantum schemes that can be considered are based on different computational hardness assumptions (in the realm of lattices, multivariate equations, error-correcting codes, etc.), each with distinct advantages and drawbacks. For example, some of the post-quantum signature schemes provide compact signatures but require large public keys, while others offer fast verification at the cost of expensive signing procedures or increased implementation complexity. On top, practical considerations such as side-channel resistance, ease of deployment, maturity of security analysis, and suitability for constrained hardware environments can vary substantially between schemes. As a consequence, the choice of an appropriate post-quantum signature scheme is a non-trivial task and in general application- and context-dependent, and it remains unclear whether a single construction will emerge in the near future as the dominant replacement across all use cases.

Hybrid Security and Cryptographic Combiners. Beyond the technical issues that result from larger keys and/or signatures, increased computational complexity, etc., an additional concern in the transition may be that these new schemes do not offer the same test-of-time assurance when it comes to their security (even against conventional classical attacks). While being unlikely given the scrutiny these schemes have undergone, the chances that one of these new post-quantum schemes unexpectedly turns out to be insecure against classical attacks is still larger than for the (factoring- and discrete-log-based) schemes that are to be replaced.

Certainly in theory, an attractive approach to mitigate this risk is by means of combining a test-of-time-approved (factoring- or discrete-log-based, and thus quantum-insecure) scheme, with a new post-quantum scheme that promises to offer security against future quantum attacks, in such a way that in order to break the combined scheme, both components need to be broken. Thus, once quantum attacks become feasible, we can rely on the security of the post-quantum secure component; on the other hand, should the post-quantum component unexpectedly turn out to be less secure (even against

classical attacks) than expected then we are hedged by the conventional scheme. A possible downside of such a hybrid solution is the additional complexity that results from having two schemes involved, as this may also increase the chances of implementation errors etc. Therefore, there is no common agreement on the suitability of this approach; for example, the EU countries are in general positive and recommend such a hybrid solution [Sec+24], the US voices reservations [Moo+24].

1.2 This Report

This report aims to facilitate decision-making on the selection of a post-quantum digital signature scheme, as well as on whether a hybrid solution should be adopted and, if so, which solution to choose. For this purpose, in the first part of this report, we discuss the three digital signature schemes chosen by NIST in the *PQC Standardization Process*, as well as the schemes that are currently still under consideration in the call for alternative post-quantum signature schemes, and we review their individual advantages and disadvantages. In the second part, we discuss the hybrid approach in detail: we put advantages and disadvantages next to each other, elaborate on the different techniques and what they achieve, and what are some limitations.

With respect to making a recommendation, we clearly advise the use of a NIST-approved solution: one out of *ML-DSA*, *FN-DSA*[1]and *SLH-DSA*, or, if (and only if) there is a good reason to not go for any of the three, then one of the remaining candidates in the ongoing NIST process for standardizing alternative post-quantum signature schemes. As mentioned above already, there is no single solution that is the right choice in every scenario. Instead, this report is meant to provide the information that should help in deciding which is an appropriate solution in the use-case at hand.

2 Preliminaries

For completeness, we provide here the formal definition of a digital signature scheme and its security.

Formally, a *(digital) signature scheme* is a triple

$$\Sigma = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrfy})$$

of PPT algorithms (where Vrfy is typically deterministic). On input the security parameter λ (in unary representation), KGen produces a secret-key/public-key pair (sk, pk). On input a secret-key sk and a message m (chosen from a set that is defined by pk), Sign produces a signature σ. Finally, on input a public-key pk, a message m and a signature σ, Vrfy outputs a bit indicating whether the signature is considered valid or not. When considering multiple signature schemes, say Σ_1, Σ_2 etc., we write $\Sigma_1.\mathsf{KGen}$ for the algorithm KGen corresponding to Σ_1, $\Sigma_2.\mathsf{KGen}$ for the algorithm KGen corresponding to Σ_2, etc.

Correctness requires that a correctly produced signature is considered valid, i.e., that $\mathsf{Vrfy}(pk, m, \sigma) = 1$ (almost) with certainty when $(sk, pk) \leftarrow \mathsf{KGen}(1^\lambda)$ and $\sigma \leftarrow \mathsf{Sign}(sk, m)$.

The formal security notion expected of a signature scheme Σ as above is *unforgeability against chosen-message attacks (UF-CMA)*. This security notion is formulated as a game between a challenger and an adversary. The challenger sets up a key pair $(sk, pk) \leftarrow \mathsf{KGen}$ and gives pk to the adversary. The adversary is then allowed to ask the challenger for signatures σ_i on messages m_i chosen by the adversary. The challenger keeps a list of the messages m_i it signed. Finally, the adversary outputs a message signature pair m^*, σ^*, and it wins the game if $\mathsf{Vrfy}(pk, m^*, \sigma^*) = 1$ and $m^* \neq m_i$ for all i, i.e. the message has not been signed before by the challenger and the signature produced by the adversary is valid w.r.t. his message. See Figure 1. Sometimes, EUF-CMA (short for *existential* unforgeability under chosen-message attacks) is used to mean the same thing as UF-CMA.

A signature scheme $\Sigma = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrfy})$ is *UF-CMA secure* if no probabilistic polynomial time (PPT) adversary has a non-negligible chance of winning the above game. The security notion can be

[1]Due to the higher implementation complexity, FN-DSA has a longer standardization timeline, and so, at the time of writing, it is does not have a FIPS standard yet, in contrast to ML-DSA and SLH-DSA.

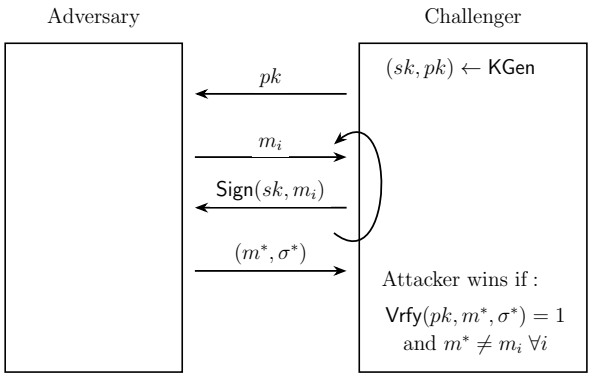

Figure 1: The UF-CMA security game.

strengthened to *strong* UF-CMA security (SUF-CMA) by relaxing the winning condition in such a way that the adversary also wins if it produces a new, *different* signature on a message that has been signed before, or weakened either to UF-naCMA (non-adaptive chosen message attacks) where the adversary is required to specify the messages before seeing the public key, or UF-NMA (no-message attacks) where the adversary does not get to ask for any signatures at all.

3 The Schemes Selected for Standardization

In this section, we recall and discuss the digital signature schemes selected by NIST in their (first) PQC Standardization Process: *CRYSTALS-Dilithium* (*ML-DSA*), *Falcon* (*FN-DSA*) and *SPHINCS*$^+$ (*SLH-DSA*). While the schemes *ML-DSA* and *SLH-DSA* have FIPS standards at the time of writing (204 and 205, respectively), *FN-DSA* is following a slower standardization path due its implementation complexity; this does not mean that the actual scheme is less mature or considered less secure.

3.1 CRYSTALS-Dilithium (ML-DSA)

The Scheme. The digital signature scheme *CRYSTALS-Dilithium* is standardized in *FIPS 204* under the name *ML-DSA*. It is a lattice-based scheme and follows the *Fiat-Shamir with Aborts* (FSwA) design principle. We refer to the specification document [Bai+21] for the details.

Somewhat simplified (for instance, in the actual scheme all the randomness is produced by expanding a short seed), the scheme works as follows. The public-key consists of a matrix A and a vector t with coefficients in the ring $R_q = \mathbb{Z}_q[X]/(X^n + 1)$ for suitable choices of n and q, so that

$$t = As_1 + s_2$$

for short vectors s_1 and s_2, which form the secret key. A signature for a message m then consists of a pair (z, c), which is computed by letting w_1 be the high-order bits of $w := Ay$ for a random short vector y, setting

$$c := H(m \| w_1) \qquad \text{and} \qquad z := y + cs_1 \,,$$

and retrying in case z turns out to be "too large". H is a public cryptographic hash function, which is modeled as a random oracle in the security analysis. For the verification, we note that

$$Az - ct = A(y + cs_1) - c(As_1 - s_2) = Ay - cs_2 = w - cs_2 \,,$$

which is close to w, and so verification works by computing the high-order bits w_1' of $w' := Az - ct$, and checking that $c = H(m \| w_1')$, and by checking that z is sufficiently small.

Security. *CRYSTALS-Dilithium* is lattice-based; concretely, the security relies on the *module learning-with-error* (MLWE) problem and (variants of) the *module short-integer-solution* (MSIS) problem. We mainly discuss the security against quantum attacks here.

Formally, the security of *CRYSTALS-Dilithium* follows a modular analysis. First, UF-CMA is reduced to UF-NMA by means of the generic results in [Bar+23; FFH26] (in the random oracle model). Neglecting small constants, the reduction comes with an additive security loss of roughly $q_S\sqrt{(q_{RO}+1)\epsilon}+\delta$, where q_{RO} and q_S are the respecting numbers of queries that the UF-CMA attacker can make to the random oracle and the signing oracle, and ϵ and δ are quantities that are computed dependent on the concrete parameters of the scheme.[2]

As for UF-NMA security, [KLS18] reduce the security tightly to the hardness of MLWE and the hardness of SelfTargetMSIS; the latter though is a non-standard assumption that is tailored to the scheme. It is mentioned in [KLS18] for the classical case, and argued in [JMW24] for the quantum case, that the hardness of SelfTargetMSIS can respectively be reduced to MSIS and to MLWE, with a non-trivial loss though.

For concrete security based on solving MLWE, [Bai+21, Table 4] offers the following estimates for the number of gates and the memory (in bits) needed to break the different NIST security levels of *CRYSTALS-Dilithium*:

	$\log_2$(#gates)	$\log_2$(#bits)
CRYSTALS-Dilithium Level 2	158.6	97.5
CRYSTALS-Dilithium Level 3	216.7	138.7
CRYSTALS-Dilithium Level 5	285.4	187.4

Bandwidth and Performance. The following table shows the public-key and signature sizes for the NIST security levels 2, 3 and 5, as provided in the specification, and performance numbers from [PQS]:

	public-key size (bytes)	signature size (bytes)	sign (cycles)	verify (cycles)
CRYSTALS-Dilithium Level 2	1312	2420	333,013	118,412
CRYSTALS-Dilithium Level 3	1952	3309	529,106	179,424
CRYSTALS-Dilithium Level 5	2592	4627	642,192	279,936

Bottom Line. *ML-DSA* offers reasonable public-key and signature sizes. It has a good all-around performance with fast signing and verification, and it admits simple constant-time implementations. The security relies on a computational mathematics problem in the realm of lattices. The problem involves structured lattices, which might potentially make it easier to attack it (compared to unstructured lattices), but so far the computational problem has successfully resisted quantum attacks or worrying quantum speed-ups. *ML-DSA* is the default recommendation by NIST.

3.2 Falcon (FN-DSA)

The Scheme. The signature scheme *Falcon* will be standardized as *FN-DSA* in *FIPS 206*. It is lattice-based and follows (a variant of) the *Hash-and-Sign* design principle put forward by [GPV08] (the so-called GPV framework) based on a preimage-sampleable trapdoor function. We refer to the specification [Fou+20] for more details.

Simplified, *Falcon* works as follows The public key consists of a ring element $h \in R_q = \mathbb{Z}_q[X]/(X^n+1)$, whose ideal then defines a lattice, and a suitable trapdoor for this lattice serves as secret key. To sign a message m, a random salt r is chosen, and the trapdoor is used to sample a short vector $(s_1, s_2) \in R_q^2$ such that $s_1 + h \cdot s_2 = H(m, r)$; the signature then consists of the triple (r, s_1, s_2). Since

[2]Concretely, ϵ the (average-case) min-entropy of the high-order bits of w, and δ is a parameter that can be chosen freely so as to optimize ϵ by means of avoiding certain undesirable choices of the key. For the different proposed NIST parameters, they are in the range $5 \cdot 10^{-499} \leq \epsilon \leq 10^{-132}$ and $2^{-99} \leq \delta \leq 2^{-641}$.

there is a slight chance that the above sampling procedure fails, the vector (s_1, s_2) is to be resampled until successful.

Recently, [GJK24] proposed and analyzed a slightly modified version called *Falcon*$^+$, and mentioned that this update has been incorporated into the latest implementation. It introduces a tweak by which not only the vector (s_1, s_2) is resampled (if necessary) but also the salt r.

Security. The security of *Falcon* is based on (some variants of) the *ring inhomogeneous short integer solution* (R-ISIS) problem for the ring $R = \mathbb{Z}[X]/(X^n + 1)$. The original *Falcon* scheme does not have a provable security guarantee, classically nor quantum. On the other hand, a variation called *Falcon*$^+$ comes with a security proof (in the random oracle model) against classical attacks. No quantum security proof for *Falcon*$^+$ has been spelled out, but should follow by using [DFM20] (for the NMA part) and [FFH26] (for the CMA part); however, certain classical tricks do not apply here, and so the reduction loss would be quite big.

We note that the lack of a formal security proof makes the case of *Falcon* less convincing from a theoretical perspective, but is generally not viewed as a concern or obstacle, neither by the NIST not the cryptographic community. *Falcon* does follow known design principles, but the current proof techniques fail due to some technical matters. As far as is known, those are solely obstacles to the proof techniques, but do not provide any actual attack opportunity.

As for concrete security, the complexity of the best attack is estimated to be as follows:

	$\log_2$(#gates)
Falcon-512 (Level 1)	143
Falcon-1024 (Level 5)	287

Bandwidth and Performance. The following table shows the public-key and signature sizes for the NIST security levels 1 and 5, as provided in the specification, and performance numbers from [PQS]:

	public-key size (bytes)	signature size (bytes)	sign (cycles)	verify (cycles)
Falcon-512 (Level 1)	897	666	1,009,764	81,036
Falcon-1024 (Level 5)	1793	1280	2,053,080	160,596

Bottom Line. *Falcon* offers by far the smallest signature sizes (among the NIST PQC standards). It has very fast verification but signing is somewhat complex due to the floating-point arithmetic with 53 bits of precision; this poses no problem for a typical software implementation, though it may prove to be a major limitation when implementing on constrained devices. There used to be some concern regarding constant-time implementations, but that seems to be resolved now [How+20]. Similarly to the case of *CRYSTALS-Dilithium*, the security of *Falcon* relies on a computational mathematics problem in the context of structured lattices, which so far has resisted quantum attacks or worrying quantum speed-ups. At the time of writing, *Falcon* does not have a FIPS standard yet; however, this is mainly a matter of time: *Falcon* is expected to become standardized as *FIPS 206* in late 2026 or in early 2027.

3.3 SPHINCS$^+$ (SLH-DSA)

The Scheme. The signature scheme *SPHINCS*$^+$ is standardized as *SLH-DSA* in *FIPS 205*. In contrast to *CRYSTALS-Dilithium* and *Falcon*, it follows a purely hash-based design and so offers less (mathematical) structure that could be exploited in an attack.

In more detail (while still being high-level), it relies on a Merkle tree construction, where the root acts as the public key of the signature scheme, and each leaf is (the hash of) the public key of (a variant of) the hash-based Winternitz one-time signature scheme (*WOTS*$^+$). To sign a message, a leaf in the tree is picked pseudorandomly (determined by the message), and then the key-pair associated to this

leaf is used to sign the message (and the corresponding authentication path to the root is provided). Since constructing the entire tree, as needed to compute the public-key, would be infeasible, the tree is actually organized as a Merkle tree of Merkle trees, where each lower-level tree is attached to a node of the parent tree by a signature (using the key pair associated to the node) on the root of the lower-level tree. This allows to produce the lower-level trees on the fly.

Security. *SPHINCS*$^+$ is proven secure against quantum and classical adversaries in the plain model (i.e. without relying on the random oracle model heuristic), based on a list of (partly non-standard) computational assumptions on the underlying hash function. Such hash-based computational assumptions are considered to be very conservative, as they do not offer any mathematical structure that could be exploited in an attack, and they are believed to be largely unaffected by quantum attacks (beyond the usual Grover-like speedups).

Bandwidth and Performance. The following table shows the public-key and signature sizes for the NIST security levels 1, 3 and 5, as provided in the specification, and performance numbers from [PQS]s, where for each level a *short* variant ('s') is provided, with shorter signatures, and a *fast* variant ('f'), with improves efficiency:

	public-key size (bytes)	signature size (bytes)	sign (cycles)	verify (cycles)
SPHINCS$^+$-128s (Level 1)	32	8080	4,682,570,992	4,764,084
SPHINCS$^+$-128f (Level 1)	32	16976	239,793,806	12,909,924
SPHINCS$^+$-192s (Level 3)	48	17064	8,091,419,556	6,465,506
SPHINCS$^+$-192f (Level 3)	48	35664	386,861,992	19,876,926
SPHINCS$^+$-256s (Level 5)	64	29792	7,085,272,100	10,216,560
SPHINCS$^+$-256f (Level 5)	64	49216	763,942,250	19,886,032

Bottom Line. *SPHINCS*$^+$ comes with very large signatures, and the signing process is orders of magnitudes slower, compared to *CRYSTALS-Dilithium* and *Falcon*. On the positive side, *SPHINCS*$^+$ is a conservative choice when it comes to security, given that it does not rely on any mathematical structure, which could be exploited in a (classical or quantum) attack. The security purely relies on the security of the underlying hash function.

3.4 Comparison and Recommendation

Among the digital signature schemes discussed above, each offers different trade-offs in terms of performance, signature size, and security assumptions. *ML-DSA* is generally considered the most balanced option, combining strong security with relatively simple and efficient implementations, making it well suited as a default choice for most applications. *FN-DSA* achieves significantly smaller signature sizes, which can be advantageous in bandwidth- or storage-constrained environments, but this comes at the cost of greater implementation complexity and higher computational overhead. *SLH-DSA*, in contrast, is purely hash-function based and so its security does not lie on any mathematically-structured computational problem (like lattice-based problems underlying *ML-DSA* and *FN-DSA*), thus relying on a more conservative assumption. However, this comes with substantially larger signatures and slower performance.

Consequently, *ML-DSA* is recommended as the default option for most use cases. *FN-DSA* is preferable when minimizing signature size is a primary requirement and the additional computational and implementation complexity can be accommodated. *SLH-DSA* is recommended for applications that prioritize conservative security assumptions and can tolerate the associated performance and size overhead.

4 The Schemes in the Current NIST Standardization Process

As mentioned in the introduction, NIST is running an additional standardization process for post-quantum signature schemes [Nis], in order to diversify the portfolio, given that so far only lattice- (and hash-) based signature schemes have been chosen for standardization. Below, we will briefly list and discuss the the candidate signature schemes that are still considered in this additional standardization process.

In order to facilitate their exposition, we first briefly explain the main construction paradigms for signature schemes. They are used very broadly, but in particular are exploited both by the standardized schemes discussed above, as well as by several of the schemes still in the ongoing standardization competition. The two main paradigms are *Hash-then-Sign* (as used by *FN-DSA*), and the *Fiat-Shamir* transform applied to a zero-knowledge proof system (as used by *ML-DSA*).

4.1 Construction Paradigms for Digital Signature Schemes

In the following, we briefly explain some common construction paradigms for signature schemes that are used both by the standardized schemes, as well as several of the schemes still in the race for standardization. The two main paradigms are Hash-then-Sign (as used by *FN-DSA*) and applying the Fiat-Shamir-transform to a zero-knowledge proof system (as used by *ML-DSA*).

4.1.1 The Fiat-Shamir Transform and Zero-Knowledge Proofs

One common way to construct a signature scheme is to apply the Fiat-Shamir transform to a public-coin zero-knowledge proof of knowledge. Often the signature is a proof of knowledge of the secret key corresponding to the verification key, or a proof that the verification key has the correct form.

The Fiat-Shamir Transform The Fiat-Shamir transform [FS87] is a way to turn a public-coin zero-knowledge proof into a non-interactive proof using a hash function. This transformation can be adapted to construct signature schemes from public-coin zero-knowledge proof systems. The transform works as follows: Instead of having the verifier sample a random challenge c, the challenge is derived as the hash $H(x, \text{trans})$ of the statement and the transcript so far, i.e., the first message from the prover. When constructing a signature, the message m is hashed as well, and the statement is the public verification key, so $c = H(pk, \text{trans}, m)$. To prove security, the hash function is modelled as a random oracle. In the following, we describe several ways to instantiate the public coin zero-knowledge proofs that are used in signatures.

Public-Coin Zero-Knowledge Proofs In an *interactive proof system*, a *prover* interacts with a *verifier* to convince the verifier that a statement x is true. We say that the system is *public coin* if the verifier's messages can be sampled randomly without knowing any secret inputs of the verifier. For the purpose of this document, we care about proof systems where, if given a secret witness w (think of w as "evidence" of the statement x), the prover is efficient.

Such proof systems are used in signatures scheme constructions with statements such as "I know the secret key for this public key" or "This public key has a certain property". Therefore, the following security properties are of interest:

- **Soundness:** This property guarantees that no (possibly computationally bounded) malicious prover can convince a verifier of that a false statement is true. In signature schemes, the statement here refers to the public key having a certain property.
- **Knowledge Soundness:** This property guarantees that no prover can convince the verifier if it doesn't know a witness. This is formalized via an extractor that can extract the witness from any successful prover. In some cases (e.g. Sigma-Protocols), a variant of this property is given

as *special soundness*, where several transcripts with the same first message from the prover, but different verifier coins can be used to compute a witness.

- **Zero-Knowledge** (and variants thereof) informally guarantees that from a proof, the verifier cannot learn any additional knowledge besides that the statement is true or that the prover knows a witness. In particular, the verifier does not learn anything about the witness. This is formalized with the existence of a *simulator* that can generate a convincing-looking transcript between prover and verifier without knowing the witness.

When ZK-protocols are used to build a signature scheme via the Fiat-Shamir transform, an appropriate soundness notion guarantees that an adversary that does not have the secret key cannot forge a signature. In the security reduction, there are usually two main strategies:

- When building on soundness, the security reduction would give the adversary a public key that does not have the "property" from above, but is indistinguishable from a "real" public key that does. Due to soundness, the adversary is now unable to compute a proof that convinces the verifier. The security of a scheme is then based on the hardness of telling the "real" and "fake" public keys apart.

- When building on knowledge soundness, the security reduction would use the extractor to extract the secret key corresponding to its own public key from the adversary. This extraction requires the adversary's forgery, so the security reduction must be able to generate signatures without the secret key as part of the UF-CMA game. The security of a scheme would be based on the hardness of obtaining a secret key from the public key.

As part of the UF-CMA game, the reduction needs to be able to provide the adversary with signatures, usually without having the secret key, or even in the case where we know it is not possible to efficiently. This can be achieved by using the *zero-knowledge simulator* along with programming the random oracle as extra leverage that the adversary does not have.

There are several classes of such protocols with slightly differing security guarantees, namely Sigma-protocols [Cra97], MPC-in-the-Head [Ish+07], and VOLE-in-the-Head [Bau+23] (Vector Oblivious Linear Evaluation in the Head). Depending on the type of statement one wants to prove, one class might be more suitable than another.

4.1.2 Hash-then-Sign

In Hash-then-Sign schemes, the message m is first hashed using a hash function H and then signed by the signature scheme. In some cases, this is done only to map an arbitrarily long message to a fixed size, but in others, the security of the scheme relies on the adversary not being able to pick the "message"(i.e. the hash to be signed). Several signature schemes follow the trapdoor-OWF (one-way function) structure. The public key describes a function f that can be evaluated efficiently, whereas to efficiently generate a preimage of a given value under f, a trapdoor information td is needed. The signature then is a value x such that $f(x) = H(m)$, and it is generated by the signer by computing $y = H(m)$ and then computing a preimage $x \in f^{-1}(y)$ using the trapdoor td. In order to forge a signature, the adversary needs to find either a hash collision, i.e. two messages m, m' with $H(m) = H(m')$, or it has to invert either f or H. Often, the hash function H is modelled as a random oracle in the security proof.

4.2 Alternative Digital Signature Schemes

The following signature schemes are still under consideration (at the time of writing) in the current round 3 of the NIST standardization process for additional signature schemes [Nis]. These alternative signature schemes are 9 out 40 candidates that were originally submitted to the NIST call. We present them here.

4.2.1 Isogeny-Based Signatures

SQISign SQISign relies on the hardness of computing the endomorphism ring of a supersingular elliptic curve. This is called the endomorphism ring problem. The signature is generated via Fiat-Shamir-transform of a Sigma-protocol that proves knowledge of an Endomorphism. We refer to the specification document for details [Aar+25].

4.2.2 Lattice-Based Signatures

HAWK The security of the HAWK signature scheme is based on the lattice isomorphism problem (for key recovery) and on the one-more shortest vector problem (for UF-CMA). The signature scheme is constructed following a Hash-then-Sign approach. For more details, we refer to the specification document [Bos+25].

4.2.3 MPC-in-the-Head Signatures

The following signatures are all based on the MPC-in-the-Head paradigm [Ish+07]. MPC stands for secure multi-party computation.

MQOM (MQ on my Mind) The MQOM signature scheme relies on the hardness of the Multivariate Quadratic (MQ) Problem which states that it is hard to find a solution to a set of random multivariate quadratic equations over a field $\mathbb{F}$. The signature is computed as a zero-knowledge proof of knowledge of a solution to a MQ instance using the Threshold-Computation-in-the-Head paradigm which generalizes MPCitH. For more details, we refer to the specification document [Ben+25].

SDitH (Syndrome Decoding in the Head) Signatures in SDitH are computed as a zero-knowledge proof of knowledge of a low-weight vector x solution of a syndrome decoding instance $y = Hx$, which is made non-interactive using the Fiat-Shamir transform. The proof is computed using VOLE-in-the-Head. For more information, see the specification document [Agu+25].

4.2.4 Multivariate Signatures

The following signature schemes rely on the hardness of finding solutions to multivariate polynomial equations, often multivariate quadratic equations.

UOV Unbalanced Oil and Vinegar secret keys consist of a *central map* $\mathcal{F}\colon \mathbb{F}_q^n \to \mathbb{F}_q^m$ and an *invertible* linear transformation $\mathcal{T}\colon \mathbb{F}_q^n \to \mathbb{F}_q^n$ which is used to hide the structure of $\mathcal{F}$. The central map has the special structure that it consists of multivariate quadratic polynomials $f^{(1)}, \ldots, f^{(m)}$ of the form

$$f^{(k)}(x_1, \ldots, x_n) = \sum_{i=1}^{v} \sum_{j=1}^{n} \alpha_{i,j}^{(k)} \cdot x_i x_j$$

where $v = n - m$. The variables $x_1, \ldots, x_v$ are called the vinegar variables and the remaining variables $x_{v+1}, \ldots, x_n$ are called oil variables. The special structure of $\mathcal{F}$ allows to efficiently compute preimages of $\mathcal{F}$ using gaussian elimination.

The public key is of the form $\mathcal{P} = \mathcal{F} \circ \mathcal{T}$. UOV follows a hash-then-sign approach where the message is hashed to $\vec{t} = \mathsf{Hash}(m)$. A signature is computed by finding a preimage $\vec{s} \in \mathcal{P}^{-1}$ as follows. First, sample a random vector $\vec{u} \leftarrow \mathbb{F}_q^v$. This induces a linear transformation $\mathcal{F}\left(\left[\begin{smallmatrix}\vec{u}\\ \cdot\end{smallmatrix}\right]\right)$ on $\mathbb{F}_q^m$. Solve using gaussian elimination for $\vec{w}$ such that $\mathcal{F}\left(\begin{bmatrix}\vec{u}\\ \vec{w}\end{bmatrix}\right) = \vec{t}$. Then, compute $\vec{s} = \mathcal{T}^{-1}\left(\left[\begin{smallmatrix}\vec{u}\\ \vec{w}\end{smallmatrix}\right]\right)$. Output $\vec{s}$ as the signature which is a preimage of $\vec{t}$ under $\mathcal{P}$. Verification checks that $\mathcal{P}(\vec{s}) = \mathsf{Hash}(m)$. For more information see the specification document [Beu+25b].

MAYO MAYO is a variant of UOV with smaller public keys. In particular, MAYO uses a smaller oil space with dimension $o \leq m$. However, this means that it may not always be possible to solve for a signature $\vec{s}$. MAYO resolves this by "whipping up" the public key $\mathcal{P}$ into a map $\mathcal{P}^* \colon \mathbb{F}_q^{kn} \to \mathbb{F}_q^m$ where k is a parameter of the scheme. The map $\mathcal{P}^*$ is defined by system parameters in such a way that it vanishes on the oil space and so that it is possible to sample signatures using the same method as for Oil and Vinegar. See the specification document for more details [Beu+25a].

QR-UOV Quotient-Ring-UOV builds on UOV but uses quotient rings with additional mathematical structure to reduce key size. The submitters compare this change to the choice of using MLWE instead of LWE. For more information, see the specification document [Fur+25].

SNOVA "Simple Non-commutative-ring based UOV with key-randomness Alignment" is based on UOV, but it uses a field $\mathbb{F}_q[S]$ as the underlying field. $\mathbb{F}_q$ is set as $GF(16)$ and S is chosen to be an $l \times l$ symmetric matrix with an irreducible characteristic polynomial. The submitters motivate these choices by shorter public key lengths and greater variability in parameter choices allowing users to pick a set of parameters that best suits their use case. For more information, see the specification document [Wan+25].

4.2.5 Symmetric-Based Signatures

FAEST FAEST is based on the hardness of breaking AES. A public key consists of a tuple (x, y) and the secret key is a symmetric AES key k such that $y = E_k(x)$, that is, y is an AES ciphertext of x under the key k. Signatures are derived from a non-interactive argument of knowledge of k using the VOLE-in-the-Head technique. We refer to the specification document [Bau+] for further details.

4.3 Final Notes on Other Signature Schemes

The schemes listed above have not been standardized yet, and hence one should proceed with caution and use the (soon to be) standardized schemes *ML-DSA*, *FN-DSA* and *SLH-DSA* whenever possible. If it is not possible to use a standardized scheme, one may consider using one of the schemes still in the race, but caution is in place, in particular also when it comes to the implementation.

5 Hybrid Security and Signature Combiners

5.1 Background

Hybrid security refers to security obtained by means of combining two (or more) different cryptographic schemes, e.g. by combining two encryption schemes into a new encryption scheme, or by combining two signature schemes into a new signature scheme. In the context of the PQC transition, one of the two schemes would typically be a conventional scheme and the other would be a post-quantum secure one. The precise meaning of the concept comes in two flavors, sometimes referred to as *hybrid-OR* and *hybrid-AND*, respectively.

A hybrid-OR combiner allows two cryptographic schemes (e.g. a conventional scheme and a PQC scheme) to be offered, with the protocol succeeding if either one can be used successfully. In practice, this often shows up as negotiation logic, where the peers attempt to use a PQC scheme if both sides support it, but transparently fall back to a conventional scheme otherwise. The OR property is therefore about operability; the system works as long as there is at least one scheme that is mutually supported. Thus, hybrid-OR designs enable incremental deployment without breaking connectivity, which is critical for large-scale systems. The main advantage is exactly this flexibility and backward compatibility, allowing systems to adopt PQC without requiring a flag day upgrade. However, on the negative side, this design introduces downgrade risks and makes the overall security only as strong as the weakest acceptable alternative, so careful downgrade protection and policy controls are essential.

The goal of hybrid-AND, on the other hand, is to ensure security even if one of the two component schemes is insecure, as long as the other one is secure. By combining a conventional with a post-quantum secure one in such a manner, one safeguards against the less-well understood post-quantum secure turning out to be insecure (or less secure as expected), making the transition from a conventional scheme to such a hybrid scheme a no-regret move.

In the remainder, if not mentioned otherwise, we will consider the AND-version of hybrid security.

5.2 Considerations

The advantage of using hybrid security is explained above: it offers security as long as one of the two component schemes is secure. This is certainly an attractive feature, in particular in the context of the PQC transition, given that the considered post-quantum secure schemes are relatively new, and mostly rely on computational problems that have not undergone the same "test of time" yet, compared to the conventional schemes that have been used over the last couple of decades.

However, there are also disadvantages. For instance, the computational complexity as well as the key- and signature- or ciphertext-sizes are (even) larger, though in the context of combining a conventional scheme with a post-quantum one, the relative overhead caused by the conventional one is small. Perhaps more relevant is that a hybrid solution introduces additional complexity in the implementation and management, and more code means more attack surface — in particular when it comes to attacks that are outside of the standard model (like RNG failure, side-channel leakage, etc.). This can be countered to some extent by having standardized hybrid solutions, rather than leaving it up to the application/implementation to decide which schemes to combine and how. Finally, if the hybrid solution is meant as a temporary solution only, then another transition is eventually required to move away from the conventional scheme for good.

For these reasons, there is no global agreement on whether to opt for hybrid security or not for the PQC transition; e.g., while the US has reservations due to the added complexity of using combiners [Moo+24], EU countries are generally in favour [Sec+24].

One might also take use-case related arguments into account, when deciding to go for hybrid security or not. For example, the *harvest-now-decrypt-later* attack applies very generally, and nicely illustrates the impact of a (possibly unexpected) future break of a cryptographic scheme that is currently deemed secure. Therefore, one can well justify that the safety-net against an enormous retroactive loss of confidentiality is worth the additional integration complexity for encryption and KEMs, except maybe in very special cases where confidentiality is clearly relevant only temporarily.

The situation is slightly more subtle for signature schemes though. In certain cases, signatures are used to merely authenticate the sender *at the time of sending the signature*, and if the signature scheme then turns out to be insecure in the future, this has no retrospective effect. To attack *authenticity*, the signature scheme has to be broken at the time of usage. So, in these cases, using a hybrid signature scheme may be considered overkill: if the signature scheme turns out to be insecure in the future, then the damage remains limited to a restricted time window (from the discovery of the vulnerability to the fix). However, certain signature applications also rely on the *non-repudiation* property of signature schemes, i.e., the guarantee to be able to convince a *third party* (like a judge) that the signer produced the signature. Here, the situation is very different: should the signature scheme turn out to be insecure in the future, all the signatures produced now lose their non-repudiation guarantee. Indeed, you can then not make a convincing case anymore that the signer produced the signature, since anyone can now forge signatures, so it could just as well be that you are trying to frame the person.

We also emphasize that a hybrid approach makes little sense for the specific post-quantum signature scheme *SLH-DSA*, whose security purely relies on the security of the underlying hash function. Indeed, hash-function security is a very conservative assumption that has a long history, and the chances of a unexpected break —classically or quantumly — are considered to be much smaller compared to the schemes whose security relies on mathematically structured computational problems.

5.3 Combiners for Signature Schemes

The technical tool to obtain hybrid security, i.e., to turn two or more component schemes into a new scheme that is at least as secure as the stronger of the two, is called a (cryptographic) *combiner*. Semi-formally, a combiner for (two) signature schemes is a transformation that turns any two signature schemes Σ_1 and Σ_2 into a new signature scheme Σ that is secure as long as at least one of Σ_1 and Σ_2 is secure. Constructing combiners for signature schemes is rather straightforward, and there is little motivation to use anything else than one of the following two.

The *concatenation combiner* works by producing two independent signatures of the message m that is to be signed: a signature σ_1 using the scheme Σ_1, and a signature σ_2 using the scheme Σ_2; the combined signature then consists of the pair (σ_1, σ_2). Verification works in the obvious way by verifying the two signatures $\sigma_1 = \Sigma_1(m)$ and $\sigma_2 = \Sigma_2(m)$ and accepting only if both are valid. Formally, given the component schemes Σ_1 and Σ_2, the combined signature scheme $\Sigma^{\|}$ is defined as specified in Figure 2.

Even if Σ_1 turns out to be insecure, so that σ_1 can be produced without the knowledge of the corresponding secret key, σ_2 still remains hard to forge (as long as Σ_2 is secure) — and vice versa.

$\Sigma^{\|}.\mathsf{KGen}(1^\lambda)$:	$\Sigma^{\|}.\mathsf{Sign}(sk, m)$:	$\Sigma^{\|}.\mathsf{Vrfy}(pk, m)$:
1: $(sk_1, pk_1) \leftarrow \Sigma_1.\mathsf{KGen}(1^\lambda)$	1: $\sigma_1 \leftarrow \Sigma_1.\mathsf{Sign}(sk_1, m)$	1: $v_1 := \Sigma_1.\mathsf{Vrfy}(pk_1, m)$
2: $(sk_2, pk_2) \leftarrow \Sigma_2.\mathsf{KGen}(1^\lambda)$	2: $\sigma_2 \leftarrow \Sigma_2.\mathsf{Sign}(sk_2, m)$	
3: $sk := (sk_1, sk_2)$, $pk := (pk_1, pk_2)$	3: $\sigma := (\sigma_1, \sigma_2)$	2: $v_2 := \Sigma_2.\mathsf{Vrfy}(pk_2, m)$
4: **return** (sk, pk)	4: **return** σ	3: **return** $v_1 \wedge v_2$

Figure 2: The concatenation combiner $\Sigma^{\|}$.

The *nested combiner* works quite similarly, except that now σ_2 is produced as a signature on the pair (m, σ_1), i.e., m is signed along with σ_1 under Σ_2. Formally, given the component schemes Σ_1 and Σ_2, the combined signature scheme Σ° is defined as specified in Figure 3.

Despite the asymmetry here, by same kind of reasoning as above, security of the combined signature scheme Σ is ensured as long as one of the two component scheme is secure.

$\Sigma^{\circ}.\mathsf{KGen}(1^\lambda)$:	$\Sigma^{\circ}.\mathsf{Sign}(sk, m)$:	$\Sigma^{\circ}.\mathsf{Vrfy}(pk, m)$:
1: $(sk_1, pk_1) \leftarrow \Sigma_1.\mathsf{KGen}(1^\lambda)$	1: $\sigma_1 \leftarrow \Sigma_1.\mathsf{Sign}(sk_1, m)$	1: $v_1 := \Sigma_1.\mathsf{Vrfy}(pk_1, m)$
2: $(sk_2, pk_2) \leftarrow \Sigma_2.\mathsf{KGen}(1^\lambda)$	2: $\sigma_2 \leftarrow \Sigma_2.\mathsf{Sign}(sk_2, (m, \sigma_1))$	
3: $sk := (sk_1, sk_2)$, $pk := (pk_1, pk_2)$	3: $\sigma := (\sigma_1, \sigma_2)$	2: $v_2 := \Sigma_2.\mathsf{Vrfy}(pk_2, m)$
4: **return** (sk, pk)	4: **return** σ	3: **return** $v_1 \wedge v_2$

Figure 3: The nested combiner Σ°.

The difference between the concatenation combiner and the nested combiner is marginal, with almost no difference in performance etc. (beyond that in the latter the second signature is to be computed on a slightly larger message). Both offer standard security (unforgeability under chosen message attacks) if at least one of the two component schemes does, with a straightforward and tight reduction.

A small difference in terms of (security) properties is that the latter prevents an attacker from turning a combined signature $\sigma = (\sigma_1, \sigma_2)$ for a message m into a signature under the component scheme Σ_2 for the same message m (since σ_2 is a signature on (m, σ_1) rather than on m) — if Σ_2 is secure. This is referred to as *2-non-separability* in [Bin+17], with the parameter 2 referring to

the hardness of extracting a signature for the *2nd* scheme Σ_2.[3] In the concatenation combiner, the combined signature *by construction* gives rise to signatures on m for both of the component schemes. Needless to say that in the nested combiner, the combined signature still gives rise to signatures on m for the first component scheme Σ_1.

In [Bin+17], this non-separability property is motivated as follows:

> "This security property is interesting in the context of a transition. Suppose a signer issues hybrid signatures during a transition. Suppose further that there is a verifier who can understand both hybrid signatures and single-scheme signatures, but possibly acts upon them differently. The goal of non-separability is to prevent an attacker from taking a hybrid signature and turning it into something that the verifier accepts as coming from a single-scheme signature — thereby misrepresenting the signer's original intention."

However, we find it hard to come up with a concrete use case, where the above is a real issue. In any application we anyway need a mechanism, typically in the form of a PKI (public key infrastructure), that declares what is a legitimate key (under what scheme) for what user (here: signer). Thus, either one designs the PKI in such a way that the components pk_1 and pk_2 of the combined key $pk = (pk_1, pk_2)$ are explicitly declared to be legitimate keys (for the respective component schemes) for the signer, because we *want* the signer to be held accountable also for parts of a combined signature, i.e., we want separability as a *feature*, or else by default the PKI specifies that the signer is associated to the public-key pk for the combined scheme, and with nothing beyond.[4] In the latter case, when facing a separable scheme, even though an adversary may be able to take a hybrid signature and turn it into something that is verified by the verification algorithm, say, of the first component scheme with key pk_1, but by the PKI's policy it would not be accepted as being a signature issued by the signer.

Furthermore, there is a simple, generic solution (which offers 1- *and* 2-non-separability simultaneously): instead of signing the message m (using your favourite combined signature scheme), sign m appended with a warning text like

```
``NB: Accept the signature to this message only when it comes as a
hybrid signature.''
```

A stronger notion of non-separability, *strong non-separability* (SNS), requires that, given a combined signature on a message m, it should be hard to produce a signature on an *arbitrary* message (chosen by the adversary) that verifies under one of the two component scheme (with the corresponding component key). It is intuitively quite clear that this notion is impossible to achieve with a combiner as considered here (see below though for "pseudo-combiners").

5.4 Signature Combiners for Strong Unforgeability

A stronger security notion for digital signature schemes than standard unforgeability is *strong unforgeability* (see Section 2). Here, the adversary wins when it succeeds in turning a valid signature σ on a message m into a *different* valid signature σ' for the same message m.[5] As a digital counterpart of a hand-written signature, it is not obvious why this kind of "forgery" should be avoided (i.e., a different signature on a message that has been legitimately signed), but this stronger notion of security turns out to be necessary in certain protocols, in order for the protocol to behave as required (when being attacked by an adversary).

It is easy to see that neither the concatenation nor the nested combiner preserve *strong* unforgeability in the sense of hybrid security. In the concatenation combiner, if either one of the two component

[3] The formal definition of *non-separability* provided in [Bin+17] is more complicated, involving a "recognizer" that is aware of the combined signature scheme; we do not go into these details here.

[4] If we also want the signer to be associated to keys for the component schemes as well, then these should be *fresh* keys. Reusing keys across different schemes is asking for trouble, whether hybrid schemes or not.

[5] We note that a digital signature scheme may well be randomized, so that in general there exist multiple valid signatures for a given message m.

schemes is not strongly unforgeable then also the combined scheme is not (even if the other component scheme is strongly unforgeable). Indeed, if Σ_1 is not strongly unforgeable then a combined signature $\sigma = (\sigma_1, \sigma_2)$ for a message m can be turned into a different signature $\sigma' = (\sigma_1', \sigma_2)$ for m simply by changing the signature σ_1 into a different signature σ_1' (by exploiting that Σ_1 is not strongly unforgeable); the same if Σ_2 is not strongly unforgeable. For the nested combiner, the situation is that the combined scheme is strongly unforgeable if and only if Σ_2 is; this means it is not a combiner for strong unforgeability (since it is not sufficient for Σ_1 to be strongly unforgeable), but the problem is slightly milder here as it is one-sided. In particular, if Σ_2 is a signature scheme with unique signatures (which is typically the case when the signing procedure is deterministic) then for the combined scheme to *not* be strongly unforgeable it is necessary that Σ_2 is not unforgeable in the ordinary sense.[6]

It is not too hard to see that, at least intuitively, the same problem occurs for *any* combiner construction for digital signature schemes. There is always an "outermost" signature, and if that one can be turned into a different signature for the same message then this breaks the strong unforgeability of the combined scheme. The matter is slightly more subtle (e.g., it could be that the choice of the "outer most" signature depends on the value of the signature, and so by changing this value one also changes the choice), and finding a full-fledged formal argument is the subject of an ongoing project. However, it seems rather clear that any *practical* signature combiner does not preserve strong unforgeability for the above reason.

5.5 Pseudo-combiners

By definition, a combiner is limited to black-box access to the component schemes. This means that its inner working is independent of the inner workings of the component schemes; it merely makes queries to the KGen, Sign and Vrfy algorithms of the component schemes (like, it asks one component scheme to sign this message, and the other component scheme to sign that message), and it is totally oblivious to how these algorithms actually compute the answers to the queries. Similarly, the security of the combined scheme should only exploit the security of (one of) the component schemes, but again should not depend on how they actually work. After all, we want the combiner to do its job no matter how the component schemes are instantiated.

In contrast to the above, in the literature one also finds suggestions for hybrid signature schemes that are not constructed in such a black-box manner, but where the component schemes are combined in a non-black-box way by exploiting their inner workings. Such constructions typically only work for certain choices of the component schemes, like schemes that follows a certain specific design principle (see [DGR25; Jan26] for a recent such examples).

Such hybrid schemes can offer stronger notions of security, like strong non-separability, or strong unforgeability (even if one of the component schemes is not strongly unforgeable). However, the distinction between such a hybrid signature scheme that combines two (particularly designed) component schemes in a non-black-box manner, and a stand-alone scheme that recycles and intertwines the design principles of the component schemes, is not clear cut. As a matter of fact, we are not aware of a convincing formal definition of a "hybrid signature scheme" that is obtained by means of non-black-box combining the component schemes, which would distinguish such a scheme from a stand alone scheme whose security relies on one-out-of-two computational problems being hard. As such, we find it somewhat hard to appreciate this line of work from a purely theoretical perspective. However, having said that, by means of this approach one may well obtain a hybrid signature scheme that is practically interesting and provides security against quantum attacks (as long as the considered quantum-hard computational problem remains quantum hard), but still offers conventional classical security in case the quantum-hard computational problem turns out to be easier to solve than believed.

A disadvantage is that, since, strictly speaking, it is a new signature scheme, it needs to undergo a similar kind of scrutiny (and possibly standardization efforts) as any cryptographic scheme that we want to use in real-life applications; one cannot blindly (i.e. in a black-box way) rely on the security

[6] Well, given that Σ_2 has unique signatures, there is no difference between strong and ordinary unforgeability.

of the component schemes, as is the case when using a standard (black-box) combiner. From this perspective, combining two standardized/certified/approved/etc. component signature schemes via a simple standard combiner, appears to be a more practical solution.

5.6 Recommendation

If one decides to go for digital signatures with hybrid security, then our recommendation (which features nothing surprising but reflects the discussions within this document) is as follows.

We see no strong motivation to use anything else than the obvious concatenation or composition combiner. Among the two, the former might be preferable given the symmetric nature, but the latter is (just as) fine as well. For typical use cases, both offer what is needed in terms of hybrid security: (standard) security if at least one of the component schemes is secure. Achieving anything beyond standard security is a subtle matter and requires special attention and inspection.

As for the particular choice of the signature schemes to combine (in the context of the PQC transition), we recommend to use the "NIST approved" PQC scheme that fits best for the considered use case (in terms of signature size, computational complexity etc.) and combine it with ECDSA (or EdDSA) or RSA. By "NIST approved" we mean one of the NIST standards *ML-DSA* or *FN-DSA* (as pointed out, there is little motivation for using *SLH-DSA* combined with a conventional scheme), or possibly one of the remaining schemes in the still ongoing NIST PQC standardization process for additional digital signature schemes, once the process has reached a suitable state.

Ideally, such a combined signature scheme would be cast as a *single* digital signature scheme, with a standardized implementation, where the inner workings of the scheme (like that the public key actually consists of two public keys, the signing procedure runs two signing algorithms, etc.) are abstracted away and can then be safely ignored when using the (combined) signature scheme; very much like one can ignore the inner workings of an ordinary, "non-combined" signature scheme. Then, the combined scheme can just be treated like any signature scheme when it comes to using the scheme in protocols, certificates etc., and issues like "where to put the second key?" do not occur. However, it turns out that there is little effort into standardizing combined signature schemes in that sense, which then leaves it to the application (the protocol, the certificate, etc.) to handle the "combining". This introduces complications, which could then be a source for mistakes, and thus requires extra care.

References

[Aar+25] Marius A. Aardal, Gora Adj, Diego F. Aranha, Andrea Basso, Isaac Andrés Canales Martínez, Jorge Chávez-Saab, Maria Corte-Real Santos, Pierrick Dartois, Luca De Feo, Max Duparc, Jonathan Komada Eriksen, Tako Boris Fouotsa, Décio Luiz Gazzoni Filho, Basil Hess, David Kohel, Antonin Leroux, Patrick Longa, Luciano Maino, Michael Meyer, Kohei Nakagawa, Hiroshi Onuki, Lorenz Panny, Sikhar Patranabis, Christophe Petit, Giacomo Pope, Krijn Reijnders, Francisco Rodríguez Henríquez, Sina Schaeffler, and Benjamin Wesolowski. *SQIsign Algorithm specifications and supporting documentation Version 2.0*. Feb. 2025. URL: `https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/round-2/spec-files/sqisign-spec-round2-web.pdf`.

[Agu+25] Carlos Aguilar Melchor, Slim Bettaieb, Lo¨ıc Bidoux, Thibauld Feneuil, Philippe Gaborit, Nicolas Gama Shay Gueron, James Howe, Andreas H¨ulsing, David Joseph, Antoine Joux, Mukul Kulkarni, Edoardo Persichetti, Tovohery H. Randrianarisoa, Matthieu Rivain, and Dongze Yue. *The Syndrome Decoding in the Head (SD-in-the-Head) Signature Scheme Algorithm Specifications and Supporting Documentation – Version 2.0*. Feb. 2025. URL: `https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/round-2/spec-files/sdith-spec-round2-web.pdf`.

[Bai+21] Shi Bai, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. *CRYSTALS-Dilithium Algorithm Specifications and Supporting Documentation (Version 3.1)*. Feb. 2021. URL: https://pq-crystals.org/dilithium/data/dilithium-specification-round3-20210208.pdf.

[Bar+23] Manuel Barbosa, Gilles Barthe, Christian Doczkal, Jelle Don, Serge Fehr, Benjamin Grégoire, Yu-Hsuan Huang, Andreas Hülsing, Yi Lee, and Xiaodi Wu. "Fixing and Mechanizing the Security Proof of Fiat-Shamir with Aborts and Dilithium". In: *CRYPTO 2023, Part V*. Ed. by Helena Handschuh and Anna Lysyanskaya. Vol. 14085. LNCS. Springer, Cham, Aug. 2023, pp. 358–389. DOI: 10.1007/978-3-031-38554-4_12.

[Bau+] Carsten Baum, Ward Beullens, Lennart Braun, Cyprien Delpech de Saint Guilhem, Michael Klooß, Christian Majenz, Shibam Mukherjee, Emmanuela Orsini, Sebastian Ramacher, Christian Rechberger, Lawrence Roy, and Peter Scholl. *FAEST v2: Algorithm Specifications Version 2.0*. URL: https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/round-2/spec-files/faest-spec-round2-web.pdf.

[Bau+23] Carsten Baum, Lennart Braun, Cyprien Delpech de Saint Guilhem, Michael Klooß, Emmanuela Orsini, Lawrence Roy, and Peter Scholl. "Publicly Verifiable Zero-Knowledge and Post-Quantum Signatures from VOLE-in-the-Head". In: *CRYPTO 2023, Part V*. Ed. by Helena Handschuh and Anna Lysyanskaya. Vol. 14085. LNCS. Springer, Cham, Aug. 2023, pp. 581–615. DOI: 10.1007/978-3-031-38554-4_19.

[Ben+25] Ryad Benadjila, Charles Bouillaguet, Thibauld Feneuil, and Matthieu Rivain. *MQOM: MQ on my Mind Algorithm Specifications and Supporting Documentation (Version 2.1)*. Sept. 2025. URL: https://mqom.org/docs/mqom-v2.1.pdf.

[Beu+25a] Ward Beullens, Fabio Campos, Sof´ıa Celi, Basil Hess, and Matthias J. Kannwischer. *MAYO Round 2 version*. 2025. URL: https://pqmayo.org/assets/specs/mayo-round2.pdf.

[Beu+25b] Ward Beullens, Ming-Shing Chen, Jintai Ding, Boru Gong, Matthias J. Kannwischer, Jacques Patarin, Bo-Yuan Peng, Dieter Schmidt, Cheng-Jhih Shih, Chengdong Tao, and Bo-Yin Yang. *UOV: Unbalanced Oil and Vinegar Algorithm Specifications and Supporting Documentation Version 2.0*. Feb. 2025. URL: https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/round-2/spec-files/uov-spec-round2-web.pdf.

[Bin+17] Nina Bindel, Udyani Herath, Matthew McKague, and Douglas Stebila. "Transitioning to a Quantum-Resistant Public Key Infrastructure". In: *PQCrypto 2017*. Ed. by Tanja Lange and Tsuyoshi Takagi. Springer, Cham, 2017, pp. 384–405. DOI: 10.1007/978-3-319-59879-6_22.

[Bos+25] Joppe W. Bos, Olivier Bronchain, Léo Ducas, Serge Fehr, Yu-Hsuan Huang, Thomas Pornin, Eamonn W. Postlethwaite, Thomas Prest, Ludo N. Pulles, and Wessel van Woerden. *HAWK version 1.1 (February 5, 2025)*. Feb. 2025. URL: https://hawk-sign.info/hawk-spec.pdf.

[Cra97] Ronald Cramer. "Modular Design of Secure yet Practical Cryptographic Protocols". PhD thesis. Jan. 1997.

[DFM20] Jelle Don, Serge Fehr, and Christian Majenz. "The Measure-and-Reprogram Technique 2.0: Multi-round Fiat-Shamir and More". In: *CRYPTO 2020, Part III*. Ed. by Daniele Micciancio and Thomas Ristenpart. Vol. 12172. LNCS. Springer, Cham, Aug. 2020, pp. 602–631. DOI: 10.1007/978-3-030-56877-1_21.

[DGR25] Julien Devevey, Morgane Guerreau, and Maxime Roméas. *Compact, Efficient and Non-Separable Hybrid Signatures*. Cryptology ePrint Archive, Report 2025/2059. 2025. URL: https://eprint.iacr.org/2025/2059.

[DH76] Whitfield Diffie and Martin E. Hellman. "New Directions in Cryptography". In: *IEEE Transactions on Information Theory* 22.6 (1976), pp. 644–654. DOI: 10.1109/TIT.1976.1055638.

[FFH25] Pouria Fallahpour, Serge Fehr, and Yu-Hsuan Huang. *Tighter Quantum Security for Fiat-Shamir-with-Aborts and Hash-and-Sign-with-Retry Signatures*. Cryptology ePrint Archive, Report 2025/985. 2025. URL: https://eprint.iacr.org/2025/985.

[FFH26] Pouria Fallapour, Serge Fehr, and Yu-Hsuan Huang. "Tighter Quantum Security for Fiat-Shamir-with-Aborts and Hash-and-Sign-with-Retry Signatures". In: *Advances in Cryptology – CRYPTO 2026*. Ed. by Nadia Heninger and Mike Rosulek. To appear. IACR. Cham: Springer Nature Switzerland, Aug. 2026. URL: https://eprint.iacr.org/2025/985.

[Fou+20] Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Prest, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. *Falcon: Fast-Fourier Lattice-based Compact Signatures over NTRU Specification v1.2 – 01/10/2020*. Oct. 2020.

[FS87] Amos Fiat and Adi Shamir. "How to Prove Yourself: Practical Solutions to Identification and Signature Problems". In: *CRYPTO'86*. Ed. by Andrew M. Odlyzko. Vol. 263. LNCS. Springer, Berlin, Heidelberg, Aug. 1987, pp. 186–194. DOI: 10.1007/3-540-47721-7_12.

[Fur+25] Hiroki Furue, Yasuhiko Ikematsu, Fumitaka Hoshino, Tsuyoshi Takagi, Haruhisa Kosuge, Kimihiro Yamakoshi, Rika Akiyama, Satoshi Nakamura, Shingo Orihara, and Koha Kinjo. *QR-UOV*. Feb. 2025. URL: https://info.isl.ntt.co.jp/crypt/qruov/files/qruov_Spec-v2.0.pdf.

[GJK24] Phillip Gajland, Jonas Janneck, and Eike Kiltz. *A Closer Look at Falcon*. Cryptology ePrint Archive, Report 2024/1769. 2024. URL: https://eprint.iacr.org/2024/1769.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. "Trapdoors for hard lattices and new cryptographic constructions". In: *40th ACM STOC*. Ed. by Richard E. Ladner and Cynthia Dwork. ACM Press, May 2008, pp. 197–206. DOI: 10.1145/1374376.1374407.

[How+20] James Howe, Thomas Prest, Thomas Ricosset, and Mélissa Rossi. "Isochronous Gaussian Sampling: From Inception to Implementation". In: *PQCrypto 2020*. Ed. by Jintai Ding and Jean-Pierre Tillich. Springer, Cham, 2020, pp. 53–71. DOI: 10.1007/978-3-030-44223-1_4.

[Ish+07] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. "Zero-knowledge from secure multiparty computation". In: *39th ACM STOC*. Ed. by David S. Johnson and Uriel Feige. ACM Press, June 2007, pp. 21–30. DOI: 10.1145/1250790.1250794.

[Jan26] Jonas Janneck. "Bird of Prey: Practical Signature Combiners Preserving Strong Unforgeability". In: *Advances in Cryptology - EUROCRYPT 2026 - 45th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Rome, Italy, May 10-14, 2026, Proceedings, Part II*. Ed. by Joan Daemen and Emmanuel Thomé. Vol. 16542. Lecture Notes in Computer Science. Springer, 2026, pp. 222–250. DOI: 10.1007/978-3-032-25317-0_8. URL: https://doi.org/10.1007/978-3-032-25317-0_8.

[JMW24] Kelsey A. Jackson, Carl A. Miller, and Daochen Wang. "Evaluating the Security of CRYSTALS-Dilithium in the Quantum Random Oracle Model". In: *EUROCRYPT 2024, Part VI*. Ed. by Marc Joye and Gregor Leander. Vol. 14656. LNCS. Springer, Cham, May 2024, pp. 418–446. DOI: 10.1007/978-3-031-58751-1_15.

[KLS18] Eike Kiltz, Vadim Lyubashevsky, and Christian Schaffner. "A Concrete Treatment of Fiat-Shamir Signatures in the Quantum Random-Oracle Model". In: *EUROCRYPT 2018, Part III*. Ed. by Jesper Buus Nielsen and Vincent Rijmen. Vol. 10822. LNCS. Springer, Cham, 2018, pp. 552–586. DOI: 10.1007/978-3-319-78372-7_18.

[Moo+24] Dustin Moody, Ray Perlner, Andrew Regenscheid, Angela Robinson, and David Cooper. *Transition to Post-Quantum Cryptography Standards*. NIST Internal Report NIST IR 8547 ipd. NIST, Nov. 2024. URL: https://doi.org/10.6028/NIST.IR.8547.ipd.

[Nis] *Call for Additional Digital Signature Schemes for the Post-Quantum Cryptography Standardization Process*. Tech. rep. Sept. 2022. URL: https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/call-for-proposals-dig-sig-sept-2022.pdf.

[PQS] PQShield. *Post-Quantum signatures zoo*. Accessed 23-03-2026. URL: https://pqshield.github.io/nist-sigs-zoo/.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". In: *Communications of the Association for Computing Machinery* 21.2 (Feb. 1978), pp. 120–126. DOI: 10.1145/359340.359342.

[Sec+24] Secure Information Technology Center Austria, Centre for Cybersecurity Belgium, National Cyber and Information Security Agency Czech Republic, Centre for Cyber Security Denmark, Information System Authority Estonia, Finnish transport and Communication Agency, French National Agency for the Security of Information Systems, Federal Office for Information Security Germany, National Cyber Security Authority Hellenic Republic, National Cyber Security Centre Ireland, National Cybersecurity Agency Italy, Ministry of Defense Latvia, National Cyber Security Centre Ministry of Defense Lithuania, High Commission for National Protection Luxemburg, Netherlands National Communication Security Agency, Ministry of Interior and Kingdom Relations Netherlands, National Cyber Security Centre Ministry of Security and Justice Netherlands, Research and Academic Research Center Poland, Government Information Security Office Slovenia, and National Cryptologic Center Spain. *Securing Tomorrow, Today: Transitioning to Post-Quantum Cryptography*. www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Crypto/PQC-joint-statement.pdf. Accessed 2026-01-16. Nov. 2024.

[Sho94] Peter W. Shor. "Algorithms for Quantum Computation: Discrete Logarithms and Factoring". In: *35th FOCS*. IEEE Computer Society Press, Nov. 1994, pp. 124–134. DOI: 10.1109/SFCS.1994.365700.

[Wan+25] Lih-Chung Wang, Chun-Yen Chou, Jintai Ding, Yen-Liang Kuan, Jan Adriaan Leegwater, Ming-Siou Li, Bo-Shu Tseng, Po-En Tseng, and Chia-Chun Wang. *SNOVA Proposal for NISTPQC: Additional Digital Signature Schemes Version 2.0*. Jan. 2025. URL: https://snova.pqclab.org/files/SNOVA_Round2.pdf.