Transitioning to a Quantum-Resistant Public Key Infrastructure

Nina Bindel[1] Udyani Herath[2] Matthew McKague[2]
Douglas Stebila[3]

[1] *Technische Universität Darmstadt, Darmstadt, Germany*
[2] *Queensland University of Technology, Brisbane, Australia*
[3] *McMaster University, Hamilton, Ontario, Canada*

`nbindel@cdc.informatik.tu-darmstadt.de`, `udyani.mudiyanselage@hdr.qut.edu.au`, `matthew.mckague@qut.edu.au`, `stebilad@mcmaster.ca`

May 24, 2017

Abstract

To ensure uninterrupted cryptographic security, it is important to begin planning the transition to post-quantum cryptography. In addition to creating post-quantum primitives, we must also plan how to adapt the cryptographic infrastructure for the transition, especially in scenarios such as public key infrastructures (PKIs) with many participants. The use of hybrids—multiple algorithms in parallel—will likely play a role during the transition for two reasons: "hedging our bets" when the security of newer primitives is not yet certain but the security of older primitives is already in question; and to achieve security and functionality both in post-quantum-aware and in a backwards-compatible way with not-yet-upgraded software.

In this paper, we investigate the use of hybrid digital signature schemes. We consider several methods for combining signature schemes, and give conditions on when the resulting hybrid signature scheme is unforgeable. Additionally we address a new notion about the inability of an adversary to separate a hybrid signature into its components. For both unforgeability and non-separability, we give a novel security hierarchy based on how quantum the attack is. We then turn to three real-world standards involving digital signatures and PKI: certificates (X.509), secure channels (TLS), and email (S/MIME). We identify possible approaches to supporting hybrid signatures in these standards while retaining backwards compatibility, which we test in popular cryptographic libraries and implementations, noting especially the inability of some software to handle larger certificates.

Contents

1 Introduction

Since the initial advent of modern symmetric and public key cryptography in the 1970s, there have only been a handful of transitions from one widely deployed algorithm to another. These include: from DES and Triple-DES to AES; from MD5 and SHA-1 to the SHA-2 family; from RSA key transport and finite field Diffie–Hellman to elliptic curve Diffie–Hellman key exchange; and from RSA and DSA certificates to ECDSA certificates. Some of these transitions have gone well: AES is nearly ubiquitous today, and modern communication protocols predominantly use ECDH key exchange. Transitions involving public key infrastructure have a more mixed record: browser vendors and CAs have had a long transition period from SHA-1 to SHA-2 in certificates, with repeated delay of deadlines; the transition to elliptic curve certificates has been even slower, and still today the vast majority of certificates issued for the web use RSA.

In the medium-term, we are likely to see another transition to *post-quantum* public key cryptography. Some aspects of the post-quantum transition will be straightforward: using post-quantum key exchange in protocols that support negotiation such as the Transport Layer has already been demonstrated [7, 8], and can be adopted piecewise. Other migrations will be harder, especially when it is difficult for old and new configurations to operate simultaneously. A recent whitepaper [10] discusses some of these issues at a high level.

The transition to post-quantum cryptography is further complicated by the relative immaturity of some of the underlying mathematical assumptions in current candidates: because they have not been studied for very long, there is a higher risk that they might be insecure. This motivates *hybrid* operation, in which both a traditional algorithm and one or more post-quantum algorithms are used in parallel: as long as one of them remains unbroken, confidentiality or authenticity can be ensured. This leads us to three research questions:

1. What are the appropriate security properties for hybrid digital signatures?
2. How should we combine signature schemes to construct hybrid signatures?
3. How can hybrid signatures be realized in popular standards and software, ideally in a backwards-compatible way?

1. Security notions for hybrid digital signatures. The widely accepted security notion for digital signatures is *unforgeability under chosen message attack* (EUF-CMA): the adversary interacts with a signing oracle to obtain signatures on any desired messages, and then must output a forgery on a new message. Hybrid signatures should retain that property. Boneh and Zhandry [6] first studied security notions for digital signature schemes against quantum adversaries, and gave a quantum analogue of EUF-CMA in which a quantum adversary is able to interact with a quantum signing oracle and thereby obtain signatures on quantum states of its choosing (which may be in superposition).

As we transition to post-quantum digital signatures, Boneh and Zhandry's definition might be overly strong: for example, we might be using a signature scheme for the next five years, and during this period we are confident that no adversary has a quantum computer, and moreover we are definitely not signing anything in superposition; but later, the adversary may eventually be able to use a quantum computer. We describe several security notions depending on how quantum the adversary is. We use the notation $\mathsf{X}^{\mathsf{y}}\mathsf{Z}$ to denote the adversary's type with respect to three options:

- X: whether the adversary is classical ($\mathsf{X} = \mathsf{C}$) or quantum ($\mathsf{X} = \mathsf{Q}$) *during* the period in which it can interact with the signing oracle;
- y: whether the adversary can interact with the signing oracle classically ($\mathsf{y} = \mathsf{c}$) or quantumly ($\mathsf{y} = \mathsf{q}$); and
- Z: whether the adversary is classical ($\mathsf{Z} = \mathsf{C}$) or quantum ($\mathsf{Z} = \mathsf{Q}$) *after* the period in which it

$$\begin{tikzcd}
\mathsf{C^cC} \arrow[r, bend left, "\text{Thm. 2}", "/" marking] & \mathsf{C^cQ} \arrow[l, bend left, "\text{Thm. 1}"] \arrow[r, bend left, "\text{Thm. 3}", "/" marking] & \mathsf{Q^cQ} \arrow[l, bend left, "\text{Thm. 1}"] \arrow[r, bend left, "\text{Thm. 4}", "/" marking] & \mathsf{Q^qQ} \arrow[l, bend left, "\text{Thm. 1}"]
\end{tikzcd}$$

Figure 1: Implications and separations between unforgeability notions ($\mathsf{X^yZ\text{-}eufcma}$) for signature schemes. $B \to A$ denotes an implication: every B-secure scheme is also A-secure. $A \not\to B$ denotes a separation: there exist A-secure schemes that are not B-secure.

can interact with the signing oracle.

These security notions form a natural hierarchy as shown in Figure 1, with separations between each level of the hierarchy.

We describe a second security property specifically related to hybrid signatures, called *non-separability*: for a hybrid signature involving two (or more) signature schemes, is it possible for an adversary to separate the hybrid signature into a valid signature in any of the component signature schemes? 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. Specifically, if $\Sigma' = C(\Sigma_1, \Sigma_2)$ is the hybrid signature scheme from using combiner C to combine signature schemes Σ_1 and Σ_2, then we say Σ' is $\mathsf{X^yZ}$-τ-nonsep if it is hard for an $\mathsf{X^yZ}$-adversary to construct a valid Σ_τ signature given access to a signing oracle for Σ'. Our notions are aided by a *recognizer* algorithm, which a verifier can apply to a signature to attempt to help distinguish separated hybrid signatures.

2. Signature combiners. Having laid out security properties for hybrid signatures, we proceed to investigate how to construct hybrid schemes using *combiners*. Table 1 shows the combiners we consider: concatenation and three forms of nested signatures. These particular combiners are motivated by two factors: that they are fairly natural constructions, and three of them arise in our applications.

3. Hybrid signatures in standards and software. Our goal is to provide a guide for how to transition to hybrid post-quantum digital signatures in various standards and software. We consider three standards: X.509 for certificates [12], the Transport Layer Security (TLS) protocol for secure channels [14], and Cryptographic Message Syntax (CMS) [19] as part of Secure/Multipurpose Internet Mail Extensions (S/MIME) [21] for secure email. For each, we ask:

3.a) How can hybrid / multiple signature schemes be used in the standard?
3.b) Is this approach backwards-compatible with old software?
3.c) Are there potential problems involving large public keys or signatures?

We identify promising techniques for hybrid X.509 certificates, and hybrid S/MIME signed messages; using multiple signature algorithms in TLS does not seem immediately possible, though one mechanism in the current draft of TLS 1.3 seems to allow multiple client authentications and a recent proposal could allow multiple server authentications. The software we tested had no problems with certificates or extensions up to ∼10 kB (accommodating ideal lattice schemes), and some software up to ∼80 kB (accommodating hash-based schemes), but none could handle the megabyte-plus size public keys of the largest lattice-based scheme; details appear in Section 5.

Combiner	Combined signature $\sigma = (\sigma_1, \sigma_2)$	Unforgeability	Non-separability
Single-message combiners			
$C_{\|}$	$\sigma_1 \leftarrow\$ \mathsf{Sign}_1(m); \sigma_2 \leftarrow\$ \mathsf{Sign}_2(m)$	$\max\{\mathsf{X^yZ}, \mathsf{U^vW}\}$	no
$C_{\text{weak-nest}}$	$\sigma_1 \leftarrow\$ \mathsf{Sign}_1(m); \sigma_2 \leftarrow\$ \mathsf{Sign}_2(\sigma_1)$	$\mathsf{X^yZ}$	$\mathsf{U^cW}$-2
$C_{\text{str-nest}}$	$\sigma_1 \leftarrow\$ \mathsf{Sign}_1(m); \sigma_2 \leftarrow\$ \mathsf{Sign}_2((m, \sigma_1))$	$\max\{\mathsf{X^yZ}, \mathsf{U^vW}\}$	$\mathsf{U^cW}$-2
Dual-message combiners			
D_{nest}	$\sigma_1 \leftarrow\$ \mathsf{Sign}_1(m_1); \sigma_2 \leftarrow\$ \mathsf{Sign}_2((m_1, \sigma_1, m_2))$	$\mathsf{U^vW}$, $\mathsf{X^yZ}^*$	$\mathsf{U^cW}$-2

Unforgeability: If Σ_1 is $\mathsf{X^yZ}$-eufcma and Σ_2 is $\mathsf{U^vW}$-eufcma, then $C(\Sigma_1, \Sigma_2)$ is ...-eufcma.
Non-separability: If Σ_1 is $\mathsf{X^yZ}$-eufcma and Σ_2 is $\mathsf{U^vW}$-eufcma, then $C(\Sigma_1, \Sigma_2)$ is ...-nonsep.
* Unforgeability of $D_{\text{nest}}(\Sigma_1, \Sigma_2)$ under $\mathsf{X^yZ}$-eufcma-security of Σ_1 in a restricted sense.

Table 1: Combiners for constructing hybrid signatures using schemes Σ_1 and Σ_2; $\max\{\mathsf{X^yZ}, \mathsf{U^vW}\}$ denotes the stronger unforgeability notion with respect to the natural hierarchy of security notions shown in Figure 1.

2 Signature schemes and unforgeability

We review definitions of signature schemes and existential unforgeability by classical and quantum adversaries, then extend these to account for partially-quantum adversaries.

Notation. Let S be a finite set. We write $s \leftarrow\$ \mathcal{U}(S)$, or simply $s \leftarrow\$ S$, to indicate that an element s is sampled uniformly at random from S. If A is a probabilistic algorithm, $y \leftarrow\$ A(x)$ denotes running A with input x and fresh random coins, and assigning the output to y.

Definition 1 (Signature scheme)**.** A *digital signature scheme* Σ is a tuple $\Sigma = (\Sigma.\mathsf{KeyGen}, \Sigma.\mathsf{Sign}, \Sigma.\mathsf{Verify})$ of algorithms:

- $\Sigma.\mathsf{KeyGen}() \; \$\!\!\rightarrow (sk, vk)$: The probabilistic key generation algorithm that returns a secret or signing key sk and public or verification key $vk \in \mathcal{VK}_\Sigma$.
- $\Sigma.\mathsf{Sign}(sk, m) \; \$\!\!\rightarrow \sigma$: The probabilistic signature generation algorithm which takes as input a signing key sk and a message $m \in \mathcal{M}_\Sigma$, and outputs a signature $\sigma \in \mathcal{S}_\Sigma$. The space of random coins is $\mathcal{R}_\Sigma$.
- $\Sigma.\mathsf{Verify}(vk, m, \sigma) \to 0$ or 1: The verification algorithm which takes as input a verification key vk, a message m, and a signature σ, and returns a bit $b \in \{0, 1\}$. If $b = 1$, we say that the algorithm accepts, otherwise we say that it rejects the signature σ for message m.

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.
We say that Σ is *ϵ-correct* if, for every message m in the message space, we have that

$$\Pr\left[\mathsf{Verify}(vk, m, \sigma) = 1 : (sk, vk) \leftarrow\$ \mathsf{KeyGen}(), \sigma \leftarrow\$ \mathsf{Sign}(sk, m)\right] \geq 1 - \epsilon$$

where the probability is taken over the randomness of the probabilistic algorithms.

2.1 Unforgeability security definitions

The standard definition of security for signature schemes is *existential unforgeability under chosen message attack* (EUF-CMA). In the traditional formulation of unforgeability dating back to Goldwasser, Micali, and Rivest [17], the adversary can obtain q_S signatures via signing oracle queries, and must output one valid signature on a message not queried to the oracle. This cannot be directly quantized: if the adversary is allowed to query oracles in superposition, we cannot restrain the adversary's forgery to be on a *new* message since the experiment cannot keep a copy of messages queried to the signing oracle for later checking.

An equivalent formulation in the classical setting demands the adversary output $q_S + 1$ valid signatures on distinct messages. Boneh and Zhandry [6] use this formulation to give a quantum analogue of EUF-CMA.[1] That notion involves a fully quantum adversary throughout so is quite strong. We envision a hierarchy of intermediate notions, distinguishing between whether the honest parties are signing classical or quantum messages (i.e., does the adversary have access to a classical or quantum signing oracle?) and whether the adversary is classical or quantum during the period it has access to the signing oracle (i.e., are we concerned about a quantum adversary only in the future, or also now?).

Our hierarchy is as follows:

1. *Fully classical* ($\mathsf{C}^\mathsf{c}\mathsf{C}$): The adversary at all times uses only a classical computer, and obtains signatures (and possibly hash values) from a classical signing oracle. This corresponds to the traditional EUF-CMA notion.
2. *Future quantum* ($\mathsf{C}^\mathsf{c}\mathsf{Q}$): The adversary uses only a classical computer during the period of time it is interacting with the (classical) signing oracle. At a later point in time, the adversary may use a quantum computer, but no longer has access to a signing oracle. However, the adversary has quantum acces to a hash oracle in the random oracle model. $\mathsf{C}^\mathsf{c}\mathsf{Q}$ models the scenario of a system which is in use today, but where parties eventually stop signing new documents; however, the documents signed with the system need to remain unforgeable for a long time, even after quantum computers become available.
3. *Quantum adversary, classical queries* ($\mathsf{Q}^\mathsf{c}\mathsf{Q}$) : The adversary at all times can use a quantum computer. However, the honest participants in the system only ever use their signing keys on classical computers, so the signing oracle is always accessed classically.
4. *Fully quantum* ($\mathsf{Q}^\mathsf{q}\mathsf{Q}$): The adversary at all times can use a quantum computer, obtains signatures from a quantum signing oracle, and obtains hash values from a quantum random oracle. This corresponds to Boneh and Zhandry's notion.

In our notation $\mathsf{X}^\mathsf{y}\mathsf{Z}$, X denotes the type of adversary ($\mathsf{X} = \mathsf{C}$ for classical or Q for quantum) while the adversary is able to interact with the signing oracle; y denotes the type of access the adversary has to the signing oracle; and Z denotes the type of adversary after it no longer has access to its signing oracle. Combinatorially, there are $2^3 = 8$ possibilities, but some do not make sense, such as $\mathsf{C}^\mathsf{q}\mathsf{Z}$ or $\mathsf{Q}^\mathsf{y}\mathsf{C}$. Figure 2 shows our unified definition for EUF-CMA parameterized for any of the four types of adversaries in the standard model, i.e., without access to a random (or hash) oracle. It follows the EUF-CMA formulation of Boneh and Zhandry [6] but separates out the adversary to be a two-stage adversary $(\mathcal{A}_1, \mathcal{A}_2)$, where $\mathcal{A}_1$ (of type X) interacts with either a signing oracle (of type y) and outputs an intermediate state *st* (of type X), which $\mathcal{A}_2$ (of type Z) then processes. The input to $\mathcal{A}_1$ and the output of $\mathcal{A}_2$ are always classical.

Figure 3 shows how the experiment is altered in the classical or quantum random oracle model: at the start of the experiment, a random function H is sampled uniformly from the space of all such functions $\mathcal{H}_\Sigma$. (In the classical setting, it is common to formulate the random oracle using lazy sampling, but such a formulation does not work in the quantum setting.) For simplicity, we assume the adversary has quantum random oracle access whenever it is quantum.

[1]A brief overview of notation for quantum computing appears in Appendix A.

```
$\mathrm{Expt}_{\Sigma}^{\mathsf{X^yZ\text{-}eufcma}}(\mathcal{A}_1, \mathcal{A}_2)$:
1  $q_S \leftarrow 0$
2  $(sk, vk) \leftarrow\!\!\$\ \Sigma.\mathsf{KeyGen}()$
3  $st \leftarrow\!\!\$\ \mathcal{A}_1^{\mathcal{O}_S(\cdot)}(vk)$
4  $((m_1^*, \sigma_1^*), \ldots, (m_{q_S+1}^*, \sigma_{q_S+1}^*)) \leftarrow\!\!\$\ \mathcal{A}_2(st)$
5  If $(\Sigma.\mathsf{Verify}(vk, m_i^*, \sigma_i^*) = 1\ \forall\ i \in [1, q_S + 1])$
   $\wedge \left(m_i^* \neq m_j^*\ \forall\ i \neq j\right)$:
6    Return 1
7  Else return 0
```

```
Classical signing oracle $\mathcal{O}_S(m)$:
8  $q_S \leftarrow q_S + 1$
9  $\sigma \leftarrow\!\!\$\ \Sigma.\mathsf{Sign}(sk, m)$
10 Return $\sigma$ to $\mathcal{A}$
```

```
Quantum signing oracle $\mathcal{O}_S(\sum_{m,t,z} \psi_{m,t,z} |m, t, z\rangle)$:
11 $q_S \leftarrow q_S + 1$
12 $r \leftarrow\!\!\$\ \mathcal{R}_\Sigma$
13 Return state $\sum_{m,t,z} \psi_{m,t,z} |m, t \oplus \Sigma.\mathsf{Sign}(sk, m; r), z\rangle$ to $\mathcal{A}$
```

Figure 2: Unified security experiment for $\mathsf{X^yZ\text{-}eufcma}$ in the standard model: existential unforgeability under chosen-message attack of a signature scheme Σ for a two-stage adversary $\mathcal{A}_1$ (of type X), $\mathcal{A}_2$ (of type Z) with signing oracle of type y; if $\mathsf{y} = \mathsf{c}$ then $\mathcal{A}_1$ has classical access to the signing orcale, otherwise quantum access.

```
$\mathrm{Expt}_{\Sigma}^{\mathsf{X^yZ\text{-}eufcma}}(\mathcal{A}_1, \mathcal{A}_2)$:
0  $H \leftarrow\!\!\$\ \mathcal{H}_\Sigma$
1  $q_H \leftarrow 0$, $q_S \leftarrow 0$
2  $(sk, vk) \leftarrow\!\!\$\ \Sigma.\mathsf{KeyGen}()$
3  $st \leftarrow\!\!\$\ \mathcal{A}_1^{\mathcal{O}_S(\cdot), \mathcal{O}_H(\cdot)}(vk)$
4  $((m_1^*, \sigma_1^*), \ldots, (m_{q_S+1}^*, \sigma_{q_S+1}^*))$
       $\leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}_H(\cdot)}(st)$
5  // continues as in Figure 2
```

```
Classical random oracle $\mathcal{O}_H(x)$:
14 $q_H \leftarrow q_H + 1$
15 Return $H(x)$
Quantum random oracle
$\mathcal{O}_H(\sum_{x,t,z} \psi_{x,t,z} |x, t, z\rangle)$:
16 $q_H \leftarrow q_H + 1$
17 Return state $\sum_{x,t,z} \psi_{x,t,z} |x, t \oplus H(x), z\rangle$
```

Figure 3: $\mathsf{X^yZ\text{-}eufcma}$ experiment in the classical and quantum random oracle models; if $\mathsf{X} = \mathsf{C}$, then $\mathcal{A}_1$ has classical access to the random oracle, otherwise quantum access; similarly for Z and $\mathcal{A}_2$.

We define advantage as

$$\mathrm{Adv}_{\Sigma}^{\mathsf{X^yZ\text{-}eufcma}}(\mathcal{A}) = \Pr\left[\mathrm{Expt}_{\Sigma}^{\mathsf{X^yZ\text{-}eufcma}}(\mathcal{A}) = 1\right] \ .$$

The strongly unforgeable variant can be obtained by adapting line 5 of Figure 2 in an analogous way to the discussion in Section 2.1.

2.2 Separations and implications

Our family of notions $\mathsf{C^cC}$-, $\mathsf{C^cQ}$-, $\mathsf{Q^cQ}$-, $\mathsf{Q^qQ\text{-}eufcma}$ form a natural hierarchy. Figure 1 shows the implications and separations between these notions. These implication induce an ordering on security notions, so we sometimes write $\mathsf{C^cC} \leq \mathsf{C^cQ}$ etc. Note, the stronger the security notion the smaller the advantage of an adversary $\mathcal{A}$ breaking a signature scheme of corresponding security. For example, let the signature scheme Σ be $\mathsf{C^cQ}$-secure. $\mathsf{C^cQ}$ is stronger than $\mathsf{C^cC}$, i.e., $\mathsf{C^cQ} \geq \mathsf{C^cC}$. Hence, $\mathrm{Adv}_{\Sigma}^{\mathsf{C^cQ\text{-}eufcma}}(\mathcal{A}) \leq \mathrm{Adv}_{\Sigma}^{\mathsf{C^cC\text{-}eufcma}}(\mathcal{A})$. Similarly we use $\max\{\cdot, \cdot\}$ based on this ordering.

The implications in Figure 1 are straightforward. Each of the separations $A \not\Longrightarrow B$ follows from a common technique: from an A-secure scheme Σ, construct a (degenerate) A-secure scheme Σ' that is not B-secure, because the additional powers available to a B-adversary allow it to recover the secret signing key of Σ that was cleverly embedded somewhere in Σ'.

- $\mathsf{C^cC}$-eufcma $\not\Longrightarrow$ $\mathsf{C^cQ}$-eufcma: In the public key for Σ', include a copy of the signing secret key encrypted using an RSA-based public key encryption scheme. Assuming breaking RSA is classically hard, the encrypted signing key is useless to a $\mathsf{C^cC}$-adversary, but a $\mathsf{C^cQ}$-adversary will be able to break the public key encryption, recover the signing key, and forge signatures.
- $\mathsf{C^cQ}$-eufcma $\not\Longrightarrow$ $\mathsf{Q^cQ}$-eufcma: In the public key for Σ', include an RSA-encrypted random challenge string, and redefine the $\Sigma'.\mathsf{Sign}$ so that, if the adversary queries the signing oracle on the random challenge string, the signing key is returned. Assuming breaking RSA is hard for a classical algorithm, a $\mathsf{C^cQ}$-adversary will not be able to recover the challenge while it has access to the signing oracle, and thus cannot make use of the degeneracy to recover the signing key; a $\mathsf{Q^cQ}$ adversary can.
- $\mathsf{Q^cQ}$-eufcma $\not\Longrightarrow$ $\mathsf{Q^qQ}$-eufcma: Here we hide the secret using a query-complexity problem that can be solved with just a few queries by a quantum algorithm making queries in superposition, but takes exponential queries when asking classical queries. The specific problem we use is a variant of the *hidden linear structure* problem [13].

In the following we state and prove the above statements formally.

Theorem 1 ($\mathsf{Q^qQ} \Longrightarrow \mathsf{Q^cQ} \Longrightarrow \mathsf{C^cQ} \Longrightarrow \mathsf{C^cC}$)**.** *If Σ is a $\mathsf{Q^qQ}$-eufcma-secure signature scheme, then Σ is also $\mathsf{Q^cQ}$-eufcma-secure. If Σ is a $\mathsf{Q^cQ}$-eufcma-secure signature scheme, then Σ is also $\mathsf{C^cQ}$-eufcma-secure. If Σ is a $\mathsf{C^cQ}$-eufcma-secure signature scheme, then Σ is also $\mathsf{C^cC}$-eufcma-secure.*

Proof. Suppose $(\mathcal{A}_1, \mathcal{A}_2)$ is an adversary against $\mathsf{C^cC}$-eufcma: both $\mathcal{A}_1$ and $\mathcal{A}_2$ are classical. Every classical algorithm is also a quantum algorithm, and thus they win the $\mathsf{C^cQ}$-eufcma experiment with at least the same probability as they win the $\mathsf{C^cC}$-eufcma experiment: $\mathrm{Adv}_{\Sigma}^{\mathsf{C^cQ\text{-}eufcma}}(\mathcal{A}_1, \mathcal{A}_2) \geq \mathrm{Adv}_{\Sigma}^{\mathsf{C^cC\text{-}eufcma}}(\mathcal{A}_1, \mathcal{A}_2)$. Similarly, $\mathrm{Adv}_{\Sigma}^{\mathsf{Q^cQ\text{-}eufcma}}(\mathcal{A}_1, \mathcal{A}_2) \geq \mathrm{Adv}_{\Sigma}^{\mathsf{C^cQ\text{-}eufcma}}(\mathcal{A}_1, \mathcal{A}_2)$.

Finally, an adversary $(\mathcal{A}_1, \mathcal{A}_2)$ against $\mathsf{Q^cQ}$-eufcma is also an adversary against $\mathsf{Q^qQ}$-eufcma that simply does not query its oracle in superposition, and thus

$$\mathrm{Adv}_{\Sigma}^{\mathsf{Q^qQ\text{-}eufcma}}(\mathcal{A}_1, \mathcal{A}_2) \geq \mathrm{Adv}_{\Sigma}^{\mathsf{Q^cQ\text{-}eufcma}}(\mathcal{A}_1, \mathcal{A}_2) \ .$$

□

Theorem 2 ($\mathsf{C^cC} \not\Longrightarrow \mathsf{C^cQ}$)**.** *If the RSA problem is hard for classical computers and there exists a signature scheme Σ that is $\mathsf{C^cC}$-eufcma-secure, then there exists a signature scheme Σ' that is $\mathsf{C^cC}$-eufcma-secure but not $\mathsf{C^cQ}$-eufcma-secure.*

Proof. Let Π be a public key encryption scheme that is IND-CPA-secure against classical adversaries and whose security relies on the hardness of the RSA problem, e.g., [16] or OAEP [4]. However, a quantum adversary could use Shor's algorithm to factor the modulus and decrypt ciphertexts encrypted using Π. We construct a scheme Σ' that is based on Σ, but the public key of Σ' includes a Π-encrypted copy of the Σ secret key:

- $\Sigma'.\mathsf{KeyGen}()$: $(sk, vk) \leftarrow\$ \Sigma.\mathsf{KeyGen}()$. $(dk, ek) \leftarrow\$ \Pi.\mathsf{KeyGen}()$. $c \leftarrow\$ \Pi.\mathsf{Enc}(ek, sk)$. $vk' \leftarrow (vk, ek, c)$. Return (sk, vk').
- $\Sigma'.\mathsf{Sign}(sk, m)$: Return $\Sigma.\mathsf{Sign}(sk, m)$.
- $\Sigma'.\mathsf{Verify}(vk' = (vk, ek, c), m, \sigma)$: Return $\Sigma.\mathsf{Verify}(vk, m, \sigma)$.

The theorem then follows as a consequence of the following two claims. □

Claim 1. *If Π is IND-CPA-secure against a classical adversary and Σ is $\mathsf{C^cC}$-eufcma-secure, then Σ' is $\mathsf{C^cC}$-eufcma-secure.*

Proof. The proof follows from a simple substitution. Since Π is IND-CPA-secure, no passive adversary can distinguish Π-encryptions of sk from encryptions of $0^{|sk|}$ with significant advantage. So we can replace c in vk' with an encryption of zeros, while still successfully simulating answers to the signing oracle in the $\mathsf{C^cC}$-eufcma experiment. A Σ' forgery is immediately a Σ forgery. $\square$

Claim 2. *If there exists an efficient quantum adversary $\mathcal{A}$ against the message recovery of* Π*, then* Σ' *is not* $\mathsf{C^cQ}$-eufcma*-secure.*

Proof. This proof is obvious. Given Σ' verification key $vk' = (vk, ek, c)$, run $\mathcal{A}$ on ek and c to recover sk. We can now forge signatures in Σ' by using the signing algorithm with sk. $\square$

Theorem 3 ($\mathsf{C^cQ} \not\Rightarrow \mathsf{Q^cQ}$)**.** *If the RSA problem is hard for classical computers and there exists a signature scheme* Σ *that is* $\mathsf{C^cQ}$-eufcma*-secure, then there exists a signature scheme* Σ' *that is* $\mathsf{C^cQ}$-eufcma*-secure but not* $\mathsf{Q^cQ}$-eufcma*-secure.*

Proof. Let Π be a public key encryption scheme that is IND-CPA-secure against classical adversaries and whose security relies on the hardness of the RSA problem. However, a quantum adversary could use Shor's algorithm to factor the modulus and decrypt ciphertexts encrypted using Π. Our signature scheme Σ' is designed so that the signing oracle can be used by an online quantum adversary to learn the signing key, but not by an offline quantum adversary. Recall that in the proof of Theorem 2, an encrypted copy of the secret signing key was provided to the adversary in the public verification key. Here, we put an encrypted *random challenge* in the public verification key, and if the adversary asks for that challenge to be signed, we have the signing oracle return the signing key. Intuitively, only an adversary that can break the challenge while it has access to the signing oracle (i.e., a quantum stage-1 adversary) can solve the challenge. The scheme Σ' is shown below.

- $\Sigma'.\mathsf{KeyGen}()$: $(sk, vk) \leftarrow_\$ \Sigma.\mathsf{KeyGen}()$. $(dk, ek) \leftarrow_\$ \Pi.\mathsf{KeyGen}()$. $s^* \leftarrow_\$ \{0,1\}^{256}$. $ch \leftarrow \Pi.\mathsf{Enc}(ek, s^*)$. $vk' \leftarrow (vk, ek, ch)$. $sk' \leftarrow (sk, s^*)$. Return (sk', vk').
- $\Sigma'.\mathsf{Sign}(sk' = (sk, s^*), m)$: If $m = s^*$, return sk. Else, return $\Sigma.\mathsf{Sign}(sk, m)$.
- $\Sigma'.\mathsf{Verify}(vk' = (vk, ek, ch), m, \sigma)$: Return $\Sigma.\mathsf{Verify}(vk, m, \sigma)$.

Note first that if Σ is ϵ-correct, then Σ' is $(\epsilon + \frac{1}{2^{256}})$-correct. The theorem is a consequence of the following two claims. $\square$

Claim 3. *If* Π *is IND-CPA-secure against a classical adversary and* Σ *is* $\mathsf{C^cQ}$-eufcma*-secure, then* Σ' *is* $\mathsf{C^cQ}$-eufcma*-secure.*

Proof. The proof of this claim is as follows. Since Π is IND-CPA-secure, no passive adversary can distinguish Π-encryptions of s^* from encryptions of 0^{256} with significant advantage. An adversary could guess s^* with a probability of $\frac{q_S}{2^{256}}$, where q_S is the number of signing queries. Therefore, we can replace ch in vk' with an encryption of zeros, while still successfully simulating answers to the signing oracle in the $\mathsf{C^cQ}$-eufcma experiment. At the quantum stage, without access to the signing oracle, an adversary is unable to simulate answers. An Σ' forgery is immediately an Σ forgery, so the claim follows. $\square$

Claim 4. *If there exists an efficient quantum adversary $\mathcal{A}$ against the message recovery of* Π*, then* Σ' *is not* $\mathsf{Q^cQ}$-eufcma*-secure.*

Proof. Suppose $\mathcal{A}$ breaks the message recovery of Π, i.e., decrypts ciphertexts encrypted using Π. Consider an algorithm $\mathcal{B}_1$ which uses $\mathcal{A}$ to decrypt ch and recover the correct solution s^*, which it can then query to its signing oracle to obtain the signing key sk. $\mathcal{B}_1$ will return a valid forgery. Taking $\mathcal{B}_2$ as the identity function, $\mathcal{B}_1, \mathcal{B}_2$ is a forger for Σ' if $\mathcal{A}$ is a decrypter for Π. $\square$

Theorem 4 ($\mathsf{Q^cQ} \not\Rightarrow \mathsf{Q^qQ}$)**.** *Assuming there exists a quantum-secure pseudorandom family of permutations, and a signature scheme Σ that is* $\mathsf{Q^cQ}$-eufcma*-secure, then there exists a signature scheme Σ' that is* $\mathsf{Q^cQ}$-eufcma*-secure but not* $\mathsf{Q^qQ}$-eufcma*-secure.*

Similar to Theorem 3, we will construct a signature scheme where the secret key is hidden behind a problem which is hard for some adversaries and easy for others. In Theorem 3 the problem was computational, and easily solved by the additional computational resources available to a quantum computer. Here, the problem with be an oracle problem. In an oracle problem, we are given access to an oracle that implements a specific function, and asked to determine some property of the oracle with as few calls to the oracle as possible. The minimal number of calls to the oracle required is called the *query complexity*. For some oracle problems, being able to query the oracle in superposition reduces the query complexity exponentially over classical calls to the oracle. The specific oracle problem we use is a variant of the *hidden linear structure* problem [13], which has a constant query complexity with quantum oracle access and exponential query complexity with classical oracle access. Note that it is the type of access to the oracle which determines the complexity — having access to quantum computation does not decrease the query complexity if we only have classical access to the oracle.

Let us for now suppose that a suitable query complexity problems exist. We will later construct one from a known example in the literature. To be more precise, suppose that there exists a family of functions $\mathcal{B}_{s,t} : GF(2^k) \times GF(2^k) \to GF(2^k) \times GF(2^k)$ such that, given oracle access to $\mathcal{B}_{s,t}$, at least $c_{\mathcal{B}}$ classical queries are required to determine s with probability greater than $p_{\mathcal{B}}$, whereas a single query suffices when given access to $\mathcal{B}_{s,t}$ via a quantum oracle. In our construction, s will be a secret which will unlock access to the signing key. The second parameter, t, needs to be secret but is otherwise not important for our application.

Our construction starts with a $\mathsf{Q^cQ}$-eufcma-secure signature scheme Σ. For our purposes, we will need $\Sigma.\mathsf{Sign}$ to be deterministic. That is, for a particular message and signing key the signature should always be the same. If this is not the case, then we can use standard techniques to make it so, for example by providing randomness through a quantum-secure PRF applied to the signing key and the message. Let us suppose that it takes at least time c_Σ for an adversary to win the $\mathsf{Q^cQ}$-eufcma security game with probability at least p_Σ.

We will need to address several parts of messages for signing. For a message m we will define $m.x, m.y, m.z$ to be bits 1 to 256, bits 257 to 512, and bits 513 to 768 of m, respectively. In particular, m must be at least 768 bits long. Bits beyond 768 will play no special role in the signing algorithm, but remain part of the message. Also let $\delta_{a,b}$ be the Kronecker delta, which is 1 when $a = b$ and 0 otherwise.

We now define our signature scheme Σ' as follows:

- $\Sigma'.\mathsf{KeyGen}()$: $(sk, vk) \leftarrow_\$ \Sigma.\mathsf{KeyGen}()$. $s \leftarrow_\$ \{0,1\}^{256}$. $t \leftarrow_\$ \{0,1\}^{256}$. $vk' \leftarrow (vk)$. $sk' \leftarrow (sk, s, t)$. Return (sk', vk').
- $\Sigma'.\mathsf{Sign}(sk', m)$: Return $(\Sigma.\mathsf{Sign}(sk, m), \mathcal{B}_{s,t}(m.x, m.y), sk \cdot \delta_{s,m.z})$.
- $\Sigma'.\mathsf{Verify}(vk', m, (\sigma, u, v, w))$: If $\Sigma.\mathsf{Verify}(vk, m, \sigma)$ accepts, $(u, v) = \mathcal{B}_{s,t}(m.x, m.y)$ and $w = sk \cdot \delta_{m.z,s}$ then accept, otherwise reject.

Since we are interested in the case of quantum access, we define the quantum version of the signing oracle by $U_{\Sigma',sk}$, which has the action

$$U_{\Sigma',sk} |m, a, b, c, d\rangle = |m, a \oplus \sigma, b \oplus u, c \oplus v, d \oplus w\rangle$$

where $\sigma = \Sigma.\mathsf{Sign}(sk, m)$, $(u, v) = \mathcal{B}_{s,t}(m.x, m.y)$, and $w = sk \cdot \delta_{s,m.z}$. Note that $U_{\Sigma',sk}$ is its own inverse.

Lemma 1. *Suppose that, with classical queries, at least $c_{\mathcal{B}}$ queries to $\mathcal{B}_{s,t}$ are required to determine s with probability $p_{\mathcal{B}}$, and that it takes at least time c_Σ for an adversary to win the* $\mathsf{Q^cQ\text{-}eufcma}$ *security game for Σ with probability at least p_Σ. If a (possibly quantum) adversary $\mathcal{A}$ with classical access to a $\Sigma'.\mathsf{Sign}$ oracle and vk runs for time $c < \min\{c_{\mathcal{B}}, c_\Sigma\}$, then $\mathcal{A}$ wins the* $\mathsf{Q^cQ\text{-}eufcma}$ *security game for Σ' with probability at most $p \leq p_{\mathcal{B}} + p_\Sigma + 2^{-256}c$.*

Proof. First note that there is no relationship between Σ and s. In particular, having access to a Σ signing oracle and vk, does not reduce the complexity of determining s from $\mathcal{B}_{s,t}$. Likewise, having oracle access to $\mathcal{B}_{s,t}$ does not make it easier to forge signatures for Σ. Otherwise, an adversary could just randomly choose s and t and simulate $\mathcal{B}_{s,t}$. The only way that they are related is via the third input and output strings on $\Sigma'.\mathsf{Sign}$, and these do not help unless $\mathcal{A}$ inputs an m with $m.z = s$ to the signing oracle, in which case $\mathcal{A}$ learns sk directly. This could only happen if $\mathcal{A}$ solves the $\mathcal{B}_{s,t}$ oracle problem, or guesses s directly.

Now since $c < c_{\mathcal{B}}$, $\mathcal{A}$ makes fewer than $c_{\mathcal{B}}$ queries and hence learns s via the oracle problem with probability at most $p_{\mathcal{B}}$. The probability of learning s by guessing is no more than $2^{-256}c$. Thus the signing oracle returns something of the form $(\cdot,\cdot,\cdot,sk)$ with probability no more than $p_{\mathcal{B}} + 2^{-256}c$.

Meanwhile, since $c < c_\Sigma$, the probability of $\mathcal{A}$ successfully forging enough Σ signatures to win the $\mathsf{Q^cQ\text{-}eufcma}$ security game for Σ is at most p_Σ unless one of the previous cases applies. Now every valid signature for Σ' contains a valid signature for Σ. Hence the total probability of forging enough Σ' signatures to win the $\mathsf{Q^cQ\text{-}eufcma}$ security game for Σ' is at most $p \leq p_{\mathcal{B}} + p_\Sigma + 2^{-256}c$. ☐

Lemma 2. *Suppose $\Sigma.\mathsf{Sign}$ is deterministic. If, given quantum query access to $\mathcal{B}_{s,t}$ it is possible to recover s with 1 query, then 3 quantum queries to $U_{\Sigma',sk}$ suffice to efficiently generate any polynomial number of valid signatures for Σ'.*

Proof. An adversary can, given a quantum oracle for $U_{\Sigma',sk}$, construct a quantum oracle for $\mathcal{B}_{s,t}(x,y)$. This is done using a standard technique called uncomputing, which uses two calls to $U_{\Sigma',sk}$. The circuit is shown in Figure 4. In the circuit we can see that the output contains two registers which are in the state $|u,v\rangle$. We also have four additional registers which remain in the state $|0\rangle$ and hence are unentangled with the output $|u,v\rangle$. This circuit works for superpositions of inputs as well.

Using 1 call to $\mathcal{B}_{s,t}$ the adversary can recover s. One more query to $U_{\Sigma',sk}$ with any message m such that $m.x = 0$, $u.y = 0$ and $m.z = s$ returns $|\cdot, u, v, sk\rangle$ where $(u,v) = \mathcal{B}_{s,t}(0,0)$. The adversary then has enough information to efficiently compute $\Sigma.\mathsf{Sign}$, $\mathcal{B}_{s,t}(0,0)$ and $\delta_{m.z,s}$ and hence can efficiently compute $\Sigma'.\mathsf{Sign}(m)$ for any m such that $m.x = m.y = 0$. ☐

We now turn our attention to providing a suitable $\mathcal{B}_{s,t}$. We will modify an existing oracle problem from the literature:

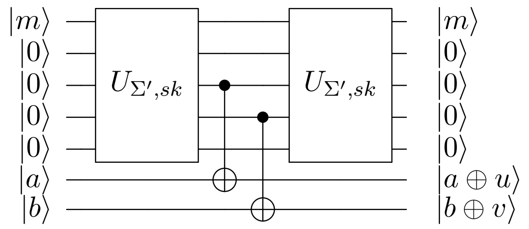

Figure 4: Quantum circuit to provide oracle access to $\mathcal{B}_{s,t}$ using two calls to $U_{\Sigma',sk}$, with $(u,v) = \mathcal{B}_{s,t}(m.x, m.y)$.

Definition 2 ([13])**.** The *hidden linear structure* problem is as follows: given oracle access to $\mathcal{B}_{s,\pi}(x,y) = (x, \pi(y \oplus sx))$, where $x, y, s \in GF(2^n)$ and $\pi \in Perm(\{0,1\}^n)$ with s and π chosen uniformly at random, determine s. (Here, $Perm(S)$ denotes the set of all permutations on a set S.)

The hidden linear structure problem is well suited to our purposes since it has a constant query complexity with quantum oracle access, and exponential query complexity with classical oracle access, as summarized by the following theorem, which we can obtain by looking through the proof of Theorem 3 in [13] and applying some inequalities.

Theorem 5 ([13])**.** *The hidden linear structure problem has query complexity $\Omega(2^{n/2})$ for classical queries, and 1 for quantum queries. More specifically, there exists a quantum algorithm which solves the hidden linear structure problem with 1 query and probability 1, while any algorithm which queries the oracle classically and uses 2^b queries with $2b \leq n-2$ outputs the correct s with probability at most 2^{2b-n+1}.*

Unfortunately, it takes an exponential amount of space to describe $\mathcal{B}$, and we need a description which is at most polynomially sized to add to the public key of Σ'. We thus define a restricted version of the hidden linear structure problem which replaces π with a pseudorandom permutation. To this end, suppose that $\mathcal{P} = \{\pi_t : t \in \{0,1\}^k\}$ is a family of pseudorandom permutations on $\{0,1\}^{|sk|}$ which is secure under the following definition:

Definition 3. We say a set $\mathcal{P} = \left\{\pi_t : t \in \{0,1\}^k\right\}$ of pseudorandom permutations on the set $\{0,1\}^l$ is $(c_\mathcal{P}, p_\mathcal{P})$-quantum-indistinguishable if no quantum algorithm with running time less than $c_\mathcal{P}$ can win the following distinguishability game with probability more than $1/2 + p_\mathcal{P}$ (i.e. with advantage $p_\mathcal{P}$) when using $c_\mathcal{P}$ quantum oracle queries:

1. $a \leftarrow_\$ \{0,1\}$
2. If $a = 0$, then $\pi \leftarrow_\$ \mathcal{P}$. Else $\pi \leftarrow_\$ Perm(\{0,1\}^l)$.
3. $a' \leftarrow_\$ \mathcal{A}^{\pi(\cdot)}$
4. $\mathcal{A}$ wins if $a' = a$

Definition 4. The *quantum-safe hidden linear structure* problem is a hidden linear structure problem where π is drawn from a set $\mathcal{P}$ of quantum-indistinguishable pseudorandom permutations.

Lemma 3. *Assuming there exists a (d, δ)-quantum indistinguishable family of pseudorandom permutations $\mathcal{P} = \{\pi_t : t \in \{0,1\}^k\}$ on $\{0,1\}^n$, then the quantum-safe hidden linear structure problem is indistinguishable from the hidden linear structure problem in time d with advantage greater than δ in the following distinguishability game:*

1. $a \leftarrow_\$ \{0,1\}$
2. *If $a = 0$, then $\pi \leftarrow_\$ P$. Else $\pi \leftarrow_\$ Perm(\{0,1\}^l)$.*
3. $s \leftarrow_\$ \{0,1\}^n$
4. $a' \leftarrow_\$ \mathcal{A}^{B_{s,\pi}(\cdot)}$
5. *$\mathcal{A}$ wins if $a' = a$*

Proof. Suppose that there exists an efficient quantum algorithm $\mathcal{A}$ which uses time d and distinguishes between the two oracle distributions with advantage δ' in a standard distinguishability game. Straightforwardly we can construct an algorithm $\mathcal{A}'$ which distinguishes between $\mathcal{P}$ and the full set of permutations. $\mathcal{A}'$ receives oracle access to unknown permutation π, and then chooses $s \leftarrow_\$ \{0,1\}^n$. $\mathcal{A}'$ can then simulate the $B_{s,\pi}$ oracle for $\mathcal{A}$ and outputs whatever $\mathcal{A}$ outputs. The result will be correct with probability at least $1/2 + \delta'$. Given that $\mathcal{P}$ is (d, δ)-quantum indistinguishable, we learn that $\delta' < \delta$. □

We are now in a position to prove Theorem 4.

Proof of Theorem 4. We use Σ' as defined earlier, with $\mathcal{B}_{s,t}$ being the oracle for a quantum safe hidden linear structure problem, which exists by the existence of $\mathcal{P}$. By Lemma 2, Σ' is not $\mathsf{Q^qQ}$-eufcma-secure since a quantum adversary allowed quantum oracle access to $\Sigma'.\mathsf{Sign}$ can efficiently generate a polynomial number of signatures using a constant number of oracle queries.

Now suppose we have a quantum adversary $\mathcal{A}$ which has classical oracle access to $\Sigma'.\mathsf{Sign}$ and runs in time $2^b < \max\{2^{n/2-2}, c_\Sigma\}$. By Lemma 3 and Theorem 5, $\mathcal{A}$ obtains s through classical oracle access to $\mathcal{B}$ with probability at most $2^{2b-n+1} + p_\mathcal{P}$. Then we can set $p_\mathcal{B} = 2^{2b-n+1} + p_\mathcal{P}$ and apply Lemma 1 to find that $\mathcal{A}$ breaks unforgeability of Σ' with probability at most

$$p_\Sigma + 2^{2b-n+1} + \delta + 2^{b-256} \ .$$

If $\mathcal{A}$ runs in polynomial time, then $b \in O(\log(\mathrm{poly}(n))$ and hence Σ' is $\mathsf{Q^cQ}$-eufcma-secure. □

3 Separability of hybrid signatures

In Section 4, we will investigate combiners for constructing one signature scheme from two. Before looking at specific combiners, an interesting security property arises in general for combined signature schemes: is it possible for a signature in a combined signature scheme to be separated out into valid signatures for either of its individual component schemes?

Let C be a combiner, and let Σ_1, Σ_2 be signature schemes. Let $\Sigma' = C(\Sigma_1, \Sigma_2)$. The initial idea for the security notion for non-separability is based on the standard EUF-CMA experiment: given a signing oracle that produces signatures for the combined scheme Σ', it should be hard for an adversary to produce a valid signature for Σ_1 ("1-non-separability") or Σ_2 ("2-non-separability").

However, this approach needs some refinement. In all of the combiners we consider in Section 4, the combined signature contains subcomponents which are valid in the underlying schemes. This makes it impossible to satisfy the naive version of non-separability. For example, suppose we have a signer issuing signatures from a combined signature scheme. Suppose further we have a verifier for one of the underlying signature schemes *who is also aware of the combined signature scheme.* Given signatures from the combined scheme, it should be hard to make a signature that is valid in one of the underlying signature schemes *and which is not recognized as being from the combined signature scheme.* In sum: separability is about downgrading.

Figure 5 shows the τ-non-separability security experiment, $\mathsf{X^yZ}$-τ-nonsep, with $\tau \in \{1, 2\}$ for signature scheme Σ_1 or Σ_2. It checks the ability of a two-stage $\mathsf{X^yZ}$-adversary $(\mathcal{A}_1, \mathcal{A}_2)$ to create a valid Σ_τ signature; the adversary's pair has to "fool" the recognizer algorithm $\Sigma.R$ which is supposed to recognize values used in a combined signature scheme Σ.

Formally, a recognizer related to a combined signature scheme $\Sigma = C(\Sigma_1, \Sigma_2)$ is a function $\Sigma.R$ that takes one input and outputs a single bit. For a signature scheme Σ_τ with message space $\mathcal{M}_{\Sigma_\tau}$, it may be that $\Sigma.R$ yields 1 on some elements of $\mathcal{M}_{\Sigma_\tau}$ and 0 on others: the purpose of R is to recognize whether certain inputs are associated with a *second* signature scheme.

Because the $\mathsf{X^yZ}$-τ-nonsep experiment is parameterized by the recognizer algorithm R, each R gives rise to a different security notion. If R is vacuous, it can lead to a useless security notion. We can make statements of the form "If $\langle$*some assumption on* Σ_1 *and* $\Sigma_2\rangle$ and R is the algorithm ..., then $C(\Sigma_1, \Sigma_2)$ is $\mathsf{X^yZ}$-1-non-separable with respect to recognizer algorithm R." However, a statement of the form "C is not 1-non-separable" is more difficult: one must quantify over all (or some class of) recognizer algorithms R. We will say informally that "C is not τ-non-separable" if the way that Σ_τ is used in C is to effectively sign the message without any modification, since the only recognizer that would recognize all signatures from C would cover the entire message space of Σ_τ.

```
$\mathrm{Expt}^{\mathsf{X}^{\mathsf{y}}\mathsf{Z}\text{-}\tau\text{-}\mathsf{nonsep}}_{C,\Sigma_1,\Sigma_2,C(\Sigma_1,\Sigma_2).R}(\mathcal{A}_1,\mathcal{A}_2)$:
1  $q_S \leftarrow 0$
2  $(sk', vk') \leftarrow\!\$\; C(\Sigma_1,\Sigma_2).\mathsf{KeyGen}()$
3  $st \leftarrow\!\$\; \mathcal{A}_1^{\mathcal{O}_S(\cdot)}(vk')$
4  $(m^*, \sigma^*) \leftarrow\!\$\; \mathcal{A}_2(st)$
5  If $(\Sigma_\tau.\mathsf{Verify}((vk')_\tau, m^*, \sigma^*) = 1)$
   $\wedge\ (C(\Sigma_1,\Sigma_2).R(m^*) = 0)$
6    Return 1
7  Else return 0

$\mathcal{O}_S$:
8  If $\mathsf{y} = \mathsf{c}$, use classical signing oracle $\mathcal{O}_S$ from Figure 2 for $C(\Sigma_1,\Sigma_2)$.
9  If $\mathsf{y} = \mathsf{q}$, use quantum signing oracle $\mathcal{O}_S$ from Figure 2 for $C(\Sigma_1,\Sigma_2)$.
```

Figure 5: Unified security experiment for $\mathsf{X}^{\mathsf{y}}\mathsf{Z}\text{-}\tau\text{-}\mathsf{nonsep}$: τ-non-separability of a combiner C with signature schemes Σ_1, Σ_2 with respect to a recognizer $C(\Sigma_1,\Sigma_2).R$ for a two-stage adversary $\mathcal{A}_1$ (of type X), $\mathcal{A}_2$ (of type Z) with signing oracle of type y. $(vk')_\tau$ denotes the projection (extraction) of the public key associated with scheme Σ_τ from the combined scheme's public key vk', which we assume is possible.

One can view a recognizer algorithm as a generalization of the long-standing technique of "domain separation". Starting from scratch, it might be preferable to build combiners that explicitly include domain separation in their construction, thereby eliminating the need for recognizer algorithms. Since our combiners are motivated by existing protocol constraints, we do not have that luxury.

4 Combiners

We now examine several methods of using two signature schemes Σ_1 and Σ_2 to produce hybrid signatures. For all our combiners, the key generation of the combined scheme will simply be the concatenation of the two schemes' keys:

- $C(\Sigma_1,\Sigma_2).\mathsf{KeyGen}()$: $(sk_1, vk_1) \leftarrow\!\$\; \Sigma_1.\mathsf{KeyGen}()$, $(sk_2, vk_2) \leftarrow\!\$\; \Sigma_2.\mathsf{KeyGen}()$. Return $(sk' \leftarrow (sk_1, sk_2), vk' \leftarrow (vk_1, vk_2))$.

The verification algorithm in each case is defined in the natural way.

4.1 $C_{\|}$: Concatenation

This "combiner" is the trivial combiner, which just places independent signatures from the two schemes side-by-side:

- $C_{\|}(\Sigma_1,\Sigma_2).\mathsf{Sign}(sk', m)$: $\sigma_1 \leftarrow\!\$\; \Sigma_1.\mathsf{Sign}(sk_1, m)$, $\sigma_2 \leftarrow\!\$\; \Sigma_2.\mathsf{Sign}(sk_2, m)$. Return $\sigma' \leftarrow \sigma_1 \| \sigma_2$.

Theorem 6 (Unforgeability of $C_{\|}$). *If either Σ_1 or Σ_2 is unforgeable in the classical (or quantum) random oracle model, then $\Sigma' = C_{\|}(\Sigma_1,\Sigma_2)$ is unforgeable in the classical (or quantum, respectively) random oracle model. More precisely, if Σ_1 is $\mathsf{X}^{\mathsf{y}}\mathsf{Z}$-$\mathsf{eufcma}$-secure or Σ_2 is $\mathsf{U}^{\mathsf{v}}\mathsf{W}$-$\mathsf{eufcma}$-secure, then $\Sigma' = C_{\|}(\Sigma_1,\Sigma_2)$ is $\max\{\mathsf{X}^{\mathsf{y}}\mathsf{Z}, \mathsf{U}^{\mathsf{v}}\mathsf{W}\}$-$\mathsf{eufcma}$-secure.*

Proof. Suppose $\mathcal{A}$ is an $\mathsf{R}^{\mathsf{s}}\mathsf{T}$-adversary that finds a forgery in $\Sigma' = C_{\|}(\Sigma_1,\Sigma_2)$ — in other words, it outputs $q_S + 1$ valid signatures under Σ' on distinct messages. We can construct an $\mathsf{R}^{\mathsf{s}}\mathsf{T}$ algorithm $\mathcal{B}_1$ that finds a forgery in Σ_1. $\mathcal{B}_1$ interacts with an $\mathsf{R}^{\mathsf{s}}\mathsf{T}$ challenger for Σ_1 which provides a public key vk_1. $\mathcal{B}_1$ generates a key pair $(sk_2, vk_2) \leftarrow\!\$\; \Sigma_2.\mathsf{KeyGen}()$ and sets the public key for Σ' to be (vk_1, vk_2). When $\mathcal{A}$ asks for $\sum_{m,t,z} \psi_{m,t,z} |m,t,z\rangle)$ to be signed using Σ', we treat t as consisting of two registers $t_1 \| t_2$, $\mathcal{B}_1$ proceeds by passing the m, t_1, and z registers to its signing oracle for Σ_1, then runs the quantum signing operation from Figure 2 for $\Sigma_2.\mathsf{Sign}$ on the m, t_2, and z registers.

There is a one-to-one correspondence between $\mathcal{A}$'s queries to its signing oracle and $\mathcal{B}_1$'s queries to its signing oracle.

If Σ_1 is proven to be secure in the random oracle (rather than standard) model, then this proof of $C_{\|}(\Sigma_1, \Sigma_2)$ also proceeds in the random oracle model: $\mathcal{B}_1$ relays $\mathcal{A}$'s hash oracle queries directly to its oracle, giving a one-to-one correspondence between $\mathcal{A}$'s queries to its hash oracle and $\mathcal{B}_1$'s queries to its hash oracle. This holds in either the classical or quantum random oracle model.

If $\mathcal{A}$ wins the $\mathsf{R^sT\text{-}eufcma}$ game, then it has returned $q_S + 1$ valid signatures $\sigma'_i = (\sigma'_{i,1}, \sigma'_{i,2})$ on distinct messages m_i such that $\Sigma_1.\mathsf{Verify}(vk_1, m_i, \sigma'_{i,1}) = 1$ and $\Sigma_2.\mathsf{Verify}(vk_2, m_i, \sigma'_{i,2}) = 1$. $\mathcal{B}_1$ can extract from this $q_S + 1$ valid signatures under Σ_1 on distinct messages. Thus, $\mathrm{Adv}^{\mathsf{R^sT\text{-}eufcma}}_{\Sigma'}(\mathcal{A}) \leq \mathrm{Adv}^{\mathsf{R^sT\text{-}eufcma}}_{\Sigma_1}(\mathcal{B}_1)$. Similarly it holds for Σ_2: $\mathrm{Adv}^{\mathsf{R^sT\text{-}eufcma}}_{\Sigma'}(\mathcal{A}) \leq \mathrm{Adv}^{\mathsf{R^sT\text{-}eufcma}}_{\Sigma_2}(\mathcal{B}_2)$.

It follows that

$$\mathrm{Adv}^{\mathsf{R^sT\text{-}eufcma}}_{\Sigma'}(\mathcal{A}) \leq \min\{\mathrm{Adv}^{\mathsf{R^sT\text{-}eufcma}}_{\Sigma_1}(\mathcal{B}_1), \mathrm{Adv}^{\mathsf{R^sT\text{-}eufcma}}_{\Sigma_2}(\mathcal{B}_2)\} \ .$$

Thus, if either $\mathrm{Adv}^{\mathsf{R^sT\text{-}eufcma}}_{\Sigma_1}(\mathcal{B}_1)$ or $\mathrm{Adv}^{\mathsf{R^sT\text{-}eufcma}}_{\Sigma_2}(\mathcal{B}_2)$ is small, then so too is $\mathrm{Adv}^{\mathsf{R^sT\text{-}eufcma}}_{\Sigma'}(\mathcal{A})$. □

Clearly, $C_{\|}$ is neither 1-non-separable nor 2-non-separable: σ_1 is immediately a Σ_1-signature for m, with no way of recognizing this as being different from typical Σ_1 signatures. Similarly for σ_2.

Another natural nesting combiner which we call weak nesting ($C_{\text{weak-nest}}$: $\sigma_1 \leftarrow\!\!\$\ \mathsf{Sign}_1(m); \sigma_2 \leftarrow\!\!\$\ \mathsf{Sign}_2(\sigma_1)$; $\sigma \leftarrow (\sigma_1, \sigma_2)$) appears in the appendix, but is not used in any of our applications in Section 5 so we omit it here.)

4.2 $C_{\text{str-nest}}$: Strong nesting

For this combiner, the second signature scheme signs both the message and the signature from the first signature scheme:

- $C_{\text{str-nest}}(\Sigma_1, \Sigma_2).\mathsf{Sign}(sk', m)$: $\sigma_1 \leftarrow\!\!\$\ \Sigma_1.\mathsf{Sign}(sk_1, m)$, $\sigma_2 \leftarrow\!\!\$\ \Sigma_2.\mathsf{Sign}(sk_2, (m, \sigma_1))$. Return $\sigma' \leftarrow (\sigma_1, \sigma_2)$.

$C_{\text{str-nest}}$ retains $C_{\text{weak-nest}}$'s 2-non-separability while adding unforgeability, and retains $C_{\|}$'s unforgeability while adding 2-non-separability.

Theorem 7 (Unforgeability of $C_{\text{str-nest}}$)**.** *If either Σ_1 or Σ_2 is unforgeable in the classical (or quantum) random oracle model, then $\Sigma' = C_{\text{str-nest}}(\Sigma_1, \Sigma_2)$ is unforgeable in the classical (or quantum, respectively) random oracle model. More precisely, if Σ_1 is $\mathsf{X^yZ\text{-}eufcma}$-secure or Σ_2 is $\mathsf{U^vW\text{-}eufcma}$-secure, then $\Sigma' = C_{\text{str-nest}}(\Sigma_1, \Sigma_2)$ is $\max\{\mathsf{X^yZ}, \mathsf{U^vW}\}\text{-}\mathsf{eufcma}$-secure.*

Proof. This proof follows the same approach as the proof of unforgeability for $C_{\|}$ (Theorem 6). Suppose $\mathcal{A}$ is an $\mathsf{R^sT}$-adversary that finds a forgery in $\Sigma' = C_{\text{str-nest}}$ — in other words, it outputs $q_S + 1$ valid signatures under Σ' on distinct messages. We can construct an $\mathsf{R^sT}$ algorithm $\mathcal{B}_1$ that finds a forgery in Σ_1 and an $\mathsf{R^sT}$ algorithm $\mathcal{B}_2$ that finds a forgery in Σ_2.

$\mathcal{B}_1$ interacts with an $\mathsf{R^sT}$ challenger for Σ_1 which provides a public key vk_1. $\mathcal{B}_1$ generates a key pair $(sk_2, vk_2) \leftarrow\!\!\$\ \Sigma_2.\mathsf{KeyGen}()$ and sets the public key for Σ' to be (vk_1, vk_2). When $\mathcal{A}$ asks for $\sum_{m,t,z} \psi_{m,t,z} |m, t, z\rangle)$ to be signed using Σ', we treat t as consisting of two registers $t_1 \| t_2$, $\mathcal{B}_1$ proceeds by passing the m, t_1, and z registers to its signing oracle for Σ_1, then runs the quantum signing operation from Figure 2 for $\Sigma_2.\mathsf{Sign}$ on the m, t_2, and z registers. There is a one-to-one correspondence between $\mathcal{A}$'s queries to its oracle and $\mathcal{B}_1$'s queries to its oracle. As before in the proof of Theorem 6, if Σ_1 is proven to be secure in the random oracle model, then this proof of $C_{\text{weak-nest}}(\Sigma_1, \Sigma_2)$ also proceeds in the random oracle model: $\mathcal{B}_1$ relays $\mathcal{A}$'s hash oracle queries directly to its hash oracle, giving a one-to-one correspondence between $\mathcal{A}$'s queries to its (classical or quantum) hash oracle and $\mathcal{B}_1$'s queries to its (classical or quantum, respectively) hash oracle.

If $\mathcal{A}$ wins the $\mathsf{R}^{\mathsf{s}}\mathsf{T}$-$\mathsf{eufcma}$ game, then it has returned q_S+1 valid signatures $\sigma'_i = (\sigma'_{i,1}, \sigma'_{i,2})$ on distinct messages m_i such that $\Sigma_1.\mathsf{Verify}(vk_1, m_i, \sigma'_{i,1}) = 1$ and $\Sigma_2.\mathsf{Verify}(vk_2, (m_i, \sigma'_{i,1}), \sigma'_{i,2}) = 1$. $\mathcal{B}_1$ can extract from this q_S+1 valid signatures under Σ_1 on distinct messages. Thus, $\mathrm{Adv}_{\Sigma'}^{\mathsf{R^sT\text{-}eufcma}}(\mathcal{A}) \leq \mathrm{Adv}_{\Sigma_1}^{\mathsf{R^sT\text{-}eufcma}}(\mathcal{B}_1)$. Similarly it holds for Σ_2: $\mathrm{Adv}_{\Sigma'}^{\mathsf{R^sT\text{-}eufcma}}(\mathcal{A}) \leq \mathrm{Adv}_{\Sigma_2}^{\mathsf{R^sT\text{-}eufcma}}(\mathcal{B}_2)$. It follows that

$$\mathrm{Adv}_{\Sigma'}^{\mathsf{R^sT\text{-}eufcma}}(\mathcal{A}) \leq \min\{\mathrm{Adv}_{\Sigma_1}^{\mathsf{R^sT\text{-}eufcma}}(\mathcal{B}_1), \mathrm{Adv}_{\Sigma_2}^{\mathsf{R^sT\text{-}eufcma}}(\mathcal{B}_2)\} \ .$$

Thus, if either $\mathrm{Adv}_{\Sigma_1}^{\mathsf{R^sT\text{-}eufcma}}(\mathcal{B}_1)$ or $\mathrm{Adv}_{\Sigma_2}^{\mathsf{R^sT\text{-}eufcma}}(\mathcal{B}_2)$ is small, so too is $\mathrm{Adv}_{\Sigma'}^{\mathsf{R^sT\text{-}eufcma}}(\mathcal{A})$. □

$C_{\text{str-nest}}$ is not 1-non-separable: σ_1 is immediately a Σ_1-signature for m, with no way of recognizing this as being different from typical Σ_1 signatures. However, since the inputs to Σ_2 in $C_{\text{str-nest}}$ have a particular form, we can recognize those and achieve 2-non-separability:

Theorem 8 (2-non-separability of $C_{\text{str-nest}}$)**.** *If Σ_2 is $\mathsf{X^yZ}$-eufcma-secure, then $\Sigma' = C_{\text{str-nest}}(\Sigma_1, \Sigma_2)$ is $\mathsf{X^cZ}$-$\mathsf{2}$-nonsep with recognizer $R(m) = (m \in \{0,1\}^* \times \mathcal{S}_{\Sigma_1})$.*

Proof. We can construct a reduction $\mathcal{B}_2$ which is an $\mathsf{X^cZ}$-eufcma adversary for Σ_2. $\mathcal{B}_2$ generates a keypair (vk_1, sk_1) for Σ_1, and interacts with an $\mathsf{X^cZ}$-eufcma challenge for Σ_2. When $\mathcal{A}$ classically queries its signing oracle to obtain a signature under Σ' of m_i, $\mathcal{B}_2$ signs m_i with Σ_1 to obtain $\sigma_{i,1}$. Afterwards, $\mathcal{B}_2$ passes $(m, \sigma_{i,1})$ to its Σ_2 signing oracle and returns the resulting $\sigma_{i,2}$ to $\mathcal{A}$. Eventually, $\mathcal{A}$ returns (μ^*, σ^*) such that $\Sigma_2.\mathsf{Verify}(vk_2, \mu^*, \sigma^*) = 1$ but $\Sigma'.R(\mu^*) = 0$, i.e., $\mu^* \notin \{0,1\}^* \times \mathcal{S}_{\Sigma_1}$. This means in particular that $\mu^* \neq (m_i, \sigma_{1,i})$ for all i. Moreover, all the $(m_i, \sigma_{1,i})$ are distinct, since all m_i are distinct. This means we have q_S+1 valid message-signature pairs under Σ_2, yielding a successful forgery for the $\mathsf{X^cZ}$-eufcma experiment for Σ_2. Thus,

$$\Pr[S_1] \leq \mathrm{Adv}_{\Sigma_2}^{\mathsf{X^cZ\text{-}eufcma}}(\mathcal{B}_2) \ .$$

□

4.3 D_{nest}: Dual message combiner using nesting

Some of our applications in Section 5 require a combiner for two (possibly related) messages signed with two signature schemes. For example, in our X.509 certificates application, we generate one certificate signed with Σ_1, then embed that certificates as an extension inside a second certificate signed with Σ_2.

- $D_{\text{nest}}(\Sigma_1, \Sigma_2).\mathsf{Sign}(sk', (m_1, m_2))$: $\sigma_1 \leftarrow\$ \Sigma_1.\mathsf{Sign}(sk_1, m_1)$, $\sigma_2 \leftarrow\$ \Sigma_2.\mathsf{Sign}(sk_2, (m_1, \sigma_1, m_2))$. Return $\sigma' \leftarrow (\sigma_1, \sigma_2)$.

This dual-message combiner is not designed to give unforgeability of *both* messages under *either* signature scheme, though it does preserve unforgeability of each message under its corresponding signature scheme, as well as give unforgeability of both messages under the outer signature scheme Σ_2.

Theorem 9 (Unforgeability of D_{nest})**.** *If either Σ_1 or Σ_2 is unforgeable in the classical (or quantum) random oracle model, then $D_{\text{nest}}(\Sigma_1, \Sigma_2)$ is unforgeable (in a certain sense) in the classical (or quantum, respectively) random oracle model. More precisely, if Σ_1 is $\mathsf{X^yZ}$-eufcma-secure, then the combined scheme $D_{\text{nest}}(\Sigma_1, \Sigma_2)$ is $\mathsf{X^yZ}$-eufcma-secure with respect to its first message component only. If Σ_2 is $\mathsf{U^vW}$-eufcma-secure, then $D_{\text{nest}}(\Sigma_1, \Sigma_2)$ is $\mathsf{U^vW}$-eufcma-secure.*

Proof. This theorem contains two statements. The first statement is: If Σ_1 is $\mathsf{X^yZ}$-eufcma-secure, then $D_{\text{nest}}(\Sigma_1, \Sigma_2)$ is $\mathsf{X^yZ}$-eufcma-secure with respect to its first message component only. $D_{\text{nest}}(\Sigma_1, \Sigma_2)$, when restricted to its first message component only, is just Σ_1, so the first statement follows vacuously.

Now consider the second statement: $D_{\text{nest}}(\Sigma_1, \Sigma_2)$ is $\mathsf{U^vW}$-eufcma-secure if Σ_2 is $\mathsf{U^vW}$-eufcma-secure. Suppose $\mathcal{A}$ is a $\mathsf{U^vW}$ algorithm that outputs a forgery for $\Sigma' = D_{\text{nest}}(\Sigma_1, \Sigma_2)$ — in other words, it outputs $q_S + 1$ valid signatures under Σ' on distinct messages. We can construct an $\mathsf{U^vW}$ algorithm $\mathcal{B}_2$ that finds a forgery in Σ_2. $\mathcal{B}_2$ interacts with an $\mathsf{U^vW}$ challenger for Σ_2 which provides a public key vk_2. $\mathcal{B}_2$ generates a key pair $(sk_1, vk_1) \leftarrow\$ \Sigma_1.\mathsf{KeyGen}()$ and sets the public key for Σ' to be (vk_1, vk_2). When $\mathcal{A}$ asks for $\sum_{m,t,z} \psi_{m,t,z} |m, t, z\rangle)$ to be signed using Σ', we treat t as consisting of two registers $t_1 \| t_2$, $\mathcal{B}_2$ proceeds by passing the m, t_2, and z registers to its signing oracle for Σ_2, then runs the quantum signing operation from Figure 2 for $\Sigma_1.\mathsf{Sign}$ on the m, t_1, and z registers. There is a one-to-one correspondence between $\mathcal{A}$'s queries to its oracle and $\mathcal{B}_2$'s queries to its oracle. As before in the proof of Theorem 6, if Σ_1 is proven to be secure in the random oracle model, then this proof of $C_{\text{weak-nest}}(\Sigma_1, \Sigma_2)$ also proceeds in the random oracle model: $\mathcal{B}_2$ relays $\mathcal{A}$'s hash oracle queries directly to its hash oracle, giving a one-to-one correspondence between $\mathcal{A}$'s queries to its (classical or quantum) hash oracle and $\mathcal{B}_2$'s queries to its (classical or quantum, respectively) hash oracle.

If $\mathcal{A}$ wins the $\mathsf{U^vW}$-eufcma game, then it has returned $q_S + 1$ distinct tuples $(m_{1,i}, m_{2,i}, \sigma_{1,i}, \sigma_{2,i})$ such that

$$\Sigma_1.\mathsf{Verify}(vk_1, m_{1,i}, \sigma_{1,i}) = 1$$

and

$$\Sigma_2.\mathsf{Verify}(vk_2, (m_{1,i}, \sigma_{1,i}, m_{2,i}), \sigma_{2,i}) = 1 \ .$$

Hence, $\mathcal{B}_2$ can extract $q_S + 1$ valid signatures under Σ_2 and thus it holds that $\mathrm{Adv}^{\mathsf{R^sT\text{-}eufcma}}_{\Sigma'}(\mathcal{A}) \leq \mathrm{Adv}^{\mathsf{R^sT\text{-}eufcma}}_{\Sigma_2}(\mathcal{B}_2)$. □

D_{nest} is not 1-non-separable: σ_1 is immediately a Σ_1-signature for m, with no way of recognizing this as being different from typical Σ_1 signatures. However, since the inputs to Σ_2 in D_{nest} have a particular form, we can recognize those and achieve 2-non-separability:

Theorem 10 (2-non-separability of D_{nest})**.** *If Σ_2 is $\mathsf{X^yZ}$-eufcma-secure, then $\Sigma' = D_{\text{nest}}(\Sigma_1, \Sigma_2)$ is $\mathsf{X^cZ}$-2-nonsep with respect to recognizer algorithm $R(m) = (m \in \{0,1\}^* \times \mathcal{S}_{\Sigma_1} \times \{0,1\}^*)$.*

Proof. This proof follows the same approach as the proof of 2-non-separability for $C_{\text{str-nest}}$ (Theorem 8).

We can construct a reduction $\mathcal{B}_2$ which is an $\mathsf{X^cZ}$-eufcma adversary for Σ_2. $\mathcal{B}_2$ generates a keypair (vk_1, sk_1) for Σ_1, and interacts with an $\mathsf{X^cZ}$-eufcma challenge for Σ_2. When $\mathcal{A}$ classically queries its signing oracle to obtain a signature under Σ' of $(m_{i,1}, m_{i,2})$, $\mathcal{B}_2$ signs $m_{i,1}$ with Σ_1 to obtain $\sigma_{i,1}$. Afterwards, $\mathcal{B}$ passes $(m_{i,1}, \sigma_{i,1}, m_{i,2})$ to its Σ_2 signing oracle and returns the resulting $\sigma_{i,2}$ to $\mathcal{A}$. Eventually, $\mathcal{A}$ returns (μ^*, σ^*) such that $\Sigma_2.\mathsf{Verify}(vk_2, \mu^*, \sigma^*) = 1$ but $\Sigma'.R(\mu^*) = 0$, i.e., $\mu^* \notin \{0,1\}^* \times \mathcal{S}_{\Sigma_1} \times \{0,1\}^*$. This means in particular that $\mu^* \neq (m_{i,1}, \sigma_{1,i}, m_{i,2})$ for all i. Moreover, all the $(m_{i,1}, \sigma_{1,i}, m_{i,2})$ are distinct, since all $(m_{i,1}, m_{i,2}$ are distinct. This means we have $q_S + 1$ valid message-signature pairs under Σ_2, yielding a successful forgery for the $\mathsf{X^cZ}$-eufcma experiment for Σ_2. Thus,

$$\Pr[S_1] \leq \mathrm{Adv}^{\mathsf{X^cZ\text{-}eufcma}}_{\Sigma_2}(\mathcal{B}_2) \ .$$

□

4.4 Adding labels

We briefly remark that a simple way to achieve 1- or 2-non-separability (or both) for all our combiners is to add distinguished labels to the Σ_1 or Σ_2 signing operation (or both). For example, to achieve 1-non-separability in $C_{\|}$:

- $C_{\|}^{(1)}(\Sigma_1, \Sigma_2).\mathsf{Sign}(sk', m)$:
  $\sigma_1 \leftarrow_{\$} \Sigma_1.\mathsf{Sign}(sk_1, \texttt{label}\|m)$, $\sigma_2 \leftarrow_{\$} \Sigma_2.\mathsf{Sign}(sk_2, m)$. Return $\sigma' \leftarrow (\sigma_1, \sigma_2)$.
- $R_{\|}^{(1)}(vk_1, m, \sigma_1)$:
  Return $\Sigma_1.\mathsf{Verify}(vk_1, \texttt{label}\|m, \sigma_1)$.

One could then readily show that, if Σ_1 is $\mathsf{X^yZ}$-eufcma-secure, then $C_{\|}^{(1)}(\Sigma_1, \Sigma_2)$ is $\mathsf{X^yZ}$-$\mathsf{1}$-nonsep with respect to recognizer algorithm $R_{\|}^{(1)}$. We could analogously apply this approach for 1- and 2-non-separability in $C_{\|}$, $C_{\text{weak-nest}}$, and $C_{\text{str-nest}}$. We refrain from examining this in detail due the fact that adding labels impacts backwards compatibility, which is the goal of Section 5.

5 Hybrid signatures in standards

We now examine three standards which make significant use of digital signatures—X.509 for certificates, TLS for secure channels, and S/MIME for secure email—to identify how hybrid signatures might be used in PKI standards, and evaluate backwards-compatibility of various approaches with existing software. Source code for generating the hybrid certificates and messages we used for testing, as well as scripts for running the tests, are available online at `https://www.douglas.stebila.ca/code/pq-pki-tests/`.

5.1 X.509v3 certificates

The X.509 standard version 3 [12] specifies a widely used format for public key certificates, as well as mechanisms for managing and revoking certificates. In X.509-based public key infrastructures, there is a hierarchical arrangement of trusted third party *certificate authorities* (CAs) which issue certificates binding a subject's identity to its public key, signed using the CA's private key.

The structure of an X.509v3 certificate is as follows. The body of the certificate (called a `tbsCertificate`) contains the name of the certificate authority as well as information about the subject, including the distinguished name of the subject, the subject's public key (including an algorithm identifier), and optionally some extensions, each of which consists of an extension identifier, value, and a flag whether the extension is to be considered critical. (If a critical extension can not be recognized or processed, the system must reject it; a non-critical extension should be processed if it is recognized and may ignored if it is not.) The `tbsCertificate` is followed by CA's signature over the `tbsCertificate` signed using the CA's private key, along with an algorithm identifier.

Our goal will be to construct a hybrid certificate which somehow includes two public keys for the subject (e.g., one traditional and one post-quantum algorithm) and two CA signatures. Notably, the X.509v3 standard says that a certificate can contain exactly one `tbsCertificate`, which can contain exactly one subject public key, and all of this can be signed using exactly one CA signature. This makes creating backwards-compatible hybrid certificates challenging.

Approach 1: Dual certificates. The simplest approach is of course to create separate certificates: one for the traditional algorithm, and the other for the post-quantum algorithms. (This would be a dual-message analogue of the concatenation combiner $C_{\|}$ from Section 4.1.) This approach leaves the task of conveying the "hybrid" certificate (actually, two certificates) to the application, which

Scheme	**Size** (bytes)				**Security**
	public key	secret key	signature	certificate	(bits)
Lattice-based					
GLP [18]	1 536	256	1 186	3.0 KiB	100
Ring-TESLA-II [1]	3 328	1 920	1 488	5.1 KiB	128
TESLA#-I [3]	3 328	2 112	1 616	5.2 KiB	128
BLISS [15]	7 168	2 048	1 559	9.0 KiB	128
TESLA-416 [2]	1 331 200	1 011 744	1 280	1 332.8 KiB	128
Hash-based					
XMSS [9]	912	19	2 451	3.6 KiB	82
SPHINCS [5]	1,056	1,088	41,000	42.3 KiB	>128
Rainbow [11]	44 160	86 240	37	44.5 KiB	80

Table 2: Post-quantum signature schemes; keys and signature sizes, estimated certificate sizes, and claimed security level

will suffice in some settings (e.g., in S/MIME and some TLS settings, see the following sections), but is unsatisfactory in others. (This approach and the next both require assigning additional object identifiers (OIDs) for each post-quantum algorithms, but this can be easily done.)

Approach 2: Second certificate in extension. Since X.509v3 does not provide any direct way of putting two public keys or two signatures in the same certificate, one option is to use the standard's extension mechanism. Let c_1 be the certificate obtained by the CA signing `tbsCertificate` m_1 (containing subject public key vk_1) using signature scheme Σ_1. Construct certificate c_2 by the CA signing `tbsCertificate` m_2 (containing subject public key vk_2 as well as (an encoding of) c_1 as an extension in m_2) using signature scheme Σ_2. The extension containing c_1 would use a distinct extension identifier saying "this is an additional certificate" and would be marked as non-critical. This is an instantiation of the dual message nested combiner D_{nest} from Section 4.3. (Alternatively, the extension could contain a subset of fields, such as just the public key and CA's signature, rather than a whole certificate.)

By marking the "additional certificate" extension as non-critical, existing software (not aware of the hybrid structure) *should* ignore the unrecognized extension and continue validating the certificate and using it in applications without change. Is this really the case—is this approach backwards-compatible with old software, and do large public keys or signatures cause problems?

Experimental evaluation of approach 2. We constructed hybrid certificates following approach 2. The "outside" certificate c_2 contains a 2048-bit RSA public key, and is signed by a CA using 2048-bit RSA key. The extension for embedding c_1 in c_2 is identified by a distinct and previously unused algorithm identifier (OID), and is marked as non-critical. Because post-quantum public keys and signatures vary substantially in size, we use a range of extension sizes to simulate the expected size of an embedded certificate for various post-quantum signature algorithms; the extension sizes we use are across the columns of Table 3, derived from public key and signature sizes summarized in Table 2. For our purposes of evaluating backwards compatibility, it does not matter whether the extension actually *contains* a valid post-quantum certificate, just that it is the *size* of such a

	Extension size (and corresponding example signature scheme)				
	1.5 KiB (RSA)	3.5 KiB (GLP [18])	9.0 KiB (BLISS [15])	43.0 KiB (SPHINCS [5])	1333.0 KiB (TESLA-416 [2])
GnuTLS 3.5.11	✓	✓	✓	✓	✓
Java SE 1.8.0_131	✓	✓	✓	✓	✓
mbedTLS 2.4.2	✓	✓	✓	✓	✓
NSS 3.29.1	✓	✓	✓	✓	✓
OpenSSL 1.0.2k	✓	✓	✓	✓	✓

Table 3: Compatibility of hybrid X.509v3 certificates containing large extensions.

certificate. The hybrid certificates were created using a custom-written Java program using the BouncyCastle library.

Table 3 shows the results of using command-line certificate verification programs in various libraries; all libraries we tested were able to parse and verify X.509v3 certificates containing unrecognized extensions of all sizes we tested.

5.2 TLS

TLSv1.2 [14] is the currently standardized version and is widely deployed. Ciphersuites with digital signatures allow servers and (optionally) clients to authenticate each other by presenting their public key (usually in an X.509v3 certificate) and signing certain messages. While the parties can negotiate which signature algorithms to use, once having done so they can each use only a single public key and signature algorithm to authenticate.

The current draft of TLSv1.3 [22] does not change how server authentication works. However, it has a "post-handshake authentication" mode for clients [22, §4.5.2], where clients can be requested to (further) authenticate using a certificate for a given algorithm. This would allow client authentication using two (or more) signature schemes. This is an example of the concatenation combiner $C_{\|}$ from Section 4.1, since each client signature is over the same handshake context data structure. A proposal for "exported authenticators" [23] is currently before the TLS working group and would allow a similar approach for server authentication, although it envisions that this takes place out-of-band (e.g., at the application layer). Neither would require hybrid certificates as in Section 5.1.

TLS data structures allow certificates of size up to 2^{24} bytes = 16 MiB, which would accommodate even very large post-quantum algorithms. However, TLS record layer fragments can be at most 16 KiB; TLS messages can be split across multiple fragments, but this increases the risk of incompatibility with poorly implemented software and can be problematic with datagram transport (UDP).

Experimental evaluation of hybrid certificates in TLS. Table 4 shows the results of testing the compatibility of a variety of TLS libraries and web browsers when using the hybrid certificates from Approach 2 of Section 5.1.

In the top half of the table, we test whether popular TLS libraries can be used to establish a TLS 1.2 connection using an RSA certificate with an extension of the given size. In each case, the experiment is carried out between that library's own TLS server and TLS client command-line programs (in the case of Java, we wrote a simple HTTPS server and client using built-in libraries). Only Java completes connections with extensions the size of a TESLA-416 [2] certificate (1.3 MiB),

	Extension size in KiB				
	1.5	3.5	9.0	43.0	1333.0
Libraries (library's command-line client talking to library's command-line server					
GnuTLS 3.5.11	✓	✓	✓	✓	×
Java SE 1.8.0_131	✓	✓	✓	✓	✓
mbedTLS 2.4.2	✓	✓	✓	×	×
NSS 3.29.1	✓	✓	✓	✓	×
OpenSSL 1.0.2k	✓	✓	✓	✓	×
Web browsers (talking to OpenSSL's command-line server)					
Apple Safari 10.1 (12603.1.30.0.34)	✓	✓	✓	✓	✓
Google Chrome 58.0.3029.81	✓	✓	✓	✓	×
Microsoft Edge 38.14393.1066.0	✓	✓	✓	×	×
Microsoft IE 11.1066.14393.0	✓	✓	✓	×	×
Mozilla Firefox 53.0	✓	✓	✓	✓	×
Opera 44.0.2510.1218	✓	✓	✓	✓	×

Table 4: Compatibility of TLS connections using hybrid X.509v3 certificates containing large extensions.

and mbedTLS cannot handle certificates with extensions the size of a SPHINCS [5] certificate (43 KiB). (For GnuTLS and OpenSSL, we found they could handle an 80 KiB extension but not a 90 KiB extension).

In the bottom half of the table, we test whether popular web browsers can be used to establish a TLS 1.2 connection to a TLS server run using the OpenSSL command-line `s_server` program using an RSA certificate with an extension of the given size. Microsoft browsers on Windows 10 cannot handle SPHINCS-sized extensions, and no browser except Safari could handle TESLA-416-sized extensions (1.3 MiB). (Curiously, Safari was able to handle a 1.3 MiB extension with an OpenSSL command-line server despite OpenSSL's own command-line client not being able to handle it.)

5.3 CMS and S/MIME

Cryptographic Message Syntax (CMS) [19] is the main cryptographic component of S/MIME [21], which enables public key encryption and digital signatures for email. In an S/MIME signed email, a header is used to specify the algorithms used, and then the body of the email is divided into chunks: one chunk is the body to be signed, and the other is a Base-64 encoding of a CMS `SignedData` object. The `SignedData` object contains several fields, including a set of certificates and a set of `SignerInfo` objects. Each `SignerInfo` object contains a signer identifier, algorithm identifier, signature, and optional signed and unsigned attributes.

Approach 1: Parallel `SignerInfos`. To construct a standards-compliant hybrid signature in S/MIME, one could put the certificate for each algorithm in the `SignedData` object's set of certificates (with no need for hybrid certificates from Section 5.1), and then include `SignerInfo` objects for the signature from each algorithm. This is an example of the concatenation combiner $C_{\|}$ from Section 4.1.

Approach 2: Nested signature in `SignerInfo` attributes. For an alternative and still standards-

	Approach								
	0	**1**	**2.a**	**2.b** (attribute size in KiB)					**2.c**
				1.5	3.5	9.0	43.0	1333.0	
Apple Mail 10.2 (3259)	✓	✓	✓	✓	✓	✓	✓	✓	✓
BouncyCastle 1.56 with Java SE 1.8.0_131	✓	–	✓	✓	✓	✓	✓	✓	✓
Microsoft Outlook 2016 16.0.7870.2031	✓	×	✓	✓	✓	✓	✓	✓	✓
Mozilla Thunderbird 45.7.1	✓	×	✓	✓	✓	✓	✓	×	✓
OpenSSL 1.0.2k	✓	×	✓	✓	✓	✓	✓	✓	✓

Approach 0: both RSA-2048.
Approach 1: one RSA-2048, one with unknown algorithm of similar key and signature size; unable to test Approach 1 on BouncyCastle.
Approach 2.a, 2.c: both RSA-2048.
Approach 2.b: outer RSA-2048, inner random binary extension of given size.

Table 5: Compatibility of hybrid S/MIME approaches.

compliant approach, we could use the optional attributes in the `SignerInfo` object to embed a second signature. We need to convey the certificate for the second algorithm, as well as the signature (using the second algorithm) for the message. There are several options based on the standards:

2.a) Put a second certificate in the set of certificates, and put a second `SignerInfo` in an attribute of the first `SignerInfo`.
2.b) Put a hybrid certificate in the set of certificates, and put a second `SignerInfo` in an attribute of the first `SignerInfo`.
2.c) Put a second `SignedData` in an attribute of the first `SignerInfo`.

These approaches require defining a new attribute type, but this is easily done. The CMS standard indicates that verifiers can accept signatures with unrecognized attributes, so this approach results in backwards-compatible signatures that should be accepted by existing software.)

If the extra data is put in the *signed* attribute of the first `SignerInfo`, then we are using the strong nesting combiner $C_{\text{str-nest}}$ from Section 4.2; if the extra data is put in the *unsigned* attribute of the first `SignerInfo`, then we are using the concatenation combiner $C_{\|}$ from Section 4.1.

Experimental evaluation of CMS and S/MIME approaches. We tested five S/MIME libraries/applications for acceptance of S/MIME messages from each approach above. The configurations and results appear in Table 5.

Regarding approach 1, the S/MIME and CMS standards appear to be silent on how to validate multiple `SignerInfo` objects: should a signed message be considered valid if *any* of the `SignerInfo` objects is valid, or only if *all* of them are? Apple Mail accepted in this case, whereas the three others we were able to test rejected, so approach 1 is not fully backwards-compatible. In principle all the tested libraries support approaches 2.a–2.c. We only tested multiple attribute sizes in one of these three approaches (2.b), but the results should generalize to 2.a and 2.c. Only Thunderbird struggled with very large attributes.

5.4 Discussion

Summarizing our experimental observations, backwards compatibility is most easily maintained when post-quantum objects can be placed as non-critical extensions (attributes in the S/MIME case) of pre-quantum objects. These constructions end up leading to one of our nested combiners, either D_{nest} for X.509 certificate extensions or $C_{\text{str-nest}}$ for S/MIME and CMS. Both these combiners offer unforgeability under the assumption that either scheme is unforgeable. For non-separability, the pre-quantum algorithm would be the "outside" signature Σ_2 in both D_{nest} and $C_{\text{str-nest}}$, and so we get 2-non-separability under unforgeability of the pre-quantum scheme, not the post-quantum scheme. Extensions up to 43.0 KiB were mostly (but not entirely) handled successfully, which covers many but not all post-quantum schemes; the largest schemes such as TESLA-416 would be more problematic to use in hybrid modes with existing software.

Acknowledgements

NB acknowledges support by the German Research Foundation (DFG) as part of project P1 within the CRC 1119 CROSSING. DS acknowledges support from Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery grant RGPIN-2016-05146.

References

[1] S. Akleylek, N. Bindel, J. A. Buchmann, J. Krämer, and G. A. Marson. An efficient lattice-based signature scheme with provably secure instantiation. In D. Pointcheval, A. Nitaj, and T. Rachidi, editors, *AFRICACRYPT 16*, volume 9646 of *LNCS*, pages 44–60. Springer, Heidelberg, Apr. 2016.

[2] E. Alkim, N. Bindel, J. Buchmann, and Ö. Dagdelen. TESLA: Tightly-secure efficient signatures from standard lattices. Cryptology ePrint Archive, Report 2015/755, 2015. `http://eprint.iacr.org/2015/755`.

[3] P. S. L. M. Barreto, P. Longa, M. Naehrig, J. E. Ricardini, and G. Zanon. Sharper ring-LWE signatures. Cryptology ePrint Archive, Report 2016/1026, 2016. `http://eprint.iacr.org/2016/1026`.

[4] M. Bellare and P. Rogaway. Optimal asymmetric encryption. In A. D. Santis, editor, *EUROCRYPT'94*, volume 950 of *LNCS*, pages 92–111. Springer, Heidelberg, May 1995.

[5] D. J. Bernstein, D. Hopwood, A. Hülsing, T. Lange, R. Niederhagen, L. Papachristodoulou, M. Schneider, P. Schwabe, and Z. Wilcox-O'Hearn. SPHINCS: Practical stateless hash-based signatures. In E. Oswald and M. Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 368–397. Springer, Heidelberg, Apr. 2015.

[6] D. Boneh and M. Zhandry. Secure signatures and chosen ciphertext security in a quantum computing world. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 361–379. Springer, Heidelberg, Aug. 2013.

[7] J. W. Bos, C. Costello, M. Naehrig, and D. Stebila. Post-quantum key exchange for the TLS protocol from the ring learning with errors problem. In *2015 IEEE Symposium on Security and Privacy*, pages 553–570. IEEE Computer Society Press, May 2015.

[8] M. Braithwaite. Google Security Blog: Experimenting with post-quantum cryptography, July 2016. `https://security.googleblog.com/2016/07/experimenting-with-post-quantum.html`.

[9] J. Buchmann, E. Dahmen, and A. Hülsing. XMSS - a practical forward secure signature scheme based on minimal security assumptions. In B.-Y. Yang, editor, *PQCrypto 2011*, volume 7071 of *LNCS*, pages 117–129. Springer, 2011.

[10] M. Campagna and et al. Quantum safe cryptography and security: An introduction, benefits, enablers and challengers. Technical report, ETSI (European Telecommunications Standards Institute), June 2015.

[11] A. I.-T. Chen, M.-S. Chen, T.-R. Chen, C.-M. Cheng, J. Ding, E. L.-H. Kuo, F. Y.-S. Lee, and B.-Y. Yang. SSE implementation of multivariate PKCs on modern x86 CPUs. In C. Clavier and K. Gaj, editors, *CHES 2009*, volume 5747 of *LNCS*, pages 33–48. Springer, Heidelberg, Sept. 2009.

[12] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280 (Proposed Standard), May 2008.

[13] N. de Beaudrap, R. Cleve, and J. Watrous. Sharp quantum versus classical query complexity separations. *Algorithmica*, 34(4):449–461, 2002.

[14] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard), Aug. 2008.

[15] L. Ducas, A. Durmus, T. Lepoint, and V. Lyubashevsky. Lattice signatures and bimodal Gaussians. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 40–56. Springer, Heidelberg, Aug. 2013.

[16] R. Fischlin and C.-P. Schnorr. Stronger security proofs for RSA and Rabin bits. *Journal of Cryptology*, 13(2):221–244, 2000.

[17] S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, Apr. 1988.

[18] T. Güneysu, V. Lyubashevsky, and T. Pöppelmann. Practical lattice-based cryptography: A signature scheme for embedded systems. In E. Prouff and P. Schaumont, editors, *CHES 2012*, volume 7428 of *LNCS*, pages 530–547. Springer, Heidelberg, Sept. 2012.

[19] R. Housley. Cryptographic Message Syntax (CMS). RFC 5652 (INTERNET STANDARD), Sept. 2009.

[20] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information.* Cambridge University Press, 2000.

[21] B. Ramsdell and S. Turner. Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification. RFC 5751 (Proposed Standard), Jan. 2010.

[22] E. Rescorla. The Transport Layer Security (TLS) protocol version 1.3, draft 19, March 2017. `https://tools.ietf.org/html/draft-ietf-tls-tls13-19`.

[23] N. Sullivan. Exported authenticators in TLS, draft 01, March 2017. `https://tools.ietf.org/html/draft-sullivan-tls-exported-authenticator-01`.

A A brief review of quantum computation

A full explanation of quantum computation is beyond the scope of this paper; see a standard text such as Nielsen and Chuang [20]. We can rely on a subset of quantum computation knowledge.

A quantum system is a complex Hilbert space $\mathcal{H}$ with an inner product. Vectors in $\mathcal{H}$ are typically denoted using "ket" notation, such as $|x\rangle$, and the complex conjugate transpose of $|y\rangle$ is denoted by $\langle y|$, so that their inner product of $|x\rangle$ and $|y\rangle$ is given by $\langle y|x\rangle$. A quantum state is a vector in $\mathcal{H}$ of norm 1. For two quantum systems $\mathcal{H}_1$ and $\mathcal{H}_2$, the joint quantum system is given by the tensor product $\mathcal{H}_1 \otimes \mathcal{H}_2$; for two states $|x_1\rangle \in \mathcal{H}_1$ and $|x_2\rangle \in \mathcal{H}_2$, the joint state is denoted by $|x_1\rangle |x_2\rangle$, or more compactly as $|x_1, x_2\rangle$.

Some quantum states can be represented as superpositions of other quantum states, such as $|x\rangle = \frac{1}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2}} |1\rangle$. More generally, if $\{|x\rangle\}_x$ is a basis for $\mathcal{H}$, then we can write any superposition in the form

$$|y\rangle = \sum_x \psi_x |x\rangle$$

where ψ_x are complex numbers such that $|y\rangle$ has norm 1.

Quantum operations on $\mathcal{H}$ can be represented by unitary transformations $\mathbf{U}$. A side effect of the fact that quantum operations are unitary transformations is that quantum computation (prior to measurement) is reversible, imposing some constraints on how we quantize classical computations. In particular, suppose we want to quantize a classical algorithm A which takes an input $x \in \{0,1\}^a$ and gives an output $y \in \{0,1\}^b$. First, we would imagine the classical reversible mapping $\{0,1\}^a \times \{0,1\}^b \to \{0,1\}^a \times \{0,1\}^b : (x,t) \mapsto (x, t \oplus A(x))$. Then we construct the corresponding unitary transformation $\mathbf{A}$ which acts linearly on superpositions of such states:

$$\mathbf{A} : \sum_{x,t} \psi_{x,t} |x, t\rangle \mapsto \sum_{x,t} \psi_{x,t} |x, t \oplus A(x)\rangle \quad .$$

For full generality, we may allow a workspace register alongside the input and output registers, and thus we in fact use

$$\mathbf{A} : \sum_{x,t,z} \psi_{x,t,z} |x, t, z\rangle \mapsto \sum_{x,t,z} \psi_{x,t,z} |x, t \oplus A(x), z\rangle \quad .$$

B $C_{\text{weak-nest}}$: Weak nesting

Here, one signature is nested inside another: the second signature algorithm is used to sign the signature generated by the first signature algorithm.

- $C_{\text{weak-nest}}(\Sigma_1, \Sigma_2).\mathsf{Sign}(sk', m)$: $\sigma_1 \leftarrow_\$ \Sigma_1.\mathsf{Sign}(sk_1, m)$, $\sigma_2 \leftarrow_\$ \Sigma_2.\mathsf{Sign}(sk_2, \sigma_1)$. Return $\sigma' \leftarrow (\sigma_1, \sigma_2)$.

Unforgeability of $C_{\text{weak-nest}}$ depends crucially on the unforgeability of Σ_1 instead of on both Σ_1 and Σ_2 as for the other proposed combiners. Hence, this combiner might not be as appropriate in our setting as our other proposals. Nevertheless, we give unforgeability and non-separability statements in the following.

Theorem 11 (Unforgeability of $C_{\text{weak-nest}}$ if Σ_1 is secure). *If Σ_1 is unforgeable in the classical (or quantum) random oracle model, then $\Sigma' = C_{\text{weak-nest}}(\Sigma_1, \Sigma_2)$ is unforgeable in the classical (or quantum, respectively) random oracle model. More precisely, if the scheme Σ_1 is $\mathsf{X}^\mathsf{y}\mathsf{Z}$-$\mathsf{eufcma}$-secure, then $\Sigma' = C_{\text{weak-nest}}(\Sigma_1, \Sigma_2)$ is $\mathsf{X}^\mathsf{y}\mathsf{Z}$-$\mathsf{eufcma}$-secure.*

Proof. The proof follows the idea of the proof of Theorem 6 closely. Suppose $\mathcal{A}$ is an $\mathsf{X^yZ}$-adversary that finds a forgery in $\Sigma' = C_{\text{weak-nest}}(\Sigma_1, \Sigma_2)$ — in other words, it outputs $q_S + 1$ valid signatures under Σ' on distinct messages. We can construct an $\mathsf{X^yZ}$ algorithm $\mathcal{B}_1$ that finds a forgery in Σ_1. $\mathcal{B}_1$ interacts with an $\mathsf{X^yZ}$ challenger for Σ_1 which provides a public key vk_1. $\mathcal{B}_1$ generates a key pair $(sk_2, vk_2) \leftarrow\!\$\; \Sigma_2.\mathsf{KeyGen}()$ and sets the public key for Σ' to be (vk_1, vk_2). When $\mathcal{A}$ asks for $\sum_{m,t,z} \psi_{m,t,z} |m, t, z\rangle)$ to be signed using Σ', we treat t as consisting of two registers $t_1 \| t_2$, $\mathcal{B}_1$ proceeds by passing the m, t_1, and z registers to its signing oracle for Σ_1, then runs the quantum signing operation from Figure 2 for $\Sigma_2.\mathsf{Sign}$ on the m, t_2, and z registers. There is a one-to-one correspondence between $\mathcal{A}$'s queries to its signging oracle and $\mathcal{B}_1$'s queries to its signing oracle. As before in the proof of Theorem 6, if Σ_1 is proven to be secure in the random oracle model, then this proof of $C_{\text{weak-nest}}(\Sigma_1, \Sigma_2)$ also proceeds in the random oracle model: $\mathcal{B}_1$ relays $\mathcal{A}$'s hash oracle queries directly to its hash oracle, giving a one-to-one correspondence between $\mathcal{A}$'s queries to its (classical or quantum) hash oracle and $\mathcal{B}_1$'s queries to its (classical or quantum, respectively) hash oracle.

If $\mathcal{A}$ wins the $\mathsf{X^yZ\text{-}eufcma}$ game, then it has returned $q_S + 1$ valid signatures $\sigma'_i = (\sigma'_{i,1}, \sigma'_{i,2})$ on distinct messages m_i such that $\Sigma_1.\mathsf{Verify}(vk_1, m_i, \sigma'_{i,1}) = 1$ and $\Sigma_2.\mathsf{Verify}(vk_2, \sigma'_{i,1}, \sigma'_{i,2}) = 1$. $\mathcal{B}_1$ can extract from this $q_S + 1$ valid signatures under Σ_1 on distinct messages. Thus, $\mathrm{Adv}^{\mathsf{X^yZ\text{-}eufcma}}_{\Sigma'}(\mathcal{A}) \leq \mathrm{Adv}^{\mathsf{X^yZ\text{-}eufcma}}_{\Sigma_1}(\mathcal{B}_1)$. □

Forgeability of $C_{\text{weak-nest}}$ if Σ_1 is insecure. However, if Σ_1 is forgeable—for example, if messages are hashed before signing and it is easy to find collisions in that hash function—then $\Sigma' = C_{\text{weak-nest}}(\Sigma_1, \Sigma_2)$ is insecure.

Separability of $C_{\text{weak-nest}}$. $C_{\text{weak-nest}}$ is not 1-non-separable: σ_1 is immediately a Σ_1-signature for m, with no way of recognizing this as being different from typical Σ_1 signatures.

Theorem 12 (2-non-separability of $C_{\text{weak-nest}}$)**.** *Let Σ_2 be $\mathsf{X^cZ\text{-}eufcma}$-secure and Σ_1 be a probabilistic signature scheme with the probability that two signatures $\sigma_1, \sigma_2 \leftarrow \Sigma_1.\mathsf{Sign}(sk, \mu)$ to the same message μ are equal, is smaller than $c \in (0,1)$. Then $\Sigma' = C_{\text{weak-nest}}(\Sigma_1, \Sigma_2)$ is $\mathsf{X^cZ\text{-}2\text{-}nonsep}$ with respect to recognizer algorithm $R(m) = (m \in \mathcal{S}_{\Sigma_1})$.*

Remark 1. While our definition of $\mathsf{X^yZ}\text{-}\tau\text{-}\mathsf{nonsep}$ is stated for all security notions $\mathsf{X^yZ} \in \{\mathsf{C^cC}, \mathsf{C^cQ}, \mathsf{Q^cQ}, \mathsf{Q^qQ}\}$, our positive results on non-separability such as Theorem 12 only hold for $\mathsf{y} = \mathsf{c}$, i.e., when the adversary's signing oracle is classical. Our results attempt to reduce non-separability to unforgeability of one of the underlying signature schemes. In the quantum setting, breaking unforgeability demands returning $q_S + 1$ valid signatures, whereas breaking non-separability demands returning just 1 valid separated signature. A reduction involving a classical signing oracle can store a copy of the results, but not so with a quantum signing oracle. Modifying the non-separability experiment to demand return of $q_S + 1$ (valid) signatures would also be problematic, since it would make non-separability imply unforgeability.

Proof of Theorem 12 – 2-non-separability of $C_{\text{weak-nest}}$. We can construct a reduction $\mathcal{B}_2$ which is an $\mathsf{X^cZ\text{-}eufcma}$ adversary for Σ_2. $\mathcal{B}_2$ generates a keypair (vk_1, sk_1) for Σ_1, and interacts with an $\mathsf{X^cZ\text{-}eufcma}$ challenge for Σ_2. When $\mathcal{A}$ classically queries its signing oracle to obtain a signature under Σ' of m_i, $\mathcal{B}_2$ signs m_i with Σ_1 to obtain $\sigma_{i,1}$. Moreover, $\mathcal{B}$ remembers the already computed signature values for Σ_1. If a signature value was already computed for another message, then $\mathcal{B}$ computes a new signature value with new randomness. By assumption on the signature scheme Σ_1, two signatures to the same message are the same with probability smaller than c. Hence, $\mathcal{B}$ has to

compute $\lceil \frac{1}{1-c} \rceil$ signatures (expected) to ensure that $\sigma_{i,1} \neq \sigma_{j,2}$ for $i \neq j$. Afterwards, $\mathcal{B}$ passes $\sigma_{i,1}$ to its Σ_2 signing oracle and returns the resulting $\sigma_{i,2}$ to $\mathcal{A}$. Eventually, $\mathcal{A}$ returns (μ^*, σ^*) such that $\Sigma_2.\mathsf{Verify}(vk_2, \mu^*, \sigma^*) = 1$ but $\Sigma'.R(\mu^*) = 0$, i.e., $\mu^* \notin \mathcal{S}_{\Sigma_1}$. This means in particular that $\mu^* \neq \sigma_{1,i}$ for all i. Moreover, all the $\sigma_{1,i}$ are distinct.
This means we have $q_S + 1$ valid message-signature pairs under Σ_2, yielding a successful forgery for the $\mathsf{X^cZ\text{-}eufcma}$ experiment for Σ_2. Thus,

$$\Pr[S_1] \leq \mathrm{Adv}_{\Sigma_2}^{\mathsf{X^cZ\text{-}eufcma}}(\mathcal{B}_2) \ .$$

□

Proof sketch of Theorem 10 – 2-non-separability of D_{nest}. This proof follows the same approach as the proof of 2-non-separability for $C_{\text{str-nest}}$ (Theorem 8).

We can construct a reduction $\mathcal{B}_2$ which is an $\mathsf{X^cZ\text{-}eufcma}$ adversary for Σ_2. $\mathcal{B}_2$ generates a keypair (vk_1, sk_1) for Σ_1, and interacts with an $\mathsf{X^cZ\text{-}eufcma}$ challenge for Σ_2. When $\mathcal{A}$ classically queries its signing oracle to obtain a signature under Σ' of $(m_{i,1}, m_{i,2})$, $\mathcal{B}_2$ signs $m_{i,1}$ with Σ_1 to obtain $\sigma_{i,1}$. Afterwards, $\mathcal{B}$ passes $(m_{i,1}, \sigma_{i,1}, m_{i,2})$ to its Σ_2 signing oracle and returns the resulting $\sigma_{i,2}$ to $\mathcal{A}$. Eventually, $\mathcal{A}$ returns (μ^*, σ^*) such that $\Sigma_2.\mathsf{Verify}(vk_2, \mu^*, \sigma^*) = 1$ but $\Sigma'.R(\mu^*) = 0$, i.e., $\mu^* \notin \{0,1\}^* \times \mathcal{S}_{\Sigma_1} \times \{0,1\}^*$. This means in particular that $\mu^* \neq (m_{i,1}, \sigma_{1,i}, m_{i,2})$ for all i. Moreover, all the $(m_{i,1}, \sigma_{1,i}, m_{i,2})$ are distinct, since all $(m_{i,1}, m_{i,2}$ are distinct. This means we have $q_S + 1$ valid message-signature pairs under Σ_2, yielding a successful forgery for the $\mathsf{X^cZ\text{-}eufcma}$ experiment for Σ_2. Thus,

$$\Pr[S_1] \leq \mathrm{Adv}_{\Sigma_2}^{\mathsf{X^cZ\text{-}eufcma}}(\mathcal{B}_2) \ .$$

□