BUFFing signature schemes beyond unforgeability and the case of post-quantum signatures[*]

Version 1.1[†], March 2021

Cas Cremers[1]   Samed Düzlü[2]   Rune Fiedler[3]   Marc Fischlin[3]   Christian Janson[3]

[1] CISPA Helmholtz Center for Information Security, Germany
cremers@cispa.de
[2] QPC, Technische Universität Darmstadt, Germany
samed@qpc.tu-darmstadt.de
[3] Cryptoplexity, Technische Universität Darmstadt, Germany
{rune.fiedler, marc.fischlin, christian.janson}@cryptoplexity.de

Abstract. Modern digital signature schemes can provide more guarantees than the standard notion of (strong) unforgeability, such as offering security even in the presence of maliciously generated keys, or requiring to know a message to produce a signature for it. The use of signature schemes that lack these properties has previously enabled attacks on real-world protocols. In this work we revisit several of these notions beyond unforgeability, establish relations among them, provide the first formal definition of non re-signability, and a transformation that can provide these properties for a given signature scheme in a provable and efficient way.

Our results are not only relevant for established schemes: for example, the ongoing NIST PQC competition towards standardizing post-quantum signature schemes has six finalists in its third round. We perform an in-depth analysis of the candidates with respect to their security properties beyond unforgeability. We show that many of them do not yet offer these stronger guarantees, which implies that the security guarantees of these post-quantum schemes are not strictly stronger than, but instead incomparable to, classical signature schemes. We show how applying our transformation would efficiently solve this, paving the way for the standardized schemes to provide these additional guarantees and thereby making them harder to misuse.

Keywords. Digital signature scheme · exclusive ownership · DSKS attack · non re-signability · message-bound signatures · NIST PQC candidates

---

[*]An extended abstract of this paper appears in the proceedings of IEEE S&P 2021. This is the full version.
[†]We provide an overview of major changes between versions in Appendix C.

Contents

1 Introduction

For digital signature schemes, there are two classical security notions: EUF-CMA, existential unforgeability [GMR88], and the stronger notion SUF-CMA, strong existential unforgeability. These security notions guarantee that signatures cannot be forged under the given public key. However, there is more to be said about the security properties of signatures beyond unforgeability: for example, the impact of maliciously generated keys, the interdependence of keys, or whether one needs to know a message to be able to produce a signature for it. In [PS05, JCCS19, BCJZ20] it was shown that some classical signature schemes provide better guarantees than others in this respect.

We highlight three main properties beyond unforgeability:

The first is **exclusive ownership** [PS05] (which generalizes earlier notions of Duplicate-Signature Key Selection (DSKS) attacks [BWM99, MS04]): the property that a signature only verifies under a single public key. For example, an early version of Let's Encrypt's ACME protocol [BHAK15b, BHAK15a] was vulnerable to an attack because the used signature scheme (RSA) did not provide this property. The protocol's goal was to act as an automatic certificate authority: to obtain evidence that a key owner has admin access to a website, upon which it will sign a certificate for the website and the signature verification key. The evidence consisted of, e.g., placing a signed challenge in a privileged position on the website or DNS records. While RSA signatures provide unforgeability, they allow constructing another key pair under which a given signature verifies. The attack [Aye15a, Aye15b] "hijacks" an existing signed challenge that is still present on a website, constructs a new key pair under whose public key the existing signature verifies, and then claims ownership. This causes the CA to produce a valid certificate for the attacker on the target website. In [JCCS19] an attack was found on the X509-Mutual authentication/WS-Security protocol that also exploits generating a new key pair for a given signature.

The second is **message-bound signatures** (a.k.a. non-colliding signatures): the property that a signature is only valid for a unique message. Signature schemes such as DSA and ECDSA do not provide this property. A possible cause can be the presence of *weak keys* that verify multiple or even all messages. The absence of this property can lead to problems in protocols that depend on uniqueness properties in the presence of adversarially chosen keys.

The third property is **non re-signability** [JCCS19] meaning that one cannot produce a signature under another key given a signature for some unknown message m. One might expect that to produce another valid signature on a message m, the signer needs to know m. However, this is not the case for, e.g., RSA signatures, where given a signature on m, another signature can be produced even without knowing m. In [JCCS19] an attack was found on the DRKey/OPT protocols for secure routing (intended for the SCION architecture) that exploits this possibility. The protocols aim to provide partial path integrity guarantees even in the presence of malicious intermediate nodes by having each intermediate node sign a symmetric key that they will share with the endpoint. Malicious nodes could violate the intended path integrity guarantees by claiming that a signature from an honest node on the path in fact came from another (colluding) malicious node, thereby making the endpoints believe that the path did not go through this honest node. This property was first proposed and defined in the symbolic model in [JCCS19]. However, until now, no formal cryptographic definition was proposed.

While there are classical signature schemes that violate each of the above properties, this need not be the case: It was proven in [BCJZ20] that the LibSodium variant of the Ed25519 signature scheme satisfies the first two properties, and the third follows by construction. The real-world implication is that depending on which signature scheme is used, the security protocols above could either be secure or insecure. From the perspective of the design of a signature scheme, it is therefore prudent to aim for the strongest guarantees from the primitive, such that the expectations of implementers are not accidentally (and needlessly) violated.

In this work, we revisit the security properties that go beyond unforgeability of signature schemes

and provide new theoretical results, including new formal definitions, establishing relations between them, and providing a simple generic transformation that provably achieves them. Our transformation is highly efficient and only increases the size of the signature moderately by a single hash digest.

Our work is partly driven by the ongoing NIST competition for post-quantum secure digital signature schemes. The schemes that have made it to round 3 are designed to be resilient against much stronger (quantum) adversaries than previous schemes, and one might therefore expect them to provide strictly stronger security properties than existing signature schemes.

Our analysis of the round 3 candidates with respect to these properties reveals that these schemes do not necessarily provide modern security properties beyond unforgeability. For example, we find that while CRYSTALS-Dilithium provides all three properties, exclusive ownership, message-bound signatures, and non re-signability, FALCON and Rainbow do not. Remarkably, this implies that e.g. Libsodium's Ed25519 provides security properties that some post-quantum candidates do not. Concretely, this would mean that implementing the previously mentioned protocols with FALCON or Rainbow would enable (classical) protocol attacks that would have been impossible with Libsodium's Ed25519. Fortunately, our transformation can be applied to the vulnerable schemes to remedy this situation.

In many ways, the situation for the NIST competition is similar to hash functions and length extension attacks in the context of the NIST SHA-3 competition. While length extension attacks had been known for years, they were not excluded by the standard hash function definitions. As a result, older schemes were not considered in this light, leading to attacks on e.g. Flickr [DR09] and TLS, IKE, and SSH [BL16]. In the final SHA-3 standard, only schemes were chosen that provide resilience against length extension attacks, even though the standard hash function definition does not require it:

> *"The SHA-3 functions are also designed to resist other attacks, such as length-extension attacks, that would be resisted by a random function of the same output length, in general providing the same security strength as a random function, up to the output length."* [Nat15a, p. 24]

Similarly, we would expect the final NIST selections for the post-quantum signature schemes to provide the strongest modern guarantees, such as offering built-in protection against maliciously generated keys, instead of leaving this up to the protocols that use the schemes. Our work therefore also fits into the wider positive trend of misuse-resistance: creating cryptographic primitives that are hard to misuse.

Our main contributions are:

- We provide new theoretical results for three security properties of signature schemes beyond unforgeability: exclusive ownership (M-S-UEO, and weaker variants such as S-CEO and S-DEO), message-bound signatures (MBS), and non re-signability (NR). Notably, we provide the first cryptographic definition for non re-signability, and construct a generic BUFF (*Beyond UnForgeability Features*) transformation that provably achieves all three properties. Our results are generic and apply equally to the classical and the post-quantum setting.

- We apply our theory in practice and perform the first analysis of the round 3 NIST candidates for post-quantum secure signature schemes w.r.t. these properties. We give an overview of our results in Table 1. We show that the security of several round 3 candidates is not strictly stronger than that of existing classical schemes: schemes like LibSodium Ed25519 offer security guarantees that FALCON and Rainbow do not. However, our simple transformation can remedy this situation: we show the minimal impact of applying the BUFF transformation to the round 3 candidates, which shows that it is practical to provably offer these properties.

The remainder of this paper is structured as follows. In Section 2 we introduce notation and further preliminaries. Section 3 overviews previous work on security properties of signatures beyond unforgeability.

In Section 4 we present our main theoretical results. In Section 5 we analyze the finalists to the NIST competition for post-quantum signature schemes w.r.t. the three security properties beyond unforgeability. We conclude in Section 6.

Table 1: Several NIST PQ Signature scheme Round 3 candidates and alternate ones lack desirable security properties beyond unforgeability. We denote by ✓ a proof of the property (under rational assumptions), by ✗ an attack against it, and by • that we currently have no proof based on standard assumptions. We provide an overview of the detailed analyses of the schemes in their versions as submitted to Round 3 in the table. Note that the signature schemes that do not have the M-S-UEO property, they do also not achieve the weaker variants S-CEO and S-DEO.

The "Conclusion" column summarizes for each scheme: ✓ indicates all properties hold. For schemes with ✗ or •, our generic transformation from Section 4.4 provably provides all properties at the cost of a slight increase in signature size (see Figure 6).

	Round 3 scheme	malicious strong univ. exclusive ownership M-S-UEO (Def. 3.1)		message-bound signatures MBS (Def. 3.2)		no re-signing without message NR (Def. 4.3)		Conclusion
main	CRYSTALS-Dilithium	✓	Prop. 5.1	✓	Prop. 5.1	✓	Prop. 5.1	✓
	FALCON	✗	Prop. 5.5	✓	Prop. 5.3	✗	Prop. 5.6	✗
	Rainbow Standard	✗	Prop. 5.9	✓	Prop. 5.7	✗	Prop. 5.8	✗
	Rainbow CZ & Compr.	•	Sec. 5.3	✓	Prop. 5.7	✗	Prop. 5.8	✗
alternate	GeMSS	✗	Prop. 5.10	✗	Prop. 5.12	✗	Prop. 5.13	✗
	Picnic	✓	Prop. 5.14	✓	Prop. 5.14	✓	Prop. 5.14	✓
	SPHINCS$^+$	•	Sec. 5.6	✓	Prop. 5.15	•	Sec. 5.6	•

2 Preliminaries

2.1 Notation

We denote by $\lambda \in \mathbb{N}$ the security parameter (usually written in unary as 1^λ) that is implicitly given to all algorithms. A function $\mu\colon \mathbb{N} \to \mathbb{R}$ is called *negligible* if, for every constant $c \geq 0$, there exists $\lambda_c \geq 0$ such that for all $\lambda \geq \lambda_c$ we have that $\mu(\lambda) \leq \lambda^{-c}$. Furthermore, we assume that all algorithms (unless specified otherwise) run in probabilistic polynomial-time which we abbreviate by PPT. Note that, as usual, we state the security notions and assumptions asymptotically, with respect to polynomial-time adversary and negligible functions. It is understood that, when analyzing actual schemes with concrete parameters, these terms must be interpreted accordingly as "reasonable" run time and success probabilities in light of the parameters.

We write a bit as $b \in \{0,1\}$ and its inversion simply as $\bar{b}$. Furthermore, we denote a (bit) string as $s \in \{0,1\}^*$ and by $|s|$ we denote its binary length. By $s\|t$ we denote the concatenation of two strings s and t but we usually assume that the encoding is such that one can recover s and t from $s\|t$, e.g., when s is of fixed length. A tuple (s,t) of strings is implicitly encoded as a single bit string if required, e.g., when processing the tuple by a hash algorithm. We assume that such encodings are one-to-one but usually omit the details. For a (finite) set S, we use the notation $s \leftarrow\!\!{\scriptstyle\$}\, S$ to denote that the string s was sampled uniformly at random from S. We also use this notation $y \leftarrow\!\!{\scriptstyle\$}\, A(x)$ to denote the random output y of

algorithm A for input x, where the probability is over A's internal randomness. We simply use the arrow $\leftarrow$ for any assignment statements.

Let P be any statement that can either be true or false, then the Iverson bracket notation $[P]$ stands for 1 if the statement is true and 0 otherwise. We often identify the Boolean variables true and false with 1 and 0, respectively. A bold variable $\mathbf{v}$ denotes a vector, a bold capital letter $\mathbf{A}$ denotes a matrix and $\mathbf{A^T}$ denotes the transposed matrix. The spectral norm of a vector $\mathbf{v}$ is denoted by $\|\mathbf{v}\|^2$.

We use the notion of min-entropy to quantify the uncertainty of the adversary about unknown data. Specifically, we follow Dodis et al. [DRS04] and define the average conditional min-entropy of random variables X and Y as $\tilde{H}_\infty(X|Y) = -\log \mathbb{E}_{y \leftarrow Y}(\max_x \Pr[X = x \mid Y = y])$. This describes the min-entropy in X given Y, but averages over the sampling of Y. For our applications it usually suffices to use the computational counterpart of this entropy, denoted as HILL entropy [HLR07]. A random variable X has average conditional HILL entropy $\tilde{H}_\infty^{\text{HILL}}(X|Y) \geq k$ conditioned on Y, if there is a random variable X' which is computationally indistinguishable from X, and such that $\tilde{H}_\infty(X'|Y) \geq k$.

2.2 Digital Signature Schemes

In the following, we present the basic definition of a digital signature scheme as well as its security properties.

Definition 2.1. *A* digital signature scheme *is a tuple of three* PPT *algorithms* $\Pi = (\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$ *with associated message space* $\mathcal{M}$*, defined as follows:*

- $(\mathsf{sk}, \mathsf{pk}) \leftarrow_\$ \mathsf{KGen}(1^\lambda)$*: On input the security parameter, this randomized algorithm returns a key pair* $(\mathsf{sk}, \mathsf{pk})$*;*

- $\sigma \leftarrow_\$ \mathsf{Sig}(\mathsf{sk}, m)$*: On input a signer secret key* sk *and a message* $m \in \mathcal{M}$*, this randomized algorithm returns a signature* σ*;*

- $d \leftarrow \mathsf{Vf}(\mathsf{pk}, m, \sigma)$*: On input a public verification key* pk*, a message* m*, and a candidate signature* σ*, this deterministic algorithm returns a bit* $d \in \{0, 1\}$*. If* $d = 1$ *we say that the signature is valid, otherwise not.*

We say that a digital signature scheme Π is *correct*, if there exists a negligible function $\mu \colon \mathbb{N} \to \mathbb{R}$ such that, for every security parameter $\lambda \in \mathbb{N}$, every $(\mathsf{sk}, \mathsf{pk}) \leftarrow_\$ \mathsf{KGen}(1^\lambda)$, every $m \in \mathcal{M}$, and random $\sigma \leftarrow_\$ \mathsf{Sig}(\mathsf{sk}, m)$, it holds that

$$\Pr[\mathsf{Vf}(\mathsf{pk}, m, \sigma) = 1] \geq 1 - \mu(\lambda).$$

Security of a digital signature scheme is defined in terms of unforgeability which can be formalized in different flavors. The notion we consider is called existential unforgeability under chosen-message attack. Intuitively, this covers that no efficient adversary who may query signatures for a few messages of its choice can produce a valid signature for a new message. The formal definition is given in Appendix A.1.

2.3 Hash Functions

In the following, we recall the definition of a (cryptographic) hash function as well as its security properties. Informally, a hash function compresses a string of arbitrary length to a string of fixed length.

Definition 2.2. *A* hash function *is a pair of* PPT *algorithms* $\mathcal{H} = (\mathsf{KGen}, \mathsf{H})$ *with associated input space* $\mathcal{M}$ *such that:*

- $\mathsf{hk} \leftarrow_\$ \mathsf{KGen}(1^\lambda)$*: On input the security parameter, this randomized algorithm generates a key* hk*;*

$\mathbf{Exp}_{\mathcal{H},\mathcal{A}}^{\mathsf{CR}}(\lambda)$:	$\mathbf{Exp}_{\mathcal{H},\mathcal{A}}^{\Phi\mathsf{NM}}(\lambda)$:
11 : $\mathsf{hk} \leftarrow\!\!\$\, \mathsf{KGen}(1^\lambda)$	21 : $\mathsf{hk} \leftarrow\!\!\$\, \mathsf{KGen}(1^\lambda)$
12 : $(x, x') \leftarrow\!\!\$\, \mathcal{A}(\mathsf{hk})$	22 : $(\mathcal{X}, \mathsf{state}) \leftarrow\!\!\$\, \mathcal{A}_d(\mathsf{hk})$
13 : $\mathbf{return}\ [\mathsf{H}(\mathsf{hk}, x) = \mathsf{H}(\mathsf{hk}, x') \ \wedge\ x \neq x']$	23 : $x \leftarrow\!\!\$\, \mathcal{X}$
	24 : $h_x \leftarrow\!\!\$\, \mathsf{hint}(\mathsf{hk}, x)$
	25 : $y \leftarrow \mathsf{H}(\mathsf{hk}, x)$
	26 : $(y', \phi) \leftarrow\!\!\$\, \mathcal{A}_y(y, h_x, \mathsf{state})$
	27 : $\mathbf{return}\ [\mathsf{H}(\mathsf{hk}, \phi(x)) = y' \wedge\ \phi(x) \neq x]$

Figure 1: Definition of the security properties for a hash function. On the left: Definition of the experiment $\mathbf{Exp}_{\mathcal{H},\mathcal{A}}^{\mathsf{CR}}(\lambda)$ from Definition 2.3. On the right: Definition of the experiment $\mathbf{Exp}_{\mathcal{H},\mathcal{A}}^{\Phi\mathsf{NM}}(\lambda)$ from Definition 2.4.

- $y \leftarrow \mathsf{H}(\mathsf{hk}, x)$: *On input a key* hk *and an input* $x \in \mathcal{M}$, *this deterministic algorithm outputs a (digest)* y.

The provided definition is the more general notion of hash functions as a family of keyed functions. The concrete hash function can be considered by the key hk which basically corresponds to an index choosing the appropriate function from the family of functions. Note that we usually refer to the family of hash functions as H and leave the key hk implicit.

Hash functions are usually required to meet certain security properties. Among the three most prominent ones are collision resistance, second-preimage resistance, and preimage resistance. In the following, it suffices to consider simply the first one. Intuitively, collision resistance means that it is computationally infeasible to find any two distinct inputs to the hash function which map to the same digest.

Definition 2.3. *Let* $\mathcal{H}$ *be a hash function. We say that* $\mathcal{H}$ *is* collision resistant *if, for any* PPT *algorithm* $\mathcal{A}$, *there exists a negligible function* $\mu\colon \mathbb{N} \to \mathbb{R}$ *such that, for every* $\lambda \in \mathbb{N}$, *it holds that*

$$\Pr[\mathbf{Exp}_{\mathcal{H},\mathcal{A}}^{\mathsf{CR}}(\lambda)] \leq \mu(\lambda),$$

where $\mathbf{Exp}_{\mathcal{H},\mathcal{A}}^{\mathsf{CR}}(\lambda)$ *is defined on the left-hand side in Figure 1.*

Besides collision resistance, we require another property called non-malleability, which has been introduced in the realm of hash functions by Boldyreva et al. [BCFW09]. On a high-level, non-malleability of a hash function covers that it should be computationally infeasible to modify a digest y into another digest y' such that the preimages are related. Here we follow the game-based approach called Φ-non-malleability as put forward by Baecher et al. [BFS11] where the adversary is tasked to maul the digest and also to specify a transformation ϕ of the preimage where the transformation is taken from the class Φ of admissible transformations. For instance, Φ could be the class of bit flips and ϕ would then describe the concrete positions of the flips in the input.

Definition 2.4. *Let* $\mathcal{H}$ *be a hash function. We say that* $\mathcal{H}$ *is* Φ-non-malleable *(with respect to a randomized function* hint*) if, for any* PPT *algorithm* $\mathcal{A} = (\mathcal{A}_d, \mathcal{A}_y)$, *there exists a negligible function* $\mu\colon \mathbb{N} \to \mathbb{R}$ *such that, for every* $\lambda \in \mathbb{N}$, *it holds that*

$$\Pr[\mathbf{Exp}_{\mathcal{H},\mathcal{A}}^{\Phi\mathsf{NM}}(\lambda)] \leq \mu(\lambda),$$

where $\mathbf{Exp}_{\mathcal{H},\mathcal{A}}^{\Phi\mathsf{NM}}(\lambda)$ *is defined on the right-hand side in Figure 1 and* $\phi \in \Phi$. *It is required that the algorithm* $\mathcal{A}_d$ *only outputs efficiently sampleable distributions* $\mathcal{X}$ *such that the conditional min-entropy* $\tilde{\mathsf{H}}_\infty^{HILL}(\mathcal{X}|\mathsf{KGen}, \mathsf{hint}) \in \omega(\log \lambda)$.

Note that the adversary is modeled as a two-stage algorithm where it is required that the algorithm $\mathcal{A}_d$ chooses a non-trivial distribution $\mathcal{X}$ requiring it to be unpredictable by demanding sufficient min-entropy. The game uses a function hint that models circumstantial knowledge about the preimage.

Baecher et al. [BFS11] discuss some function classes Φ for which the notion is achievable for constructions like Merkle–Damgård hash functions like SHA-2 based on ideal round functions. This class includes for example bit flips, as we need for our application (but not length extensions). We note that the argument extends to SHA-3 and close derivatives thereof. We discuss the assumption in light of the concrete hash functions in the signature schemes when looking at specific schemes.

We note that if we model $\mathcal{H}$ as a random oracle then the hash function satisfies the definition of Φ-non-malleability for any class Φ where the functions ϕ preserve sufficient entropy in x, as will be the case for our results. The reason is that the adversary can only output a related random oracle value y' if it has queried the random oracle about $\phi(x)$ before. But this is infeasible if $\phi(x)$ still contains enough entropy.

3 Background on Security Notions beyond Unforgeability

In this section, we revisit security properties of signature schemes that go beyond unforgeability, namely exclusive ownership, message-bound signatures, and non re-signability, and provide their appropriate game-based formalizations. In series of works it has been shown that the absence of these properties can lead to real-world attacks such as [MS04, PS05, JCCS19, BRS06, BWM99, BK00]. In [JCCS19], Jackson et al. analyzed each property in light of requirements for security protocols, and developed new symbolic models capturing those behaviors and used these with the Tamarin prover to find new protocol attacks or prove their absence. Those discussions were the starting point of this work to re-visit these notions and hence introduce "updated" notions. These security notions can also be used by protocol designers to argue about their requirements for signature schemes.

3.1 Exclusive Ownership

In the following, we consider several notions of exclusive ownership. All of the notions consider in different flavours whether a given signature can verify under a second public key. Initially, Pornin and Stern introduced in [PS05] the notions of conservative exclusive ownership (CEO), destructive exclusive ownership (DEO) as well as the combined notion universal exclusive ownership (UEO). The underlying ideas go back to Blake-Wilson and Menezes' Duplicate-Signature Key Selection (DSKS) attacks [BWM99] which were generalized by Menezes and Smart who termed this notion key substitution attack [MS04].

Let us briefly recall the intuition behind the initial formalizations of CEO and DEO. Both notions share that the attacker is given a legitimate public key pk along with a signature σ and a message m. In CEO, the attacker's goal is to output a new public key pk$'$ which verifies the signature σ for message m. In contrast, DEO requires the same with the change that the signature verifies for a *different* message m'. Note that Pornin and Stern formalized those notions as known-message attacks where an attacker has to output a new public key along with a corresponding secret key satisfying some correctness property.

Brendel et al. [BCJZ20] introduced two strictly stronger variants of universal exclusive ownership, prefixed strong and malicious-strong. These stronger variants model a chosen-message attack, where the attacker has to output a new public key without a corresponding secret key. The attacker against the strong property is given the first public key, while the attacker against the malicious-strong property may choose the first public key itself. In the following, we review the notion of malicious-strong universal exclusive ownership as formalized in [BCJZ20].

Malicious-strong universal exclusive ownership (M-S-UEO) is the strongest variant of the exclusive ownership notions presented in this paper. Here the attacker's goal is to output a tuple containing two (distinct) public keys pk$_1$ and pk$_2$, two messages m_1 and m_2 along with a signature σ such that this

$\mathbf{Exp}_{\Pi,\mathcal{A}}^{\mathsf{M\text{-}S\text{-}UEO}}(\lambda):$	$\mathbf{Exp}_{\Pi,\mathcal{A}}^{\mathsf{MBS}}(\lambda):$
$11:\quad (m_1, m_2, \sigma, \mathsf{pk}_1, \mathsf{pk}_2) \leftarrow\!\$\, \mathcal{A}()$	$21:\quad (m_1, m_2, \sigma, \mathsf{pk}) \leftarrow\!\$\, \mathcal{A}()$
$12:\quad d_1 \leftarrow \mathsf{Vf}(\mathsf{pk}_1, m_1, \sigma)$	$22:\quad d_1 \leftarrow \mathsf{Vf}(\mathsf{pk}, m_1, \sigma)$
$13:\quad d_2 \leftarrow \mathsf{Vf}(\mathsf{pk}_2, m_2, \sigma)$	$23:\quad d_2 \leftarrow \mathsf{Vf}(\mathsf{pk}, m_2, \sigma)$
$14:\quad \mathbf{return}\ \ [d_1 = 1 \wedge d_2 = 1 \wedge\ \mathsf{pk}_1 \neq \mathsf{pk}_2]$	$24:\quad \mathbf{return}\ \ [d_1 = 1\ \wedge\ d_2 = 1\ \wedge\ m_1 \neq m_2]$

Figure 2: Definition of the experiments $\mathbf{Exp}_{\Pi,\mathcal{A}}^{\mathsf{M\text{-}S\text{-}UEO}}(\lambda)$ and $\mathbf{Exp}_{\Pi,\mathcal{A}}^{\mathsf{MBS}}(\lambda)$ from Definitions 3.1 and 3.2, respectively.

signature individually verifies with both (pk_1, m_1) and (pk_2, m_2). Note that this notion corresponds to a scenario where a malicious signer may want to create ambiguity regarding the used signing keys, or where it aims to reuse a signature in a context that requires the verification keys to be different.

Definition 3.1. *Let Π be a digital signature scheme. We say that Π provides* malicious-strong universal exclusive ownership *(M-S-UEO) if, for every* PPT *algorithm $\mathcal{A}$, there exists a negligible function $\mu\colon \mathbb{N} \to \mathbb{R}$ such that, for every $\lambda \in \mathbb{N}$,*

$$\Pr[\mathbf{Exp}_{\Pi,\mathcal{A}}^{\mathsf{M\text{-}S\text{-}UEO}}(\lambda)] \leq \mu(\lambda),$$

where $\mathbf{Exp}_{\Pi,\mathcal{A}}^{\mathsf{M\text{-}S\text{-}UEO}}(\lambda)$ is defined on the left-hand side in Figure 2.

Note that this formalization allows the adversary to generate both key pairs, and thus there is no need for a signing oracle. In Appendix B, we formally prove that M-S-UEO is strictly stronger than the variant S-UEO, and hence also stronger than any other notion of exclusive ownership introduced in this paper.

3.2 Message-bound signatures

On an intuitive level, message-bound signatures capture the adversary's inability to generate a signature and a public key under which several adversarially chosen messages verify. If this were the case, an attacker could switch a message after signing, i.e., claiming that it actually signed a *different* message. Similar to exclusive ownership, this property is not covered by EUF-CMA because it may involve a maliciously generated public key. This property was initially discussed by Stern et al. [SPMS02] with the name *duplicate signature* where they provide a particular example for ECDSA, and later formally specified by Jackson et al. [JCCS19] in the symbolic model as *non-colliding signatures*. This symbolic definition does not require the adversary to specify or know the messages for which the signature verifies.

The first game-based formalization of this notion was provided by Brendel et al. [BCJZ20], who introduced the term message-bound signatures. We provide the formal details on the right-hand side in Figure 2. In the security experiment, we require the adversary to output *two* messages, a signature and a public key. It wins the game if both messages are not identical and if the signature verifies correctly for each message under the public key.

Definition 3.2. *Let Π be a digital signature scheme. We say that Π provides* message-bound signatures *(MBS) if, for every* PPT *algorithm $\mathcal{A}$, there exists a negligible function $\mu\colon \mathbb{N} \to \mathbb{R}$ such that, for every $\lambda \in \mathbb{N}$, it holds that*

$$\Pr[\mathbf{Exp}_{\Pi,\mathcal{A}}^{\mathsf{MBS}}(\lambda)] \leq \mu(\lambda),$$

where $\mathbf{Exp}_{\Pi,\mathcal{A}}^{\mathsf{MBS}}(\lambda)$ is defined on the right-hand side in Figure 2.

Chalkias et al. [CGN20] call MBS signatures *binding signatures* and define *strongly binding signatures* as the conjunction of the MBS and M-S-UEO notions from [BCJZ20].

$$\mathbf{Exp}_{\Pi,\mathcal{A}}^{\mathsf{S\text{-}CEO}}(\lambda):$$

11 : $\quad \mathcal{Q} \leftarrow \emptyset$

12 : $\quad (\mathsf{sk}, \mathsf{pk}) \leftarrow\!\!\$\, \mathsf{KGen}(1^\lambda)$

13 : $\quad (m', \sigma', \mathsf{pk}') \leftarrow\!\!\$\, \mathcal{A}^{\mathsf{Sig}(\mathsf{sk}, \cdot)}(\mathsf{pk})$

14 : $\quad d \leftarrow \mathsf{Vf}(m', \sigma', \mathsf{pk}')$

15 : $\quad \mathbf{return} \ \big[d = 1 \ \wedge \ (m', \sigma') \in \mathcal{Q} \ \wedge \ \mathsf{pk}' \neq \mathsf{pk}\big]$

$$\mathsf{Sig}(\mathsf{sk}, m):$$

21 : $\quad \sigma \leftarrow\!\!\$\, \mathsf{Sig}(\mathsf{sk}, m)$

22 : $\quad \mathcal{Q} \leftarrow \mathcal{Q} \cup \{(m, \sigma)\}$

23 : $\quad \mathbf{return} \ \sigma$

$$\mathbf{Exp}_{\Pi,\mathcal{A}}^{\mathsf{S\text{-}DEO}}(\lambda):$$

31 : $\quad \mathcal{Q} \leftarrow \emptyset$

32 : $\quad (\mathsf{sk}, \mathsf{pk}) \leftarrow\!\!\$\, \mathsf{KGen}(1^\lambda)$

33 : $\quad (m', \sigma', \mathsf{pk}') \leftarrow\!\!\$\, \mathcal{A}^{\mathsf{Sig}(\mathsf{sk}, \cdot)}(\mathsf{pk})$

34 : $\quad d \leftarrow \mathsf{Vf}(m', \sigma', \mathsf{pk}')$

35 : $\quad \mathbf{return} \ \big[d = 1 \ \wedge \ (\exists m^* \neq m' : (m^*, \sigma') \in \mathcal{Q}) \ \wedge \ \mathsf{pk}' \neq \mathsf{pk}\big]$

Figure 3: Definition of the experiments $\mathbf{Exp}_{\Pi,\mathcal{A}}^{\mathsf{S\text{-}CEO}}(\lambda)$ and $\mathbf{Exp}_{\Pi,\mathcal{A}}^{\mathsf{S\text{-}DEO}}(\lambda)$ from Definitions 4.1 and 4.2, respectively with access to the same signing oracle.

4 New Theoretical Results

In this section, we present our main new theoretical results, which apply to the classical as well as the post-quantum setting. In Section 4.1 we introduce two analogous notions of exclusive ownership. In Section 4.2 we provide the first formal security definition for non re-signability. We establish relations among the security properties in Section 4.3, before giving a generic transformation that efficiently and provably achieves our security properties beyond unforgeability in Section 4.4.

4.1 New Notions of Exclusive Ownership

Brendel et al. [BCJZ20] introduced a strong variant of universal exclusive ownership, for which the attacker is *not* required to output the corresponding secret key of the new key pair. Analogously, we introduce two notions called strong conservative exclusive ownership and strong destructive exclusive ownership, where the attacker is only required to output a new public key and is additionally equipped with a signing oracle that it can query adaptively.

Strong Conservative Exclusive Ownership (S-CEO). In the security experiment, the adversary is only given a legitimate public key pk and additionally access to a signature oracle such that it can adaptively obtain arbitrary signatures for messages of its choice. The adversary is now asked to output a triple containing a message m', a signature σ', and a new public key pk'. It wins the game if the signature correctly verifies under pk', the pair (m', σ') has been queried to the oracle, and pk' differs from pk.

Definition 4.1. *Let Π be a digital signature scheme. We say that Π provides strong conservative exclusive ownership (S-CEO) if, for every PPT algorithm $\mathcal{A}$, there exists a negligible function $\mu \colon \mathbb{N} \to \mathbb{R}$ such that, for every $\lambda \in \mathbb{N}$, it holds that*

$$\Pr[\mathbf{Exp}_{\Pi,\mathcal{A}}^{\mathsf{S\text{-}CEO}}(\lambda)] \leq \mu(\lambda),$$

where $\mathbf{Exp}_{\Pi,\mathcal{A}}^{\mathsf{S\text{-}CEO}}(\lambda)$ is defined in Figure 3.

$$
\begin{array}{ll}
\textbf{Exp}^{\mathsf{NR}}_{\Pi,\mathcal{A},\mathcal{D}}(\lambda): & \\
\hline
11: & (\mathsf{sk},\mathsf{pk}) \leftarrow\!\!\$\, \mathsf{KGen}(1^\lambda) \\
12: & (m,\mathsf{aux}) \leftarrow\!\!\$\, \mathcal{D}(1^\lambda,\mathsf{pk}) \\
13: & \sigma \leftarrow\!\!\$\, \mathsf{Sig}(\mathsf{sk},m) \\
14: & (\sigma',\mathsf{pk}') \leftarrow\!\!\$\, \mathcal{A}(\mathsf{pk},\sigma,\mathsf{aux}) \\
15: & d \leftarrow \mathsf{Vf}(\mathsf{pk}',m,\sigma') \\
16: & \textbf{return } \big[d=1 \,\wedge\, \mathsf{pk}' \neq \mathsf{pk}\big]
\end{array}
$$

Figure 4: Definition of the experiment $\textbf{Exp}^{\mathsf{NR}}_{\Pi,\mathcal{A}}(\lambda)$ from Definition 4.3.

Strong Destructive Exclusive Ownership (S-DEO). In the security experiment, the adversary is given a public key pk and after querying the signing oracle, it outputs a triple containing a message m', a signature σ' and a new public key pk'. The adversary wins the game if the provided signature σ' was returned by the oracle for a message $m^* \neq m'$, pk' differs from pk, and the signature verifies for m' under pk'.

Definition 4.2. *Let Π be a digital signature scheme. We say that Π provides* strong destructive exclusive ownership *(S-DEO) if, for every* PPT *algorithm $\mathcal{A}$, there exists a negligible function $\mu\colon \mathbb{N} \to \mathbb{R}$ such that, for every $\lambda \in \mathbb{N}$, it holds that*

$$\Pr[\textbf{Exp}^{\mathsf{S\text{-}DEO}}_{\Pi,\mathcal{A}}(\lambda)] \leq \mu(\lambda),$$

where $\textbf{Exp}^{\mathsf{S\text{-}DEO}}_{\Pi,\mathcal{A}}(\lambda)$ is defined in Figure 3.

Note that one can also combine both strong variants from above to obtain the notion of strong univerisal exclusive ownership as introduced in [BCJZ20]. We formally describe the combined notion in Appendix A.2.

Throughout the rest of the paper we will analyze schemes with respect to these strong notions.

4.2 Non Re-signability

Jackson et al. [JCCS19] observed that for some signature schemes, an adversary that obtains the signature of a message m can produce another signature that verifies m under its own key without knowing m. For example, this can happen when the scheme reveals the hash of the message, which then enables re-signing this message with a different key. This runs contrary to the intuition that to produce a signature on a message, one should know the message. Jackson et al. coined this notion *non re-signability* (NR) and gave a symbolic model for the Tamarin prover. However, they did not provide a formal cryptographic definition, which is required to prove that a given signature scheme satisifies NR. We close this gap by providing the first security experiment for non re-signability.

Intuitively, the property non re-signability states that the adversary cannot produce a legitimate signature verifying under its public key for a message it does not know. The game in Figure 4 formalizes this notion. In more detail, after generating a key pair, the game runs a PPT distribution $\mathcal{D}$ that outputs a message m along with some auxiliary information aux about the message. One can think of the auxiliary information as being some structural information about the message. The game continues with generating the signature σ on m, and the adversary is then given the legitimate public key pk, the signature σ, as well as the auxiliary information. The adversary is now tasked to output a pair containing a signature σ' and a new public key pk'. It wins the game if both public keys do not coincide and the signature σ' verifies m under pk'.

Note that we assume that the message output by the distribution $\mathcal{D}$ is unpredictable by requiring the conditional (HILL) min-entropy to be strictly greater than logarithmic in the security parameter. Without this, the adversary could predict the underlying message m from the signature and trivially re-sign the message under the new key.

Definition 4.3. *Let Π be a digital signature scheme. We say that Π is* non-resignable *(NR) if, for every* PPT *algorithms $\mathcal{A}$ and $\mathcal{D}$, there exists a negligible function $\mu\colon \mathbb{N} \to \mathbb{R}$ such that, for every $\lambda \in \mathbb{N}$, it holds that*

$$\Pr[\mathbf{Exp}^{\mathsf{NR}}_{\Pi,\mathcal{A},\mathcal{D}}(\lambda)] \leq \mu(\lambda),$$

where $\mathbf{Exp}^{\mathsf{NR}}_{\Pi,\mathcal{A},\mathcal{D}}(\lambda)$ is defined in Figure 4. It is required that the PPT *algorithm $\mathcal{D}$ outputs a pair (m, aux) such that the conditional min-entropy $\tilde{\mathsf{H}}^{HILL}_{\infty}(m|\mathsf{aux}) \in \omega(\log \lambda)$.*

4.3 Relationship

Being equipped with these security properties beyond unforgeability, we are now in the position to establish that all properties are independent in the sense that there are schemes which may have all properties except for a particular one. This holds for each property from $\mathsf{M\text{-}S\text{-}UEO}, \mathsf{S\text{-}CEO}, \mathsf{S\text{-}DEO}, \mathsf{MBS}, \mathsf{NR}$ and $\mathsf{EUF\text{-}CMA}$. In the following we exemplify the separations only for the $\mathsf{S\text{-}CEO}$ property. The remaining relationships and respective proofs can be found in Appendix B. Note that we there also prove that $\mathsf{S\text{-}CEO}$ and $\mathsf{S\text{-}DEO}$ are equivalent to $\mathsf{S\text{-}UEO}$ and that $\mathsf{M\text{-}S\text{-}UEO}$ implies $\mathsf{S\text{-}UEO}$ (and, hence, $\mathsf{S\text{-}CEO}$ and $\mathsf{S\text{-}DEO}$).

Proposition 4.4. *If there is a digital signature scheme which has properties $\mathcal{P} \subseteq \{\mathsf{EUF\text{-}CMA}, \mathsf{S\text{-}DEO}, \mathsf{NR}, \mathsf{MBS}\}$, then there is also one which has the same properties $\mathcal{P}$ but not $\mathsf{S\text{-}CEO}$.*

Note that since $\mathsf{M\text{-}S\text{-}UEO}$ implies $\mathsf{S\text{-}CEO}$ it follows that the derived scheme cannot have $\mathsf{M\text{-}S\text{-}UEO}$, and we thus also exclude this property from $\mathcal{P}$.

Proof. Modify the scheme $\Pi = (\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$ with properties $\mathcal{P}$ to scheme $\Pi^{\neg\mathsf{S\text{-}CEO}}$ by introducing an exceptional signing and verification step for message $m = 0$ and public keys of the form $\mathsf{pk}\|0$ (which the genuine key generation algorithm never outputs):

$\Pi^{\neg\mathsf{S\text{-}CEO}}.\mathsf{KGen}(1^\lambda)$:	$\Pi^{\neg\mathsf{S\text{-}CEO}}.\mathsf{Sig}(\mathsf{sk}, m)$:	$\Pi^{\neg\mathsf{S\text{-}CEO}}.\mathsf{Vf}(\mathsf{pk}\|b, m, \sigma\|c)$:
$11:\quad (\mathsf{sk}, \mathsf{pk}) \leftarrow\!\!{\scriptstyle\$}\, \Pi.\mathsf{KGen}(1^\lambda)$	$21:\quad \sigma \leftarrow\!\!{\scriptstyle\$}\, \Pi.\mathsf{Sig}(\mathsf{sk}, m)$	$31:\quad \textbf{if } b = 0 \textbf{ then}$
$12:\quad \textbf{return } (\mathsf{sk}, \mathsf{pk}\|1)$	$22:\quad \textbf{if } m = 0 \textbf{ then}$	$32:\quad\quad \textbf{return } [c = 0 \wedge m = 0]$
	$23:\quad\quad \textbf{return } \sigma\|0$	$33:\quad \textbf{else}$
	$24:\quad \textbf{else}$	$34:\quad\quad d \leftarrow \Pi.\mathsf{Vf}(m, \sigma, \mathsf{pk})$
	$25:\quad\quad \textbf{return } \sigma\|1$	$35:\quad\quad \textbf{return } d$

The scheme inherits correctness of the original scheme.

To break property $\mathsf{S\text{-}CEO}$ it suffices to request a signature $\sigma\|0$ for message $m = 0$ under given key $\mathsf{pk}\|1$, and to output this message-signature pair with key $\mathsf{pk}\|0$. This constitutes a valid forgery against $\mathsf{S\text{-}CEO}$ since the pair has been signed but is also accepted under the new key $\mathsf{pk}\|0$ ending with 0.

We need to argue that the scheme $\Pi^{\neg\mathsf{S\text{-}CEO}}$ preserves the property $\mathsf{S\text{-}DEO}$. Assume that the adversary against DEO of the modified scheme attempts $\mathsf{pk}'\|0$ in the final output. Then the only message that is accepted under this key is $m' = 0$, but then any distinct query $m \neq 0$ to the signing oracle causes the signature $\sigma\|1$ to end in 1, such that these signatures cannot be valid for $m' = 0$. If, on the other hand, the adversary uses $\mathsf{pk}'\|1$ in its attempt then we must have $\mathsf{pk}' \neq \mathsf{pk}$ and there was a query m to the signer which created the signature. In particular, the actual signature part (without the trailing bit) must match for this query and still $m \neq m'$. We then construct a black-box reduction to the $\mathsf{S\text{-}DEO}$ property of the

underlying scheme, by letting the reduction append (for signature queries) and chop off (for the forgery) the additional bits.

Next, it is easy to show that the scheme preserves the property EUF-CMA because any forgery would have to be against honestly generated public keys ending with 1, such that the exceptional step in verification cannot be triggered. Adding and removing the extra bits of the public key and the signature gives the desired security reduction to the property of the original scheme.

As for MBS note that, if the adversary chooses $\mathsf{pk}\|0$ then only one message, namely $m = 0$, is accepted at all. Hence to find distinct $m_1 \neq m_2$ with valid signature $\sigma\|c$ under some public key, the key must be of the form $\mathsf{pk}\|1$. But then m_1, m_2 together with σ and pk constitute a valid MBS-attack against the original scheme.

It remains to argue that the modified scheme preserves property NR. To see this note that $\mathcal{D}$ must have super-logarithmic min-entropy such that the probability that $m = 0$ is negligible. This means that with overwhelming probability the adversary cannot use a key of the form $\mathsf{pk}\|0$ to win. In any other case it is again immediate to reduce an attack against the modified scheme to an attack against the starting scheme. $\qquad\square$

4.4 BUFF transformation: A generic transformation for provably achieving M-S-UEO, MBS, and NR

We construct a generic transformation that ensures that the resulting signature scheme achieves Beyond UnForgeability Features (i.e., M-S-UEO, MBS, and NR): The BUFF transformation. This transformation works for both the classical and the post-quantum setting. Before we present the details, we first revisit known transformations for some individual properties. Pornin and Stern [PS05] provided three transformations to add the notions of exclusive ownership to a signature scheme. Two of their transformations make use of a collision resistant hash function and also increase the signature size, while the third one does not increase the signature size but requires a random oracle; none of them achieves NR. While Pornin and Stern prove that their transformations achieve their "weak" variants of exclusive ownership (CEO and DEO), we argue their proofs translate to the strong notions (as formalized in Section 4.1) in a straightforward manner. We briefly summarize those transformations and their guarantees.

Pornin and Stern transformation 1. Their first transformation is designed to add DEO to a signature scheme. Starting from a signature scheme $\Pi = (\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$ and transforming it into a new signature scheme $\Pi^* = (\mathsf{KGen}^*, \mathsf{Sig}^*, \mathsf{Vf}^*)$ where KGen^* is equal to KGen. For any message m the signature is derived by appending a hash *of the message*, i.e., $\mathsf{Sig}^*(\mathsf{sk}, m) = (\mathsf{Sig}(\mathsf{sk}, m), \mathsf{H}(m))$. For any signature of the form $\sigma^* = (\sigma, y)$ the verification algorithm Vf^* simply accepts the signature if σ is accepted by Vf and $y = \mathsf{H}(m)$. Assuming that the hash function is collision resistant, this ensures that each signature is exclusive to the message that was signed and thus provides DEO as well as S-DEO.

Observe that this transformation achieves MBS: the transformation binds the message through the hash function evaluation to the signature, and hence (due to the collision resistance of the hash function) the adversary is prevented from outputting a second message that the signature also verifies for. However, this transformation does not provide CEO because the signature is not necessarily exclusive to the public key. NR is in general not achieved since the signature of the original scheme σ may contain the message directly, allowing the adversary to re-sign this message under a new key.

Pornin and Stern transformation 2. The second transformation adds both CEO and DEO (and also the strong variants) to any signature scheme. The construction itself works similar to the previous one with the difference that one appends the hash *of the public key* to the signature, i.e., $\mathsf{Sig}^*(\mathsf{sk}, m) = (\mathsf{Sig}(\mathsf{sk}, m), \mathsf{H}(\mathsf{pk}))$, and verifies this hash explicitly during verification. Again by relying on the collision resistance of the hash function the scheme provides M-S-UEO since the signature cannot be reused with any *other* public key.

However, this transformation does neither achieve MBS nor NR: MBS is not guaranteed because the

signature is not bound to the message that was signed and hence the transformation cannot prevent the attacker from outputting two different messages which both verify for the same signature. It does not provide NR for the same reason as the first transformation.

Pornin and Stern transformation 3. The third transformation adds CEO and DEO to any signature scheme *without* expanding the signature size. This requires a specific property, namely resistance to existential forgeries for *all* possible keys, i.e., also the possibly weak and incorrect keys the adversary might use. Assuming this property, the transformation *derives the signature from the hash function evaluation of the message concatenated with the public key* instead of the plain message, i.e., $\mathsf{Sig}^*(\mathsf{sk}, m) = \mathsf{Sig}(\mathsf{sk}, \mathsf{H}(m, \mathsf{pk}))$. Pornin and Stern provide a proof in the random oracle model assuming the above property showing that it achieves CEO and DEO. Note that a similar transformation was previously proposed by Menezes and Smart [MS04], who prepended the message with the public key in an unambiguous way to achieve a security notion that is equivalent to CEO. We expect that this transformation also achieves S-CEO and S-DEO with a similar argument under the same assumption. Without assuming the above mentioned property, the transformation achieves none of the five security properties, since a signature scheme may have a public key under which the verify algorithm unconditionally accepts.

Table 2: Comparing transformations and known results if weak keys may be possible. ✓ indicates that a property holds and ✗ indicates an attack. A property is marked with (✓) if we know that it holds if there are no weak keys.

Transform.	Signature	S-CEO	S-DEO	M-S-UEO	MBS	NR
[PS05]-1	$\mathsf{Sig}(\mathsf{sk}, m), \mathsf{H}(m)$	✗	✓	✗	✓	✗
[PS05]-2	$\mathsf{Sig}(\mathsf{sk}, m), \mathsf{H}(\mathsf{pk})$	✓	✓	✓	✗	✗
[PS05]-3	$\mathsf{Sig}(\mathsf{sk}, \mathsf{H}(m, \mathsf{pk}))$	✗ (✓)	✗ (✓)	✗	✗	✗
BUFF	$\mathsf{Sig}(\mathsf{sk}, \mathsf{H}(m, \mathsf{pk})), \mathsf{H}(m, \mathsf{pk})$	✓	✓	✓	✓	✓

The BUFF transformation We propose a transformation that simultaneously adds all five properties (S-CEO, S-DEO, M-S-UEO, MBS and NR) and only relies on standard properties of the hash function. Our BUFF transformation builds on transformation 3, but adds the computed hash of the signed data to the resulting signature similar to transformation 1. Out of the many possible variants, it turns out that this particular combination provides protection against weak keys and achieves message-bound signatures and non re-signability. Similar to transformations 1 and 2, the signature size is increased by the output size of the hash function, but we show in Figure 6 that for the NIST round 3 schemes the relative size increase is typically negligible.

The formal details of the BUFF transformation are given in Figure 5. We start from a signature scheme $\Pi = (\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$ and transform it into a new signature scheme $\Pi^* = (\mathsf{KGen}^*, \mathsf{Sig}^*, \mathsf{Vf}^*)$ where KGen^* is equal to KGen. We derive the signature for any message m as $\mathsf{Sig}^*(\mathsf{sk}, m) = (\mathsf{Sig}(\mathsf{sk}, \mathsf{H}(m, \mathsf{pk})), \mathsf{H}(m, \mathsf{pk}))$. For any signature of the form $\sigma^* = (\hat{\sigma}, \hat{h})$ the verification algorithm Vf^* simply accepts the signature if $\hat{h} = \mathsf{H}(m, \mathsf{pk})$ and $\hat{\sigma}$ is accepted by Vf for the message $\mathsf{H}(m, \mathsf{pk})$.

Our design follows the argument order of previous transformations, but the order does not play a role in the proof. We added the hash to the signature (increasing its size) to enable a generic proof for all properties that is independent of the underlying signature scheme details. However, it is known that at least for some schemes (e.g., [BCJZ20]) the same properties can be achieved without increasing the signature size by performing appropriate checks on the public keys and providing a scheme-specific security analysis. However, we do not know of a generic way to achieve this.

Jumping ahead, we note that in some schemes a hash value with the same inputs already appears as part of the signature. Specifically, for Fiat-Shamir signatures the hash value usually appears in the

KGen*(1^λ):	Sig*(sk, m):	Vf*$(\mathsf{pk}, m, \sigma^*)$:
11 : $(\mathsf{sk}, \mathsf{pk}) \leftarrow\!\!\$\, \mathsf{KGen}(1^\lambda)$	21 : $h \leftarrow \mathsf{H}(m, \mathsf{pk})$	31 : $(\hat{\sigma}, \hat{h}) \leftarrow \sigma^*$
12 : **return** $(\mathsf{sk}, \mathsf{pk})$	22 : $\sigma \leftarrow\!\!\$\, \mathsf{Sig}(\mathsf{sk}, h)$	32 : $h \leftarrow \mathsf{H}(m, \mathsf{pk})$
	23 : $\sigma^* \leftarrow (\sigma, h)$	33 : $d \leftarrow \mathsf{Vf}(\mathsf{pk}, h, \hat{\sigma})$
	24 : **return** σ^*	34 : **return** $\left[d = 1 \,\wedge\, \hat{h} = h \right]$

Figure 5: The BUFF (Beyond UnForgeability Features) transformation, which turns any EUF-CMA-secure signature scheme Π into an EUF-CMA-secure scheme Π^* that also achieves M-S-UEO, MBS, and NR, even in the presence of weak keys.

signatures. In this case the transformation does not even require a hash function invocation nor does it bear the size penalty.

Theorem 4.5. *Let Π be an EUF-CMA-secure signature scheme. Then the application of the BUFF transformation in Figure 5 produces an EUF-CMA-secure signature scheme Π^* that additionally also provides the properties of M-S-UEO, MBS and NR assuming that the hash function H is collision resistant and Φ-non-malleable where $\Phi = \{\phi_{\mathsf{pk}'} | \mathsf{pk}' \in \mathcal{K}\}$ and $\phi_{\mathsf{pk}'}(m, \mathsf{pk}) = (m, \mathsf{pk}')$.*

Because the public key part pk in the input to $\phi_{\mathsf{pk}'}$ is known, we can rewrite the functions $\phi_{\mathsf{pk}'}$ as $\phi'_\delta(m, \mathsf{pk}) = (m, \delta \oplus \mathsf{pk})$ for $\delta = \mathsf{pk} \oplus \mathsf{pk}'$ if the key length is fixed, leaving the message part untouched. Technically we therefore require $\oplus$-non-malleability which is known to hold for example for Merkle–Damgård constructions with ideal round functions [BFS11], and with the same argument can be easily seen to hold also for Sponge-based constructions with ideal permutations. As such, the deployed hash functions in the signature schemes considered here, namely, SHAKE-256 (Dilithium, FALCON, Picnic, SPHINCS$^+$), SHA-2 (Rainbow, SPHINCS$^+$), and SHA-3 (GeMSS) should be considered to provide non-malleability in the above sense.

We note that Dilithium and Picnic, the two schemes which already include a hash value in their signatures, slightly deviate from the hash input pattern in the theorem and require a different class $\Phi = \{\phi_{\mathsf{pk}', \psi}\}$ for non-malleability. Dilithium uses $(\mathsf{pk}, m, \mathbf{w_1})$ as the input to the hash function where $\mathbf{w_1}$ is part of the signature and which can thus potentially be modified by the adversary via some function ψ, such that the operation is of the form $\phi_{\mathsf{pk}', \psi}(\mathsf{pk}, m, x) = (\mathsf{pk}', m, \psi(x))$. We note that iterated hash functions with ideal round functions still obey this form of non-malleability where one needs to modify the fixed-size public key, and the transformation theorem holds for this case as well. This is also true for Picnic where the hash input (a, pk, m) starts with a circuit description a which could be potentially mauled by the adversary to $a' = \psi(a)$.

We provide some intuition why BUFF indeed achieves the discussed properties. Intuitively, we achieve the exclusive ownership properties by assuming the hash function to be collision resistant which ensures that the signature is exclusive to the public key that was used to generate it. Similarly, the transformation provides message-bound signatures since the hash function is collision resistant and hence the attacker cannot output two different messages that the signature both verifies. Intuitively, the signature of the original scheme may leak at most the hash digest of the message bound to the public key and not the message itself. To formally reduce NR to Φ-non-malleability we rely on the explicitly appended hash digest.

Let us split the proof into smaller components such that the collection of these results yields a proof for Theorem 4.5.

Lemma 4.6. *Let* Π *be an* EUF-CMA-*secure signature scheme. Then the application of the BUFF transformation given in Figure 5 produces an* EUF-CMA-*secure signature scheme* Π^* *assuming that the hash function* H *is collision resistant.*

Proof. A successful attacker $\mathcal{A}$ against EUF-CMA-security of signature scheme Π^* can be used to construct a successful attacker $\mathcal{B}$ against EUF-CMA-security of the underlying signature scheme Π. The outer attacker $\mathcal{B}$ provides its own input to $\mathcal{A}$. It simulates the signing oracle for $\mathcal{A}$ by forwarding the hash evaluation of the public key and the message as query to its own oracle and appending the same hash digest to the signature returned from the oracle. The outer attacker $\mathcal{B}$ takes the output (m', σ^*) of $\mathcal{A}$ where σ^* is of the form (σ', h'). The adversary $\mathcal{B}$ simply parses σ^* accordingly and outputs as its forgery (h', σ'). As $\mathcal{A}$ is successful, $\mathcal{B}$ is also successful, unless h' collides with a hash value in the signature queries, contradicting the collision resistance of H. $\qquad\square$

Lemma 4.7. *Let* Π *be an* EUF-CMA-*secure signature scheme. Then the application of the BUFF transformation given in Figure 5 produces a signature scheme* Π^* *that provides* M-S-UEO *assuming that the hash function* H *is collision resistant.*

Proof. Let us assume a successful attacker against M-S-UEO of Π^* that outputs $(m_1, m_2, \sigma, \mathsf{pk}_1, \mathsf{pk}_2)$. Since $\mathsf{Vf}^*(\mathsf{pk}_1, m_1, \sigma)$ and $\mathsf{Vf}^*(\mathsf{pk}_2, m_2, \sigma)$ both yield true, it must hold that $\mathsf{H}(m_1, \mathsf{pk}_1) = h = \mathsf{H}(m_2, \mathsf{pk}_2)$ where $\mathsf{pk}_1 \neq \mathsf{pk}_2$. Therefore, the attacker has found a collision in H. Since H is collision resistant, this only happens with negligible probability. Thus, the probability of this attacker succeeding is negligible as well. $\qquad\square$

By applying the Propositions B.1 and B.2, it follows that the signature scheme Π^* also provides S-CEO and S-DEO.

Lemma 4.8. *Let* Π *be an* EUF-CMA-*secure signature scheme. Then the application of the BUFF transformation given in Figure 5 produces a signature scheme* Π^* *that provides* MBS *assuming that the hash function* H *is collision resistant.*

Proof. Let us assume a successful attacker against MBS of Π^* that outputs $(m_1, m_2, \sigma, \mathsf{pk})$. Since both evaluations of $\mathsf{Vf}^*(\mathsf{pk}, m_1, \sigma)$ and $\mathsf{Vf}^*(\mathsf{pk}, m_2, \sigma)$ yield true, it must hold that $\mathsf{H}(m_1, \mathsf{pk}) = h = \mathsf{H}(m_2, \mathsf{pk})$ while $m_1 \neq m_2$. Therefore, the attacker has found a collision in H. Since H is collision resistant, this can only happen with negligible probability. Thus, the probability of this attacker succeeding is negligible. $\qquad\square$

Lemma 4.9. *Let* Π *be an* EUF-CMA-*secure signature scheme. Then the application of the BUFF transformation given in Figure 5 produces a signature scheme* Π^* *that provides* NR *assuming that the hash function* H *is* Φ-*non-malleable where* $\Phi = \{\phi_{\mathsf{pk}'} | \mathsf{pk}' \in \mathcal{K}\}$ *and* $\phi_{\mathsf{pk}'}(m, \mathsf{pk}) = (m, \mathsf{pk}')$.

Proof. In this proof we show that the signature scheme Π^* obtained from transforming Π according to Figure 5 achieves non re-signability, assuming that the hash function is Φ-non-malleable for $\Phi = \{\phi_{\mathsf{pk}'}\}$ and $\phi_{\mathsf{pk}'}(m, \mathsf{pk}) = (m, \mathsf{pk}')$.

We start with assuming a successful attacker pair $(\mathcal{A}, \mathcal{D})$ against NR of Π^*. We construct an efficient reduction $\mathcal{B} = (\mathcal{B}_d, \mathcal{B}_y)$ against the Φ-non-malleability of the hash function H running $\mathcal{A}$ and $\mathcal{D}$ as a sub-routine. The adversary $\mathcal{B}_d$ upon receiving the hash key hk starts with initializing the parameters for the NR game. It computes the signing key pair which is then coded into the state information st which will be passed to the second stage. Further given the distribution $\mathcal{D}$ algorithm $\mathcal{B}_d$ creates (the description of) a new distribution $\mathcal{X}$ that works as $\mathcal{D}$ with the only difference that each sampled message of this distribution gets the public key pk appended. Note that the distribution $\mathcal{X}$ is required to be non-trivial by demanding sufficient min-entropy. This is simply ensured by the fact that the underlying distribution $\mathcal{D}$ is by definition unpredictable since its min-entropy grows strictly faster than logarithmic in the security parameter.

16

The challenger for $\mathcal{B}$ now samples a message from $\mathcal{X}$ of the form (m, pk) as well as some auxiliary information aux about the message part (which is captured in the Φ-non-malleability game through the hint function). Next, the challenger evaluates the hash function H on input (m, pk) obtaining the digest h and provides the second-stage adversary $\mathcal{B}_y$ with the input $(h, \mathsf{aux}, \mathsf{st})$. The adversary $\mathcal{B}_y$ begins with parsing the state information st obtaining the initial key pair. Next, it uses the secret key to sign the hash digest obtaining the signature σ. Then, it prepares the final signature σ^* as (h, σ). The adversary $\mathcal{A}$ receives $(\mathsf{pk}, \mathsf{aux}, \sigma^*)$ and outputs (σ', pk') where σ' has the form $(\tilde{h}, \tilde{\sigma})$ and $\mathsf{pk}' \neq \mathsf{pk}$. Then $\mathcal{B}_y$ parses the signature σ' and defines a function $\phi_{\mathsf{pk}'}$ with $\phi_{\mathsf{pk}'}(m, \mathsf{pk}) = (m, \mathsf{pk}')$. Finally it outputs $(\tilde{h}, \phi_{\mathsf{pk}'})$.

We observe that $\mathcal{B}$ has faithfully simulated the NR game and since $\mathcal{A}$ was successful then also $\mathcal{B}$ is successful. This is true since σ' is a valid signature on $\tilde{h} = \mathsf{H}(m, \mathsf{pk}')$ which in turn equals $\mathsf{H}(\phi_{\mathsf{pk}'}(m, \mathsf{pk}))$ and hence the first part of the winning condition of $\mathcal{B}$ is fulfilled. The second condition, namely $\phi_{\mathsf{pk}'}(x) \neq x$, is also satisfied with $x = (m, \mathsf{pk})$ and $\phi_{\mathsf{pk}'}(m, \mathsf{pk}) = (m, \mathsf{pk}') \neq (m, \mathsf{pk})$ due to $\mathsf{pk}' \neq \mathsf{pk}$. Hence the attacker has successfully mauled the input of the hash function. However this contradicts our assumption that H is Φ-non-malleable and therefore such an adversary cannot exist. $\qquad\square$

Collecting the above results proves Theorem 4.5.

5 Analyzing NIST's Round 3 Signature Schemes

In this section, we analyze the six signature schemes submitted to round 3 of NIST's call to standardize quantum-resistant schemes [Nat15b]. Our goal is to check whether these signature schemes achieve the security properties beyond unforgeability as presented in Sections 3 and 4. We first expand on the three finalists CRYSTALS-Dilithium [BDK+21], FALCON [FHK+20], and Rainbow [DCK+20], followed by an analysis of the three alternate candidates Picnic [CDG+20, Zav20], GeMSS [CFMR+20], and SPHINCS+ [ABB+20].

Anticipating our results, we prove that all three properties hold for Dilithium and Picnic, and we show that some properties do not hold for FALCON, Rainbow, and GeMSS. We provide an overview of our results in Table 1. We detail the cost of provably achieving all three properties beyond unforgeability with our transformation from Figure 5 in terms of signature size in Figure 6.

5.1 CRYSTALS-Dilithium

Dilithium [BDK+21] is a lattice-based signature scheme whose security is based on the hardness of the Learning with Errors (LWE) problem and a variant of the shortest integer solution (SIS) problem, and employs Fiat-Shamir with Aborts [Lyu09]. Figure 7 gives an algorithmic description of Dilithium.

In the following we provide a short description of Dilithium. In order to derive the key pair, the key generation algorithm starts with generating an initial string that is given as an input to an extendable output function (XOF) H generating initial strings (ρ, ς, K). Inputting ς to H generates two short vectors $\mathbf{s_1}, \mathbf{s_2}$ and a matrix $\mathbf{A}$ is derived from $\mathsf{ExpandA}(\rho)$. It computes $\mathbf{t} = \mathbf{As_1} + \mathbf{s_2}$ and splits it into its high bits $\mathbf{t_1}$ and low bits $\mathbf{t_0}$ with the functions $\mathsf{HighBits}$ and $\mathsf{LowBits}$, respectively. Furthermore, it evaluates a collision-resistant hash function on the public key outputting a string tr. Finally, the algorithm outputs the keys $\mathsf{pk} = (\rho, \mathbf{t_1})$ and $\mathsf{sk} = (K, tr, \mathbf{t_0}, \mathbf{s_1}, \mathbf{s_2}, \rho)$. To sign a message m, the signing algorithm generates a short vector $\mathbf{y}$ from intermediate values. It then computes the challenge seed $\tilde{c} \leftarrow \mathsf{H}'(\mathsf{pk}, m, \mathsf{HighBits}(\mathbf{Ay}))$ where $\mathsf{H}' = \mathsf{H} \circ \mathsf{CRH} \circ \mathsf{CRH}$ with both H and CRH being collision resistant, a challenge $\mathbf{c} \leftarrow \mathsf{SampleInBall}(\tilde{c})$, and $\mathbf{z} \leftarrow \mathbf{y} + \mathbf{cs_1}$, where $\mathsf{SampleInBall}$ produces a short vector. If the resulting $\mathbf{z}$ is not short or $\mathsf{HighBits}(\mathbf{Ay}) \neq \mathsf{HighBits}(\mathbf{Az} - \mathbf{ct})$ then the algorithm continues with sampling a fresh random $\mathbf{y}$ and proceeds as before. Otherwise, the algorithm creates a short hint h (a dense presentation of high bits) and the signature then consists of $\sigma \leftarrow (\mathbf{z}, h, \tilde{c})$. The verification algorithm first parses the

Scheme	Current signature size (B)	Size after applying our transformation (if needed) (B)	Relative increase
Rainbow	212	276	30.0%
FALCON	1280	1344	5.0%
CRYSTALS-Dilithium	4595	4595	0.0%
GeMSS256	72	136	88.9%
SPHINCS$^+$-256s	29792	29856	0.2%
Picnic3-L5	61024	61024	0.0%

The impact of provably guaranteeing NR, S-CEO/S-DEO, and MBS on signature size
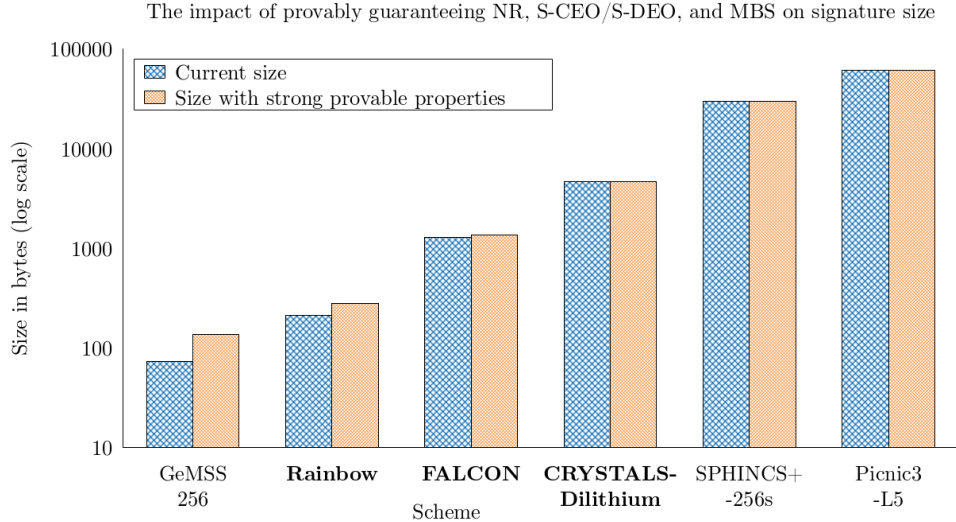
Figure 6: Provably achieving security properties beyond unforgeability for the NIST round 3 candidates: for candidates that do not provably offer these properties yet, our BUFF transformation slightly increases signature size. Since the additional size is constant (64 bytes), the largest relative increase occurs for the smallest signature size (e.g. GeMSS256 goes from 72 to 136 bytes); however, this not even impacts the relative ordering of candidates based on signature size. Since the BUFF transformation involves only a single hash, the additional computational cost is in all cases negligible compared to the signature generation and verification.

signature and recomputes the challenge $\mathbf{c} \leftarrow \mathsf{SampleInBall}(\tilde{c})$. It reconstructs the high bits of $\mathbf{Ay}$ with the help of the hint and uses this value to recompute the challenge seed. The signature is accepted if $\mathbf{z}$ is short, the recomputed challenge seed matches the challenge seed in the signature, and the hint is well-formed.

Proposition 5.1. *The signature scheme Dilithium as described in Figure 7 provides* M-S-UEO, MBS, *and* NR *if the hash function* H *is collision resistant and* Φ-*non-malleable for* $\Phi = \{\phi_{\mathsf{pk},\psi}\}$ *and* $\phi_{\mathsf{pk}',\psi}(\mathsf{pk}, m, \mathbf{w_1}) = (\mathsf{pk}', m, \psi(\mathbf{w_1}))$ *for any function* ψ.

As remarked earlier, compared to our Transformation Theorem 4.5, we need a slightly different version of non-malleability here where the hash input contains a part $\mathbf{w_1}$ of the signature at the end, which the adversary can modify as part of the new signature via function ψ. Our theorem still applies in this case, and in terms of constructions iterated hash functions with idealized round function obey this form of non-malleability, too.

KGen(1^λ)	Sig(sk, m)

KGen(1^λ)

11 : $\zeta \leftarrow\!\!\$\ \{0,1\}^{256}$

12 : $(\rho, \varsigma, K) \leftarrow \mathsf{H}(\zeta)$

13 : $(\mathbf{s_1}, \mathbf{s_2}) \leftarrow \mathsf{H}(\varsigma)$ // s_1, s_2 are short vectors

14 : $\mathbf{A} \leftarrow \mathsf{ExpandA}(\rho), \mathbf{t} \leftarrow (\mathbf{As_1} + \mathbf{s_2})$

15 : $(\mathbf{t_0}, \mathbf{t_1}) \leftarrow (\mathsf{LowBits}(\mathbf{t}), \mathsf{HighBits}(\mathbf{t}))$

16 : $tr \leftarrow \mathsf{CRH}(\rho, \mathbf{t_1})$

17 : $\mathsf{sk} \leftarrow (K, tr, \mathbf{s_1}, \mathbf{s_2}, \mathbf{t_0}, \rho), \mathsf{pk} \leftarrow (\rho, \mathbf{t_1})$

18 : **return** $(\mathsf{sk}, \mathsf{pk})$

Vf(pk, m, σ)

41 : $\mathbf{A} \leftarrow \mathsf{ExpandA}(\rho)$

42 : $\mu \leftarrow \mathsf{CRH}(\mathsf{CRH}(\rho, \mathbf{t_1}), m)$

43 : $\mathbf{c} \leftarrow \mathsf{SampleInBall}(\tilde{c})$

44 : $\mathbf{w_1'} \leftarrow \mathsf{UseHint}(h, \mathbf{Az} - \mathbf{ct})$

45 : **return** $[\mathbf{z} \text{ short} \wedge \tilde{c} = \mathsf{H}(\mu, \mathbf{w_1'}) \wedge h \text{ well-formed}]$

Sig(sk, m)

21 : $\mathbf{A} \leftarrow \mathsf{ExpandA}(\rho)$

22 : $\mu \leftarrow \mathsf{CRH}(tr, m), \rho' \leftarrow \mathsf{CRH}(K, \mu)$

23 : $\kappa \leftarrow 0, \mathbf{z} \leftarrow \perp$

24 : **while** $\mathbf{z} = \perp$

25 : $\quad \mathbf{y} \leftarrow\!\!\$\ \mathsf{ExpandMask}(\rho', \kappa)$

26 : $\quad \mathbf{w_1} \leftarrow \mathsf{HighBits}(\mathbf{Ay})$

27 : $\quad \tilde{c} \leftarrow \mathsf{H}(\mu, \mathbf{w_1})$

28 : $\quad \mathbf{c} \leftarrow \mathsf{SampleInBall}(\tilde{c})$

29 : $\quad \mathbf{z} \leftarrow \mathbf{y} + \mathbf{cs_1}$

30 : $\quad$ **if** $\mathbf{z}$ not short $\vee \mathbf{w_1} \neq \mathsf{HighBits}(\mathbf{Az} - \mathbf{ct})$ **then**

31 : $\quad\quad \mathbf{z} \leftarrow \perp$

32 : $\quad$ **else**

33 : $\quad\quad h \leftarrow \mathsf{MakeHint}(\mathbf{Ay}, \mathsf{sk})$

34 : $\quad \kappa \leftarrow \kappa + \dim(\mathbf{y})$

35 : $\sigma \leftarrow (\mathbf{z}, h, \tilde{c})$

36 : **return** σ

Figure 7: Algorithmic description of Dilithium based on Figure 4 in [BDK$^+$21].

Proof. By inspecting the details of Dilithium in Figure 7, we observe that the signature contains a hash digest that was generated from the public key and the message by evaluating H'. Note that H' is actually a composition of several hash functions, namely $\mathsf{H}' = \mathsf{H} \circ \mathsf{CRH} \circ \mathsf{CRH}$ where both H and CRH are collision resistant hash functions and in more detail the challenge seed is computed as $\tilde{c} \leftarrow \mathsf{H}(\mathsf{CRH}(\mathsf{CRH}(\mathsf{pk}), m), \mathbf{w_1})$. We further observe that this digest is explicitly checked by the verification algorithm. Hence, Dilithium implements our BUFF transformation as specified in Figure 5 and therefore Theorem 4.5 applies to Dilithium. From this we can conclude that Dilithium provides M-S-UEO and message-bound signatures by assuming H' to be collision resistant. Non re-signability directly follows by assuming H' to be collision resistant and Φ-non-malleable for $\Phi = \{\phi_{\mathsf{pk}', \psi}\}$ where $\phi_{\mathsf{pk}', \psi}(\mathsf{pk}, m, \mathbf{w_1}) = (\mathsf{pk}', m, \psi(\mathbf{w_1}))$ for any function ψ. $\qquad\qquad\square$

Note that the hash function CRH in Dilithium is SHAKE-256 truncated to 384 bits output, with (injective) bit-packing encoding tuples into bit strings. This means that any bit string inserted into the hash function CRH allows to recover the individual input components. The hash function H squeezes SHAKE-256 on its input and uses the outputs to generate a 256-bit element $\mathbf{c}$ in the ball B_{60} of vectors with exactly 60 entries from ± 1. The overall hash function is conceivably non-malleable for the aforementioned function class. The only way to create a valid hash value of a related key and the same (unknown) message for the adversary seems to require to compute $\mu' = \mathsf{CRH}(\mathsf{CRH}(\mathsf{pk}'), m)$, else $\tilde{c}' = \mathsf{H}(\mu', \mathbf{w_1'})$ would not most likely not hold in the final verification step for the adversary's signature. Indeed if we assume that finding $\tilde{c}'$ without knowing μ' is infeasible and model the round function of $\mathsf{CRH} = $ SHAKE-256 as a random permutation, then the adversary must iterate CRH on pk' and m to succeed with non-negligible probability, in which case the adversary must already know m, contradicting its super-logarithmic entropy.

KGen(1^λ)	Sig(sk, m)	Vf(pk, m, σ)
11 : $(f, g, F, G) \leftarrow\!\!\$\ \mathsf{NTRUGen}(\phi, q)$	21 : $r \leftarrow\!\!\$\ \{0,1\}^{320}$	31 : $(r, s) \leftarrow \sigma$
12 : $\mathbf{B} \leftarrow \begin{bmatrix} g & -f \\ G & -F \end{bmatrix}$	22 : $c \leftarrow \mathsf{H}(r, m)$	32 : $c \leftarrow \mathsf{H}(r, m)$
	23 : $\mathbf{t} \leftarrow (\mathsf{FFT}(c), \mathsf{FFT}(0)) \cdot \hat{\mathbf{B}}^{-1}$	33 : $s_2 \leftarrow \mathsf{Decompress}(s)$
13 : $\hat{\mathbf{B}} \leftarrow \mathsf{FFT}(\mathbf{B})$	24 : $\mathbf{s} \leftarrow\!\!\$\ \mathsf{FFSampling}(\mathbf{t}, T, \lfloor \beta^2 \rfloor)$	34 : $s_1 \leftarrow c - s_2 h$
14 : $T \leftarrow \mathsf{FalconTree}(\hat{\mathbf{B}})$	25 : $(s_1, s_2) \leftarrow \mathsf{FFT}^{-1}(\mathbf{s})$	35 : $\textbf{return}\ \ \left[\|(s_1, s_2)\|^2 \leq \lfloor \beta^2 \rfloor \right]$
15 : $\mathsf{sk} \leftarrow (\hat{\mathbf{B}}, T)$	26 : $s \leftarrow \mathsf{Compress}(s_2)$	
16 : $h \leftarrow g f^{-1}$	27 : $\sigma \leftarrow (r, s)$	
17 : $\mathsf{pk} \leftarrow h$	28 : $\textbf{return}\ \sigma$	
18 : $\textbf{return}\ (\mathsf{sk}, \mathsf{pk})$		

Figure 8: Algorithmic description of FALCON.

5.2 FALCON

The FALCON [FHK+20] scheme is a hash-and-sign lattice-based signature scheme based on the GPV framework [GPV08]. The proposed scheme uses the class of NTRU lattices and a new trapdoor sampler called Fast Fourier Sampler. The security of FALCON (short for <u>Fa</u>st Fourier <u>l</u>attice-based <u>compact</u> signatures over <u>NTRU</u>) is based on the shortest integer solution (SIS) problem.

In Figure 8 we provide an algorithmic description of FALCON. The key generation algorithm samples an NTRU lattice, obtains f, g, F, G solving it, and sets the matrix $\mathbf{B}$ based on the solution to the NTRU problem. Next, it computes the Fast Fourier Transform (FFT) representation of f, g, F, G obtaining a matrix $\hat{\mathbf{B}}$ and FALCON takes advantage of a new data structure called FALCON Tree from which one can sample the short vector $\mathbf{s}$ more efficiently. This tree T is computed based on $\hat{\mathbf{B}}$ and the secret key is set to $(\hat{\mathbf{B}}, T)$ while the public key is set to $h \leftarrow g f^{-1}$. Here, the modulus ϕ (and hence h) is a polynomial of maximal degree $n = 512$ (for FALCON-512) or $n = 1024$ (for FALCON-1024) over $\mathbb{Z}_q$, where $q = 12289$.

The signing algorithm samples a random salt r and hashes the salt and the message to the polynomial c. It computes a preimage $\mathbf{t}$ of c under $\hat{\mathbf{B}}$. Next, the algorithm uses Fast Fourier sampling to sample a short polynomial $\mathbf{s}$ followed by computing $(s_1, s_2) \leftarrow \mathsf{FFT}^{-1}(\mathbf{s})$ which satisfies $c = s_1 + s_2 h$ based on the preimage $\mathbf{t}$ and the FALCON Tree T for some bound β. The signature consists of the salt r and a compressed representation s of s_2.

The verification algorithm hashes the message and the salt r to c and decompresses s to s_2. Next, it computes $s_1 \leftarrow c - s_2 h$ and accepts the signature if $\|(s_1, s_2)\|^2 \leq \lfloor \beta^2 \rfloor$, i.e., if (s_1, s_2) is shorter than some bound β^2. The scheme only gives the square of β. The value β is approximately 5400 for FALCON-512 and 8400 for FALCON-1024.

We start by showing that FALCON has message-bound signatures, followed by the proof that it does *not* provide non re-signability, conservative exclusive ownership, and destructive exclusive ownership. For the proof of message-bound security we need the assumption that the hash function H is near-collision resistant, meaning that it is infeasible to find hash values which are close (but not necessarily equal):

Assumption 5.2 (Near-Collision Resistance of H). *Finding near collisions $(r, m_1) \neq (r, m_2)$ with $\|\mathsf{H}(r, m_1) - \mathsf{H}(r, m_2)\| \leq 2\beta$ for FALCON's hash function H and parameter β is infeasible, i.e., for any PPT algorithm the probability of outputting such (r, m_1, m_2) is negligible.*

FALCON uses an iterated version of SHAKE-256 to hash inputs (r, m) to degree-n polynomials c with coefficients from $\mathbb{Z}_q$. Since $q = 12289 \geq 2^{13}$ and $n = 512$ resp. $n = 1024$ the range of the hash function can

thus be assumed to be of size at least $q^n \geq 2^{6600}$, and SHAKE-256 should distribute well in this range. Hence, finding close-by hash values within the 2β-bound for the moderate values of β (below 9000) in FALCON should indeed be hard.

Proposition 5.3. *The signature scheme FALCON as described in Figure 8 provides* MBS *under the near-collision resistance assumption 5.2.*

Proof. Suppose an attacker against MBS is able to find distinct messages m_1 and m_2, a public key h and a signature (s_2, r) such that m_1 and m_2 are accepted under the given public key and signature. Let $c_1 \leftarrow \mathsf{H}(r, m_1)$ and $c_2 \leftarrow \mathsf{H}(r, m_2)$. Then using triangle inequality and monotony of the norm under appending a vector, we get $\|c_1 - c_2\| \leq \|c_1 - s_2 h\| + \|c_2 - s_2 h\| \leq \|(c_1 - s_2 h, s_2)\| + \|(c_2 - s_2 h, s_2)\| \leq 2\beta$. In other words, the adversary has found a near collision for H with small distance 2β. $\square$

To break S-DEO and NR we make an assumption about the distribution of the value s_2:

Assumption 5.4 ((Non-)Invertibility Assumption for s_2). *We assume that $s_2 \in \mathbb{Z}_q^n$ in the FALCON signature generation has a non-negligible probability of being invertible, as well as a non-negligible probability of being non-invertible.*

Invertibility is given iff all components of the NTT representation of s_2 are non-zero. If we assume that each component of $s_2 \in \mathbb{Z}_q^n$ is uniformly distributed then the probability of s_2 being invertible is $(\frac{q-1}{q})^n$ for dimension n. Recall that FALCON instantiates these values as $q = 12289$ and $n = 1024$ (or $n = 512$). This yields a probability of 92% (or 96%) for s_2 to be invertible. Correspondingly, we have a probability of 8% (or 4%) that s_2 is not invertible. Note that $(\frac{q-1}{q})^n \approx e^{-n/q}$ such that, asymptotically, if $q = \Theta(n)$ the probabilities for random s_2 being invertible and being non-invertible are roughly constant.

Proposition 5.5. *The signature scheme FALCON as described in Figure 8 does not provide* S-CEO *and therefore not* M-S-UEO. *Under the (non-)invertibility assumption for s_2 is does not provide* S-DEO *either.*

Proof. An attacker against S-CEO of FALCON is given a public key $\mathsf{pk} \leftarrow h$, queries the signature oracle on a message m, and gets a signature $\sigma \leftarrow (r, s)$ that verifies for m under this public key pk. We make a case distinction on whether $s_2 \leftarrow \mathsf{Decompress}(s)$ is invertible or not. Let us first assume that s_2 is invertible. Note that computing the inverse of s_2 can be done efficiently. The attacker sets $h' \leftarrow s_2^{-1} c$ and outputs $(\mathsf{pk}', m, \sigma)$ for $\mathsf{pk}' \leftarrow h'$. The signature σ verifies for m under pk' since $\sigma = (r, s)$ reconstructs the same c and $s_2 \leftarrow \mathsf{Decompress}(s)$ as in the original signature. In consequence, $s_1' \leftarrow c - s_2 h' = c - s_2 s_2^{-1} c = c - c = 0$ and therefore $\|(s_1', s_2)\|^2 \leq \|(s_1, s_2)\|^2 \leq \lfloor \beta^2 \rfloor$.

Let us now assume that s_2 is not invertible. Hence, there exists a non-zero $\alpha \in \mathbb{Z}_q[x]/(\phi)$ s.t. $s_2 \alpha = 0$. Computing α can be done efficiently in the FFT domain. The attacker sets $\mathsf{pk}' \leftarrow h + \alpha$ and outputs $(\mathsf{pk}', m, \sigma)$. The signature σ verifies for m under pk' since $\sigma = (r, s)$ reconstructs the same c and $s_2 \leftarrow \mathsf{Decompress}(s)$ as in the original signature. Therefore, $s_1' \leftarrow c - s_2 h' = c - s_2(h + \alpha) = c - s_2 h - s_2 \alpha = c - s_2 h$. Thus, $s_1' = s_1$ and the bound is satisfied trivially.

An attacker against S-DEO of FALCON can proceed in a similar fashion if s_2 is invertible, which it is with non-negligible probability according to our Assumption 5.4. The adversary in this case chooses a new message $m' \neq m$ and computes $c' \leftarrow \mathsf{H}(r, m')$. It sets $h' \leftarrow s_2^{-1} c'$, $\mathsf{pk}' \leftarrow h'$, and outputs $(\mathsf{pk}', m', \sigma)$. The signature σ verifies m' under pk' since $s_1' \leftarrow c' - s_2 h' = c' - s_2(s_2^{-1} c') = c' - c' = 0$ and therefore $\|(s_1', s_2)\|^2 \leq \|(s_1, s_2)\|^2 \leq \lfloor \beta^2 \rfloor$. $\square$

Proposition 5.6. *The signature scheme FALCON as described in Figure 8 does not provide* NR *under the (non-)invertibility assumption for s_2.*

Proof. An attacker against NR of FALCON is given a public key pk, a signature $\sigma \leftarrow (r, s)$ that verifies under this public key pk for a message m that is *unknown* to the attacker as well as circumstantial knowledge aux about the message. Not knowing the message prevents the adversary from mounting the same attack as in the S-CEO case when s_2 is invertible (because this requires knowledge of $c \leftarrow \mathsf{H}(r, m)$). We therefore use the attack case for s_2 not being invertible.

If $s_2 \leftarrow \mathsf{Decompress}(s)$ is not invertible, there exists a non-zero $\alpha \in \mathbb{Z}_q[x]/(\phi)$ s.t. $s_2\alpha = 0$ and the attacker can win by setting $\mathsf{pk}' \leftarrow h + \alpha$ and outputting (pk', σ). The signature σ verifies m under pk' since $s_1' \leftarrow c - s_2h' = c - s_2(h + \alpha) = c - s_2h - s_2\alpha = c - s_2h$. Thus, $s_1' = s_1$ and the bound is satisfied trivially. According to Assumption 5.4 the probability of s_2 not being invertible is non-negligible, such that the attacker succeeds with non-negligible probability as well. $\qquad\square$

We note FALCON uses SHAKE-256 as the underlying hash function H. Hence, if one would apply our general transformation with this hash function H, collision resistance and non-malleablity would conceivably hold, and the resulting scheme would obtain all security properties.

5.3 Rainbow

The signature scheme Rainbow [DCK$^+$20] is based on multivariate cryptography. In particular, its security is based on the multivariate quadratic problem. Rainbow employs a one-way function $\mathcal{P} : \mathbb{F}^n \to \mathbb{F}^k$ which is a multivariate quadratic polynomial map in $n = k + v_1$ variables where the coefficients are taken from the field $\mathbb{F}$. The trapdoor is the knowledge of the composite functions of $\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T}$ where $\mathcal{S}$ and $\mathcal{T}$ are invertible affine maps and the central map $\mathcal{F}$ is quadratic consisting of k multivariate polynomials. The first v variables are called *vinegar* variables, while the remaining k variables are called *oil* variables. The central map $\mathcal{F}$ has no quadratic terms that contain two oil variables. The maps $\mathcal{S}$ and $\mathcal{T}$ are chosen to be linear, while $\mathcal{F}$ is homogeneous of degree 2, and, hence, so is $\mathcal{P}$. We give an algorithmic description of Rainbow in Figure 9.

The key generation algorithm generates the coefficients of the three maps $\mathcal{S}$, $\mathcal{F}$, and $\mathcal{T}$ pseudorandomly with the help of a short seed s_{priv}. The coefficients of the polynomials of these maps form the signing key whereas the composition $\mathcal{P}$ yields the public key. Intuitively, a Rainbow signature is the preimage of a randomized hash of the message m under $\mathcal{P}$. That is, the signer computes $\mathbf{h} \leftarrow \mathsf{H}(\mathsf{H}(m), r)$ for a random r and then solves for $\mathbf{z}$ in $\mathcal{P}(\mathbf{z}) = \mathbf{h}$ with the help of the decomposition of $\mathcal{P}$. For this the signer first solves $\mathbf{x} = \mathcal{S}^{-1}(\mathbf{h})$ and then computes $\mathbf{y} = \mathcal{F}^{-1}(\mathbf{x})$ by fixing the vinegar variables $\mathbf{v}$ in $\mathbf{y}$ to randomly chosen values. This reduces the equation to a linear system, which can be solved with Gaussian elimination. Finally, derive $\mathbf{z} = \mathcal{T}^{-1}(\mathbf{y})$ to build the signature $(\mathbf{z}, r)$. The signing step may require to try multiple $\mathbf{v}$ and r to be able to find a solution. The verification algorithm recomputes the hash $\mathbf{h} \leftarrow \mathsf{H}(\mathsf{H}(m), r)$, and accepts if this digest is equal to $\mathcal{P}$ evaluated at $\mathbf{z}$.

Note that Rainbow proposes two additional variants which differ from standard Rainbow in the way keys are computed and stored. Instead of computing the public key from the secret key, major parts of the public key will be fixed and then the central map $\mathcal{F}$ is computed. In more detail, the CZ (circumzenithal) variant generates a portion of $\mathcal{P}$ and the matrices $\mathcal{S}$ and $\mathcal{T}$ from small seeds s_{pub} and s_{priv}, respectively, using an AES-based PRNG. From this, the central map $\mathcal{F}$ and $\mathcal{P}_2$ (the remaining parts of $\mathcal{P}$) can be computed obtaining a key pair. Note that this variant does not store the whole map $\mathcal{P}$. Instead, it only stores s_{pub} and $\mathcal{P}_2$ to reduce the public key size, and just reconstructs $\mathcal{P}$ when needed. This comes at the expense of significantly increased verification time. The compressed variant is even more compact than the CZ variant: It stores only the two seeds in the secret key and computes all matrices when they are needed. This increases both the signing and verification time.

In the following, we start showing that Rainbow achieves message-bound signatures followed by showing that it does *not* provide non re-signability, conservative exclusive ownership, and destructive exclusive

KGen(1^λ)	Sig(sk, m)	Vf(pk, m, σ)
11: $s_{priv} \leftarrow\!\!\$\ \{0,1\}^{256}$	21: **repeat**	41: $(\mathbf{z}, r) \leftarrow \sigma$
12: $(\mathcal{S}, \mathcal{T}, \mathcal{F}) \leftarrow \mathsf{PRNG}(s_{priv})$	22: $\quad \mathbf{v} \leftarrow\!\!\$\ \mathbb{F}^v$	42: $\mathbf{h} \leftarrow \mathsf{H}(\mathsf{H}(m), r)$
13: $\mathcal{P} \leftarrow \mathcal{S} \circ \mathcal{F} \circ \mathcal{T}$	23: **until** $\mathcal{F}$, with $\mathbf{v}$ set,	43: **return** $[\mathcal{P}(\mathbf{z}) = \mathbf{h}]$
14: $\mathsf{sk} \leftarrow (\mathcal{S}, \mathcal{F}, \mathcal{T})$	$\quad\quad$ is invertible	
15: $\mathsf{pk} \leftarrow \mathcal{P}$	24: **repeat**	
16: **return** (sk, pk)	25: $\quad r \leftarrow\!\!\$\ \{0,1\}^{128}$	
	26: $\quad \mathbf{h} \leftarrow \mathsf{H}(\mathsf{H}(m), r)$	
	27: $\quad \mathbf{x} \leftarrow \mathcal{S}^{-1}(\mathbf{h})$	
	28: $\quad \mathbf{o} \leftarrow \mathsf{solve}(\mathcal{F}(\mathbf{v}\|\mathbf{o}) = \mathbf{x})$	
	29: **until** $\mathbf{o}$ is a valid solution	
	30: $\mathbf{y} \leftarrow (\mathbf{v}\|\mathbf{o})$	
	31: $\mathbf{z} \leftarrow \mathcal{T}^{-1}(\mathbf{y})$	
	32: $\sigma \leftarrow (\mathbf{z}, r)$	
	33: **return** σ	

Figure 9: Algorithmic description of Rainbow.

ownership.

Proposition 5.7. *The signature scheme Rainbow as described in Figure 9 (and its two variants) achieve* MBS *if the hash function* H *is collision resistant.*

Proof. A successful attacker against MBS of Rainbow yields a public key pk, a signature $\sigma \leftarrow (\mathbf{z}, r)$, and two messages m_1, m_2, where σ verifies for both m_1 and m_2 under pk where $m_1 \neq m_2$.

In the verification algorithm the message m_1 is hashed to $\mathbf{h_1} \leftarrow \mathsf{H}(\mathsf{H}(m_1), r)$ and m_2 to $\mathbf{h_2} \leftarrow \mathsf{H}(\mathsf{H}(m_2), r)$. If $\mathbf{h_1} = \mathbf{h_2}$ the attacker breaks collision resistance of H. If $\mathbf{h_1} \neq \mathbf{h_2}$ and both messages verify it must hold that $\mathbf{h_1} = \mathcal{P}(\mathbf{z}) = \mathbf{h_2}$ while $\mathbf{h_1} \neq \mathbf{h_2}$. $\qquad\square$

Proposition 5.8. *The signature scheme Rainbow as described in Figure 9 (and its two variants) do not provide* NR.

Proof. An attacker against NR of any variant of Rainbow is given a public key pk, a signature $\sigma \leftarrow (\mathbf{z}, r)$ that verifies under this public key pk for a message m that is *unknown* to the attacker, as well as circumstantial knowledge aux about the message. Note that the attacker can reconstruct $\mathcal{P}$ from pk for all variants of Rainbow. Since the signature σ verifies for the message m, it must hold that $\mathcal{P}(\mathbf{z}) = \mathbf{h'} = \mathsf{H}(\mathsf{H}(m), r)$. Thus, the attacker can learn the hash value $\mathbf{h'}$ by simply computing $\mathbf{h'} \leftarrow \mathcal{P}(\mathbf{z})$. Equipped with this, the attacker generates its own key pair (sk′, pk′) and then executes the signing algorithm (cf. Figure 9) with its own secret key sk′ and three minor changes: First, instead of sampling a random salt the attacker reuses r from the signature it initially received. Second, instead of computing the hash value $\mathbf{h}$ as described in the scheme the attacker uses the hash value $\mathbf{h'}$ it computed before. Third, in case the Gaussian elimination does not yield a valid $\mathbf{o}$, the attacker restarts with sampling new vinegar variables. For the remaining part of the algorithm it simply proceeds as specified and finally receives a valid signature that correctly verifies under its chosen public key for the message m even without knowing the message. The attacker outputs pk′ and the output of the modified sign algorithm. $\qquad\square$

Proposition 5.9. *The signature scheme Rainbow as described in Figure 9 does not provide* S-CEO *and therefore not* M-S-UEO *and, assuming collision resistance of the hash function, neither* DEO.

Proof. An attacker against S-CEO of Rainbow is given a public key $\mathsf{pk} \leftarrow \mathcal{P}$, queries the signature oracle on a message m, and gets a signature $\sigma \leftarrow (\mathbf{z}, r)$ that verifies for m under this public key pk. Let $\mathbf{h} \leftarrow \mathsf{H}(\mathsf{H}(m), r)$. If $\mathbf{z}$ is zero, then so is $\mathbf{h}$ as it satisfies $\mathbf{h} = \mathcal{P}(\mathbf{z})$ for the homogeneous polynomial $\mathcal{P}$. In this case, the attacker can pick $\mathcal{P}'$ to be an arbitrary homogeneous polynomial of degree 2. In the case $\mathbf{z} = (z_1, \ldots, z_n)$ is non-zero with $z_\lambda \neq 0$, the attacker picks a homogeneous polynomial map $\mathcal{P}^*$ of degree 2 with $\mathcal{P}^*(\mathbf{z}) = \mathbf{h}$ as follows: For each $j \in [1, k]$ set $p_j(\mathbf{x}) = (h_j z_\lambda^{-2}) x_\lambda^2$ such that each p_j is homogeneous of degree 2. For any j it holds that $p_j(\mathbf{z}) = (h_j z_\lambda^{-2}) z_\lambda^2 = h_j$. Hence, setting $\mathcal{P}^* = (p_1, \ldots, p_k)$ we find that $\mathcal{P}^*(\mathbf{z}) = \mathbf{h}$. If $\mathcal{P}^* \neq \mathcal{P}$, the attacker returns $(\mathcal{P}^*, m, \sigma)$.

If $\mathcal{P}^*$ coincides with $\mathcal{P}$, we can compute a distinct mapping $\mathcal{P}' \neq \mathcal{P}$ as follws. Consider the set $\mathbf{S} \leftarrow \{q_{k\ell}(\mathbf{x}) = x_k x_\ell - \frac{z_k}{z_\lambda} x_\lambda x_\ell : k, \ell \in [1, n]\}$. Note that $\mathbf{S}$ consists of homogeneous polynomials of degree 2. By construction $q_{k\ell}(\mathbf{z}) = z_k z_\ell - \frac{z_k}{z_\lambda} z_\lambda z_\ell = 0$ and $p_j'(\mathbf{x}) = p_j(\mathbf{x}) + q_{k\ell}(\mathbf{x})$ is thus another polynomial with $p_j'(\mathbf{z}) = h_j$. We can therefore efficiently compute another polynomial map $\mathcal{P}' = (p_1', \ldots, p_k')$ of the required form. The attacker returns $(\mathcal{P}', m, \sigma)$.

Similarly, an attacker against S-DEO of Rainbow receives a signature $\sigma \leftarrow (\mathbf{z}, r)$ for a message m for which $\mathbf{h} \leftarrow \mathsf{H}(\mathsf{H}(m), r)$. We assume $\mathbf{z} \neq 0$, else it asks for another signature $\tilde{\sigma} \leftarrow (\tilde{\mathbf{z}}, \tilde{r})$ for another message $\tilde{m}$. If again $\tilde{\mathbf{z}} = 0$ then both hash values $\mathbf{h}, \tilde{\mathbf{h}} \leftarrow \mathsf{H}(\mathsf{H}(\tilde{m}), \tilde{r})$ of the requested signatures would collide in 0, since $\mathbf{h} = \mathcal{P}(\mathbf{z}) = \mathcal{P}(0) = 0 = \mathcal{P}(\tilde{\mathbf{z}}) = \tilde{\mathbf{h}}$, contradicting the collision resistance of H. Hence we can assume that the adversary eventually holds a signature $\sigma \leftarrow (\mathbf{z}, r)$ for m with $\mathbf{z} \neq 0$. The adversary now picks a message $m' \neq m$ and computes $\mathbf{h}' \leftarrow \mathsf{H}(\mathsf{H}(m'), r)$ for the given value r in the signature. Then it proceeds as above to obtain $\mathcal{P}' \neq \mathcal{P}$ with $\mathcal{P}'(\mathbf{z}) = h'$ and returns $(\mathcal{P}', m', \sigma)$. □

The attack against S-CEO and S-DEO does not immediately carry over to the CZ and compressed variants. The reason is that the variants use seeds to generate public keys such that we cannot pick suitable mauled keys easily. Unfortunately, we cannot provide formal proofs showing that the variants of Rainbow achieve S-CEO or S-DEO. The attack presented for standard Rainbow in the proof for Proposition 5.9 cannot be applied to the variants since the coefficients of $\mathcal{P}$ are not explicitly stored in the public key. Instead, the coefficients are partially generated with a PRNG from the seed s_{pub}. Intuitively, it is hard to find a seed s_{pub} that results in the particular coefficients you have fixed before. Another possible attack vector is to randomly sample s_{pub} and trying to set the remainder $\mathcal{P}_2$ of $\mathcal{P}$ such that $\mathcal{P}(\mathbf{z}) = \mathbf{h}$ for a preimage $\mathbf{z}$ and a hash digest $\mathbf{h}$ for a given message m and signature $\sigma \leftarrow (\mathbf{z}, r)$ obtained from the signing oracle with $\mathbf{h} \leftarrow \mathsf{H}(\mathsf{H}(m), r)$. We assume that it is hard to find such a $\mathcal{P}_2$.

Rainbow recommends SHA256 as the underlying hash function H. Hence, whereas the scheme currently does not satisfy all security properties, using our general transformation with the implemented hash function would be considered to achieve the stronger guarantees.

5.4 GeMSS

The signature scheme GeMSS [CFMR+20] is built from multivariate cryptography. GeMSS (short for <u>Ge</u>at <u>M</u>ultivariate <u>S</u>hort <u>S</u>ignature) evolved from QUARTZ [PCG01] and relies on hidden field equations (HFE) with vinegar specialization. GeMSS employs a set of k quadratic square-free non-linear polynomials in $n + v$ variables over $\mathbb{F}_2$. In particular, $\mathbf{p} = (p_1, \ldots, p_k) \in \mathbb{F}_2[z_1, \ldots, z_{n+v}]^k$. The trapdoor is the knowledge of the three components $\mathcal{S}, F$, and $\mathcal{T}$ from which the public key can be generated. The invertible matrices $\mathcal{S}$ and $\mathcal{T}$ are of degree $n + v$ and n, respectively. The polynomial $F \in \mathbb{F}_{2^n}[X, v_1, ..., v_n]$ becomes an HFE polynomial for any specialization of the vinegar variables, i.e. F is of HFEv-shape. In Figure 10, we provide an algorithmic description of GeMSS.

```
KGen(1^λ)                                                  Vf(pk, m, σ)
─────────────────────────────────────                     ─────────────────────────────────────
11 :   (S, T) ←$ (GL_{n+v}(𝔽_2) × GL_n(𝔽_2))              31 :   (s_{nb_ite}, x_{nb_ite}, ..., x_1) ← σ
12 :   F ←$ 𝔽_{2^n}[X, v_1, ..., v_v]   // with HFEv-shape  32 :   h ← H(m)
13 :   sk ← (F, S, T)                                      33 :   for i from 1 to nb_ite do
14 :   (p_1, ..., p_n) ← Eval(F, S, T)                     34 :       d_i ← first k bits of h
15 :   p ← first k polynomials of (p_1, ..., p_n)          35 :       h ← H(h)
16 :   pk ← p                                              36 :   endfor
17 :   return (sk, pk)                                     37 :   for i from nb_ite − 1 to 0 do
                                                           38 :       s_i ← p(s_{i+1}, x_{i+1}) ⊕ d_{i+1}
                                                           39 :   endfor
Sig(sk, m)                                                 40 :   return [s_0 = 0]
─────────────────────────────────────
21 :   h ← H(m)
22 :   s_0 ← 0 ∈ 𝔽_2^k
23 :   for i from 1 to nb_ite do
24 :       d_i ← first k bits of h
25 :       (s_i, x_i) ← Inv_p(d_i ⊕ s_{i−1}, sk)
26 :       h ← H(h)
27 :   endfor
28 :   σ ← (s_{nb_ite}, x_{nb_ite}, ..., x_1)
29 :   return σ
```

Figure 10: Algorithmic description of GeMSS.

In more detail, the key generation algorithm first randomly samples two invertible matrices $\mathcal{S}$ and $\mathcal{T}$. Next, it samples a polynomial F with HFEv-shape. The public key $\mathbf{p}$ is set to the first $k = n - \Delta$ polynomials that are generated through evaluation-interpolation from $F, \mathcal{S}, \mathcal{T}$, while the secret key is the knowledge of $F, \mathcal{S}$, and $\mathcal{T}$. The secret key allows to compute the inverse $\mathsf{Inv}_\mathbf{p}(\cdot, \mathsf{sk})$ to $\mathbf{p}$. The signing algorithm starts with hashing the message to the digest h and sets $\mathbf{s}_0$ to the element $\mathbf{0}$ of the vector space. The next steps are executed nb_ite times, where i is the number of the iteration: $\mathbf{d}_i$ is set to the first k bits of h; $(\mathbf{s}_i, \mathbf{x}_i)$ is computed through the inversion function $\mathsf{Inv}_\mathbf{p}$ taking as input $\mathbf{d}_i \oplus \mathbf{s}_{i-1}$ and sk. Next, h is given as an input to H outputting a new digest which is used in the next iteration for deriving $\mathbf{d_i}$. The signature consists of the final value $\mathbf{s}_{nb\_ite}$ and all $\mathbf{x}_i$ values. The verification algorithm starts with hashing the message and then sets $\mathbf{d}_i$ to the first k bits of the ith hash digest with $i \in \{1, \ldots, nb\_ite\}$. Next, it computes $\mathbf{s}_i$ for i descending from $nb\_ite - 1$ to 0 as $\mathbf{p}(\mathbf{s}_{i+1}, \mathbf{x}_{i+1}) \oplus \mathbf{d}_{i+1}$. The algorithm accepts the signature if $\mathbf{s}_0$ is equal to $\mathbf{0}$.

Note that the number of iterations nb_ite is chosen such that $2^{k\frac{nb\_ite}{nb\_ite+1}} \geq 2^\lambda$ and usually corresponds to either 3 or 4 depending on the chosen parameters.

Proposition 5.10. *The signature scheme GeMSS as described in Figure 10 provides neither* S-CEO *nor* S-DEO*, and therefore also not* M-S-UEO*.*

Proof. An attacker against S-CEO of GeMSS is given a public key pk, queries the signature oracle on a message m, and gets a signature $\sigma \leftarrow (\mathbf{s}_{nb\_ite}, \mathbf{x}_{nb\_ite}, ..., \mathbf{x}_1)$ that verifies for m under this public key pk. The attacker can now compute $\mathbf{d}_1$ as the first k bits of $\mathsf{H}(m)$ and build a new public key $\mathsf{pk}' \leftarrow \mathbf{p}'$ that has the constant components set equal to $\mathbf{d}_1$ while all other coefficients are set to 0. Since $\mathbf{p}'$ of pk' always

evaluates to $\mathbf{d}_1$, verification in the last step computes $\mathbf{s_0}$ as $\mathbf{0}$ for this message m. Hence, σ verifies for m under pk', too.

An attacker against S-DEO can proceed in a similar fashion. The difference is that this attacker computes $\mathbf{d}_1$ as the first k bits of $\mathsf{H}(m')$ with respect to a message $m' \neq m$ and sets the constant part of pk' accordingly. By construction, σ verifies for m' under pk'. $\qquad\square$

We rely on a mild assumption to attack MBS of GeMSS:

Assumption 5.11 (Distribution of $\mathbf{s}_1^{(2)}$). *Let m_1, m_2 be two distinct messages. Let $\mathbf{s}_i$ denote intermediate values for verifying m_1 as described in Figure 10 and $\mathbf{s}_i^{(2)}$ for verifying m_2 with the same signature and public key where $\mathbf{s}_{nb\_ite}^{(2)} = \mathbf{s}_{nb\_ite}$. We assume that $\mathbf{s}_1^{(2)}$ is not contained in the linear span $\langle \mathbf{s}_i, i \in [1, nb\_ite]; \mathbf{s}_j^{(2)}, j \in [2, nb\_ite] \rangle$.*

This assumption appears to be plausible, if one considers that the digests are randomly distributed and xoring values of a polynomial evaluation changes the distribution at most slightly, ensuring that the values are still distributed well. Moreover, the dimension of the space $\langle \mathbf{s}_i, i \in [1, nb\_ite]; \mathbf{s}_j^{(2)}, j \in [2, nb\_ite] \rangle$ is at most $2 nb\_ite - 2 < 10$ while $\mathbf{s}_1^{(2)}$ is an element of a k-dimensional space with $k > 150$ for any parameter set and any security level.

Proposition 5.12. *The signature scheme GeMSS as described in Figure 10 does not provide MBS under the Distribution Assumption 5.11 for $\mathbf{s}_1^{(2)}$.*

Proof. An attacker against MBS of GeMSS has to output a public key $\mathbf{p}$, a signature σ and two distinct messages m_1, m_2 that verify under the public key with the same signature. The attacker honestly generates a key pair $(\mathsf{sk}, \mathbf{p})$, chooses messages m_1, m_2 and obtains the signature σ from honestly signing m_1. Let $\mathbf{d}_i, \mathbf{s}_i$ denote intermediate values for verifying m_1 as described in Figure 10 and $\mathbf{d}_i^{(2)}, \mathbf{s}_i^{(2)}$ for verifying m_2 where $\mathbf{s}_{nb\_ite}^{(2)} = \mathbf{s}_{nb\_ite}$. Note that, a priori, it does not necessarily hold that $\mathbf{p}(\mathbf{s}_1^{(2)}, \mathbf{x}_1) \oplus \mathbf{d}_1^{(2)} = \mathbf{0}$.

For each $l \in [1, k]$, there exists a polynomial q_l that satisfies $q_l(\mathbf{s}_i, \mathbf{x}_i) = 0$ for $i \in [1, nb\_ite]$ and $q_l(\mathbf{s}_j^{(2)}, \mathbf{x}_j) = 0$ for $j \in [2, nb\_ite]$ and $q_l(\mathbf{s}_1^{(2)}, \mathbf{x}_1) = p_l(\mathbf{s}_1^{(2)}, \mathbf{x}_1) \oplus d_{1,l}^{(2)}$. In case $p_l(\mathbf{s}_1^{(2)}, \mathbf{x}_1) \oplus d_{1,l}^{(2)} = 0$ we set $q_l = 0$. Otherwise, we use Assumption 5.11 that $\beta \leftarrow \mathbf{s}_1^{(2)}$ is not contained in $\langle \mathbf{s}_i, i \in [1, nb\_ite]; \mathbf{s}_j^{(2)}, j \in [2, nb\_ite] \rangle$. Then we extend β to a basis $(\beta, \alpha_1, \ldots, \alpha_{k-1})$ of $\mathbb{F}_2^k$ such that $\mathbf{s}_i, \mathbf{s}_j^{(2)} \in U \leftarrow \langle \alpha_1, \ldots, \alpha_{k-1} \rangle$. Note that any element $x \in \mathbb{F}_2^k$ can be written uniquely as $x = \mu_\beta \beta + u$ with $u \in U$ and $\mu_\beta \in \mathbb{F}_2$. The projection $\pi \colon \mathbb{F}_2^k \to \mathbb{F}_2$ given by $\mu_\beta \beta + u \mapsto \mu_\beta$ satisfies that $\pi(\beta) = 1$ and $\pi(\mathbf{s}_i) = 0$ for all $i \in [1, nb\_ite]$ and $\pi(\mathbf{s}_j^{(2)}) = 0$ for all $j \in [2, nb\_ite]$. As the projection is linear, it is a polynomial map of degree 1. We set q_l to be the polynomial map corresponding to π viewed as element of $\mathbb{F}_2[z_1, \ldots, z_{n+v}]$. Then replacing $\mathbf{p}$ with $\mathbf{p}' \leftarrow \mathbf{p} + (q_1, \ldots, q_k)$ yields a polynomial map under which σ verifies both m_1 and m_2: When verifying the message m_1 the additional q_l terms vanish since $q_l(\mathbf{s}_i, \mathbf{x}_i) = 0$ for all $l \in [k]$. When verifying the message m_2 the additional q_l terms vanish as well except for the last iteration. There we have $p_l'(\mathbf{s}_1^{(2)}, \mathbf{x}_1) = p_l(\mathbf{s}_1^{(2)}, \mathbf{x}_1) + q_l(\mathbf{s}_1^{(2)}, \mathbf{x}_1) = p_l(\mathbf{s}_1^{(2)}, \mathbf{x}_1) \oplus p_l(\mathbf{s}_1^{(2)}, \mathbf{x}_1) \oplus d_{1,l}^{(2)} = d_{1,l}^{(2)}$. Hence, the attacker succeeds by returning $(\mathbf{p}', m_1, m_2, \sigma)$. $\qquad\square$

Proposition 5.13. *The signature scheme GeMSS as described in Figure 10 does not provide NR.*

Proof. An attacker against NR of GeMSS is given a public key $\mathbf{p}$, a signature $\sigma \leftarrow (\mathbf{s}_{nb\_ite}, \mathbf{x}_{nb\_ite}, \ldots, \mathbf{x}_1)$ that verifies under this public key $\mathbf{p}$ for a message m that is *unknown* to the attacker as well as circumstantial knowledge aux about the message. Similarly to the attack against MBS of GeMSS, the attacker constructs a polynomial q that satisfies $q(\mathbf{s}_j, \mathbf{x}_j) = 0$ for $j \in [2, nb\_ite]$ and $q(\mathbf{s}_1, \mathbf{x}_1) = p_l(\mathbf{s}_1, \mathbf{x}_1) \oplus d_{1,l}$ for an arbitrary $l \in [1, k]$. The attacker picks an element $\beta \in \mathbb{F}_2^{n+b}$ that is not contained in the linear span of $(\mathbf{s}_i, \mathbf{x}_i)$, for

$i \in [1, nb\_ite]$ and defines q as the projection onto the line generated by β such that $(\mathbf{s}_i, \mathbf{x}_i)$ are mapped to 0. The attacker outputs $\mathbf{p} + q\mathbf{e}_l$ for some l as new public key and the same signature. Here, $\mathbf{e}_l$ denotes the vector in $\mathbb{F}_2[z_1, \ldots, z_{n+v}]^n$ which has a 1 in the l-th component and 0 everywhere else, so that $\mathbf{p}$ is changed only in one component by adding q. □

GeMSS uses SHA-3 for the underlying hash operations. This hash function is believed to have the required properties to securely apply our BUFF transformation. Thus, with this hash function it is reasonable to assume that the modified scheme achieves the stronger security guarantees.

5.5 Picnic

The signature scheme Picnic [CDG+20, Zav20] is a family of digital signature algorithms using as its main building blocks a zero-knowledge proof, as well as symmetric key primitives such as a hash function and a block cipher. On a high-level, Picnic is obtained by transforming an interactive zero-knowledge proof of knowledge protocol into a non-interactive signature scheme using Fiat-Shamir transform [FS87] or Unruh transform [Unr15]. Currently, all variants of Picnic basically follow the same design principle, however relying on different variants of the proof of knowledge protocol ZKBoo [GMO16], a different implementation of the block cipher, or other parameter sets. The first variant, Picnic, can be instantiated using a variant of ZKBoo called ZKB++ [CDG+17] as proof of knowledge protocol in combination with the Fiat-Shamir transform or the Unruh transform. The other variants, Picnic2 and Picnic3, use instead a different variant of ZKBoo called KKW [KKW18, KZ20] as proof of knowledge protocol and the Fiat-Shamir transform. Since Picnic3 outperforms Picnic2 due to new parameter sets and optimizations of the block cipher, the latter was deprecated in favor of the former.

All of the proof of knowledge protocols use the so-called multi party computation (MPC)-in-the-head paradigm [IKOS09]. The general idea of the paradigm is that the prover simulates an execution of a MPC protocol, commits to the view of each party, and opens a part of the commitments according to the challenge issued by the verifier. The proof of knowledge protocol is used to prove the knowledge of a key for a block cipher that is always instantiated as LowMC [ARS+15], i.e., the secret input to a boolean circuit evaluating LowMC. Hence, in the key generation algorithm, the secret key sk is a randomly sampled key for the LowMC cipher and the public key is a randomly sampled plaintext p and its LowMC encryption under sk.

To obtain a signature from the proof of knowledge protocol, the challenge is computed deterministically as a hash function evaluation of the public key and the message among other values, as described by the Fiat-Shamir transform or Unruh transform, respectively. Further values included in the challenge relate to the execution of the MPC-in-the-head protocol, i.e., output shares and commitments to the view of each party. Finally, the signature consists of the challenge, the zero-knowledge proof and a salt. The verification step recomputes part of the proof of knowledge protocol and the challenge. If the recomputed challenge corresponds to the one provided in the signature then verification is successful, otherwise not. An algorithmic description of Picnic is provided in Figure 11.

The authors of Picnic have considered the notion of CEO for Picnic, referring to it in the terminology of [MS04]. They argue that all Picnic variants provide CEO in [CDG+20, Section 7.3].

In the following, we show that the signature scheme Picnic achieves all security notions from Sections 3 and 4.

Proposition 5.14. *The signature scheme Picnic and its variants as described in Figure 11 achieve* M-S-UEO, MBS, *and* NR, *if the hash function* H *is collision resistant and* Φ-*non-malleable for* $\Phi = \{\phi_{\mathsf{pk}',\psi}\}$ *and* $\phi_{\mathsf{pk}',\psi}(a, \mathsf{pk}, m) = (\psi(a), \mathsf{pk}', m)$ *for any function* ψ.

Proof. Inspecting the signature scheme as summarized in Figure 11 shows that the generated signature contains the hash digest that was generated from inputting the public key and the message (among other

27

KGen(1^λ)	Sig(sk, m)
11 : $p \leftarrow\!\!\$\, \{0,1\}^\lambda$	21 : $(a, salt) \leftarrow\!\!\$\, \mathsf{simulate\_circuit}(\mathsf{sk}, \mathsf{pk}, m)$
12 : $\mathsf{sk} \leftarrow\!\!\$\, \{0,1\}^\lambda$	22 : $c \leftarrow \mathsf{H}(a, \mathsf{pk}, m)$
13 : $C \leftarrow \mathsf{Enc}_{LowMC}(\mathsf{sk}, p)$	23 : $z \leftarrow \mathsf{Prove}(c, a, salt)$
14 : $\mathsf{pk} \leftarrow (C, p)$	24 : $\sigma \leftarrow (c, z, salt)$
15 : **return** $(\mathsf{sk}, \mathsf{pk})$	25 : **return** σ

Vf(pk, m, σ)

31 : parse σ as $(c, z, salt)$
32 : $a' \leftarrow \mathsf{recompute\_circuit}(\mathsf{pk}, m, c, z, salt)$
33 : $c' \leftarrow \mathsf{H}(a', \mathsf{pk}, m)$
34 : **return** $[c' = c]$

Figure 11: Algorithmic description of Picnic based on Scheme 5 in [CDG$^+$20].

values) into the hash function. Observe that the verification algorithm explicitly checks the hash value. Hence, Picnic implements our BUFF transformation as specified in Figure 5 and therefore Theorem 4.5 applies to Picnic. Thus, it follows directly that Picnic achieves M-S-UEO and message-bound signatures if the hash function is collision resistant. Non re-signability follows if the hash function is collision resistant and Φ-non-malleable for $\Phi = \{\phi_{\mathsf{pk}',\psi}\}$ where $\phi_{\mathsf{pk}',\psi}(a, \mathsf{pk}, m) = (\psi(a), \mathsf{pk}', m)$. This holds as in the proof of Theorem 4.5 for our BUFF transformation. $\qquad\square$

Picnic requires to use SHAKE-256 as the hash function (prepended with a byte to derive quasi independent hash functions). It is thus reasonable to assume that the hash function is collision resistant and non-malleable, meaning that the signature schemes already provides the other security properties, as it follows our BUFF transformation.

5.6 SPHINCS$^+$

The signature scheme SPHINCS$^+$ [ABB$^+$20] is a hash-based signature scheme based on Merkle trees, in particular on XMSS [BDH11]. SPHINCS$^+$ makes use of a one-time signature scheme (OTS) and a few-time signature scheme (FTS). Both of these schemes allow computing the public key from a signature. In the following we describe SPHINCS$^+$ and its required components on a high-level. For the full details, we refer to [ABB$^+$20] and [BHK$^+$19].

Let us start with describing the respective details about the FTS scheme which is called FORS (Forest of Random Subsets). Such a forest consists of k trees with $t = 2^a$ leaves each. The secret key consists of the random values in the leaves of all k trees, while the public key is a hash of the root of all k trees. The length of a message is exactly ka bits. To sign a message the message is split into k blocks of equal length. Each block indicates one leaf in one of the k trees. The signature then consists of the k leaves and their authentication path to the root of the corresponding tree. An authentication path in a tree consists of the sibling nodes on the way from the node to the root. FORS does not provide a verification algorithm. Instead, it provides an algorithm called pkFromSig that allows to compute the public key from the signature. The public key is then implicitly checked by the next step.

On a high level, SPHINCS$^+$ uses a hypertree to authenticate FORS public keys that are used to sign messages. The hypertree is composed of several layers of trees. The leaves of the trees on the bottom layer

$$
\begin{array}{ll}
\underline{\mathsf{KGen}(1^\lambda)} & \underline{\mathsf{Sig}(\mathsf{sk}, m)} \\[4pt]
11:\quad \mathsf{sk}.seed \leftarrow\!\$\,\{0,1\}^{8\lambda}, \mathsf{sk}.prf \leftarrow\!\$\,\{0,1\}^{8\lambda} & 21:\quad r \leftarrow \mathrm{PRF}_{msg}(\mathsf{sk}.prf, OptRand, m) \\[2pt]
12:\quad \mathsf{pk}.seed \leftarrow\!\$\,\{0,1\}^{8\lambda}, \mathsf{pk}.root \leftarrow \text{hypertree root} & 22:\quad (md\|idx) \leftarrow \mathsf{H}_{msg}(r, \mathsf{pk}, m) \\[2pt]
13:\quad \mathsf{sk} \leftarrow (\mathsf{sk}.seed, \mathsf{sk}.prf) & 23:\quad \sigma_{FORS} \leftarrow \mathsf{Sig}_{FORS}(md, \mathsf{sk}.seed, \mathsf{pk}.seed, idx) \\[2pt]
14:\quad \mathsf{pk} \leftarrow (\mathsf{pk}.root, \mathsf{pk}.seed) & 24:\quad \mathsf{pk}_{FORS} \leftarrow \mathsf{pkFromSig}_{FORS}(\sigma_{FORS}, md, \mathsf{pk}.seed, idx) \\[2pt]
15:\quad \textbf{return } (\mathsf{sk}, \mathsf{pk}) & 25:\quad \sigma_{HT} \leftarrow \mathsf{Sig}_{HT}(\mathsf{pk}_{FORS}, \mathsf{sk}.seed, \mathsf{pk}.seed, idx) \\[2pt]
 & 26:\quad \sigma \leftarrow (r, \sigma_{FORS}, \sigma_{HT}) \\[2pt]
 & 27:\quad \textbf{return } \sigma \\[10pt]
 & \underline{\mathsf{Vf}(\mathsf{pk}, m, \sigma)} \\[4pt]
 & 31:\quad (r, \sigma_{FORS}, \sigma_{HT}) \leftarrow \sigma \\[2pt]
 & 32:\quad (md\|idx) \leftarrow \mathsf{H}_{msg}(r, \mathsf{pk}, m) \\[2pt]
 & 33:\quad \mathsf{pk}_{FORS} \leftarrow \mathsf{pkFromSig}_{FORS}(\sigma_{FORS}, md, \mathsf{pk}.seed, idx) \\[2pt]
 & 34:\quad \textbf{return } \mathsf{Vf}_{HT}(\mathsf{pk}_{FORS}, \sigma_{HT}, idx, \mathsf{pk}.root)
\end{array}
$$

Figure 12: Algorithmic description of SPHINCS$^+$.

are FORS public keys. The leaves of all other trees are public keys for an OTS scheme that is used to sign the root of the tree one layer below. Each inner node of each tree (including the trees in the FORS key pairs) is a hash value of the public seed, the address of the node in the hypertree, and its two children using the hash function H, i.e. $node \leftarrow \mathsf{H}(\mathsf{pk}.seed, \mathrm{ADRS}, \mathrm{leftChild}, \mathrm{rightChild})$. In contrast, the hash function H_{msg} is used only once per execution of the signing or verification algorithm to obtain the message digest and index to be used. In Figure 12, we provide an algorithmic description of SPHINCS$^+$.

Remark. We follow the notation in [ABB$^+$20] regarding the arguments to the signing and verification algorithms of the subschemes, respectively the $\mathsf{pkFromSig}$ algorithm. As a result of adapting the notation, the message is the first argument to each algorithm. The algorithms also receive more arguments than we have defined in Definition 2.1 to include the address of the message, i.e., the index of the node in the hypertree.

The key generation algorithm sets the public key to the root of the hypertree $\mathsf{pk}.root$ and a seed $\mathsf{pk}.seed$ to tie executions of a hash function to these public values. The secret key consists of a seed $\mathsf{sk}.seed$, which is used to determine the secret keys for the underlying OTS scheme and FORS, and a PRF key $\mathsf{sk}.prf$ to generate a randomizer. The signing algorithm starts with computing the message digest md and the index idx by hashing the randomizer r, the public key pk, and the message m using the hash function H_{msg}. The index idx indicates the FORS key pair to be used. Finally, the signature consists of the randomizer r, a FORS signature on the message digest md, and the authentication path of the FORS public key in the hypertree which is referred to as a signature of the hypertree, i.e., σ_{HT}. The verification algorithm parses the signature and computes md and idx by evaluating the hash function H_{msg} on the randomizer, the public key and the message. Furthermore, it computes the FORS public key from the FORS signature, and verifies the authentication path of the FORS public key in the hypertree. That is, it uses the FORS public key and its authentication path to recompute the root and checks that value against the root denoted in the public key as $\mathsf{pk}.root$.

The security of SPHINCS$^+$ [BHK$^+$19] requires that the hash function H_{msg} has a property called

interleaved target subset resilience (ITSR). Intuitively, this states that it is infeasible for an attacker given an input to H_{msg} to find a second input to H_{msg} such that the corresponding second signature uses a particular leaf in a FORS forest, where this particular leaf was already used in the first signature. Specifically, if the resulting indices refer to the FORS forest at the same position, then the digest will refer to different leaves in each tree of this forest.

Let us start with showing that SPHINCS$^+$ achieves message-bound signatures.

Proposition 5.15. *The signature scheme SPHINCS$^+$ as described in Figure 12 achieves* MBS *if the hash function* H *is collision resistant and* H_{msg} *is interleaved target subset resilient.*

Proof. Let us assume a successful attacker with a public key pk that can craft a signature $\sigma_1 \leftarrow (r_1, \sigma_{FORS,1}, \sigma_{HT,1})$ which verifies for a message m_1 under pk and a signature $\sigma_2 \leftarrow (r_2, \sigma_{FORS,2}, \sigma_{HT,2})$ which verifies for a message m_2 under pk with $m_1 \neq m_2$ and $\sigma_1 = \sigma_2$. This implies $r_1 = r_2$.

To verify the signatures, the verifier first has to compute the respective message digests, i.e., it computes $(md_1 \| idx_1) \leftarrow H_{msg}(r_1, pk, m_1)$ and $(md_2 \| idx_2) \leftarrow H_{msg}(r_2, pk, m_2)$, respectively. Due to the ITSR property of H_{msg} it is infeasible for the attacker to find two different inputs to H_{msg} such that the same FORS key pair is used to sign the same message digest. Therefore, it cannot hold that $md_1 = md_2$ and $idx_1 = idx_2$. Firstly, we assume that $idx_1 \neq idx_2$. The signature $\sigma_{HT,1} = \sigma_{HT,2}$ needs to verify two different leaves of the same hypertree. By construction of Vf_{HT}, each node is hashed to the root of the hypertree with the sibling nodes given in the signature. Specifically, $node \leftarrow H(pk.seed, ADRS, leftChild, rightChild)$ where ADRS denotes the address of the parent node in the hypertree. Recall that the index of each node which is unique in the whole hypertree is given as argument when hashing through the forest to compute the FORS public key. For verification to accept, the resulting root nodes need to be identical. This implies finding a collision in H. Thus, a successful attacker against message-bound signatures can also break the collision resistance of H. Secondly, we consider the case that $md_1 \neq md_2$ and $idx_1 = idx_2$. In order for $\sigma_1 = \sigma_2$ to hold it must be that $\sigma_{FORS,1} = \sigma_{FORS,2}$. If the two FORS public keys extracted from the signature are different, verification of their authentication paths fail, i.e., Vf_{HT} fails, as described above. If the two FORS public keys extracted from the signature are identical, there has to be a collision while hashing the leaf nodes to the roots. Thus, a successful attacker can also break the collision resistance of H. $\square$

Unfortunately, we cannot provide formal proofs showing that SPHINCS$^+$ achieves M-S-UEO and NR under standard assumptions. In the following we provide some discussion arguing that we intuitively expect these notions to hold.

In order to achieve M-S-UEO, the scheme should at least satisfy S-CEO and S-DEO. Let us start with S-CEO and recall that the attacker needs to output a new public key pk$'$ under which a signature verifies its underlying message. Concretely for SPHINCS$^+$ this means that the attacker needs to output a public key where at least one of its components differ, i.e. pk$'$.$seed \neq$ pk.$seed$ or pk$'$.$root \neq$ pk.$root$. Let us consider that the root is identical and the seed differs. Similar to the previous proof, during the verification (while running Vf_{HT}) we require that the resulting root node is identical to the root node given in the public key, even tough we use a different seed in the hash function evaluation. Hence, this means we would find a collision in H. Next, let us assume that the seed is identical while the root differs. Thus, the verification algorithm obtains a different hash digest when evaluating H_{msg}. This change propagates at each step when hashing through the hypertree, leading to a different root node. (In case the same root node is computed, this corresponds to a collision in H.) If the attacker now changes the root node denoted in the public key, the root node computed by the verification algorithm changes yet again. This leads to a circularity in the argument since we provide H_{msg} exactly with the key that the root is already supposed to be part of. In case both components of pk$'$ differ, the seed and the root node, the verification algorithm obtains a different hash digest from H_{msg} as well. Since the seed is used as an argument to the hash function when hashing through the hypertree, the changes to the hash digest are at least as severe as in the previous case.

Therefore, the same argument applies here. Due to these observations, it seems infeasible that an attacker could succeed here. Hence, we intuitively expect that S-CEO should be satisfied.

A similar argument also holds for DEO with the only difference that it is required that the message differs. Hence the hash value of evaluating H_{msg} differs and we can make the same case distinction as for S-CEO. Therefore we also intuitively expect DEO to be satisfied.

We next argue why it is plausible that SPHINCS$^+$ provides non re-signability. Assume first that one could somehow infer the output of the evaluation of H_{msg}, i.e., $md\|idx$, from a valid signature. Then we could transform an attacker against non re-signability into an attacker against the Φ-non-malleability of H_{msg}: First, re-signing for the unknown message m under another key pk' means that $H_{msg}(r, \mathsf{pk}', m)$ must be related to the original hash value $H_{msg}(r, \mathsf{pk}, m)$ for a different input part $\mathsf{pk} \neq \mathsf{pk}'$. On the other hand, without being able to deduce $md\|idx$ (almost) entirely, the adversary cannot know which message digest to sign under which key pair. We assume that in such cases the adversary's signature is invalid with overwhelming probability. Thus, overall we expect NR to hold.

SPHINCS$^+$ comes with different instantiations for (tweakable versions of) the hash functions SHAKE-256, SHA256, and Haraka. Hence, applying our transformation for the former two it is conceivable that the derived scheme achieves all security properties; we are not aware of the underlying security properties of Haraka.

6   Conclusions

Our analysis shows that several NIST finalists do not achieve security properties beyond unforgeability that other modern schemes do. Providing these additional properties for all the candidates is likely to prevent attacks further down the line, and we see no substantial drawbacks in adapting the schemes (either directly or by our BUFF transformation) to achieve them. This suggests that it would be prudent for NIST to explicitly require these properties.

Acknowledgements

References

[ABB$^+$20]   Jean-Philippe Aumasson, Daniel J. Bernstein, Ward Beullens, Christoph Dobraunig, Maria Eichlseder, Scott Fluhrer, Stefan-Lukas Gazdag, Andreas Hülsing, Panos Kampanakis, Stefan Kölbl, Tanja Lange, Martin M. Lauridsen, Florian Mendel, Ruben Niederhagen, Christian Rechberger, Joost Rijneveld, Peter Schwabe, and Bas Westerbaan. SPHINCS+: Submission to the NIST post-quantum project, v.3. NIST Post-Quantum Cryptography Standardization Round 3 Submission, 10 2020. `https://sphincs.org/index.html`. (Cited on pages 17, 28, and 29.)

[ARS+15] Martin R. Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. Ciphers for MPC and FHE. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 430–454, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany. (Cited on page 27.)

[Aye15a] Andrew Ayer. ACME signature misuse vulnerability in draft-barnes-acme-04, 2015. (Cited on page 3.)

[Aye15b] Andrew Ayer. Duplicate Signature Key Selection Attack in Let's Encrypt, 2015. (Cited on page 3.)

[BCFW09] Alexandra Boldyreva, David Cash, Marc Fischlin, and Bogdan Warinschi. Foundations of non-malleable hash and one-way functions. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 524–541, Tokyo, Japan, December 6–10, 2009. Springer, Heidelberg, Germany. (Cited on page 7.)

[BCJZ20] Jacqueline Brendel, Cas Cremers, Dennis Jackson, and Mang Zhao. The provable security of Ed25519: Theory and practice. *IACR Cryptol. ePrint Arch.*, 2020:823, 2020. (Cited on pages 3, 8, 9, 10, 11, 14, and 36.)

[BDH11] Johannes A. Buchmann, Erik Dahmen, and Andreas Hülsing. XMSS - A practical forward secure signature scheme based on minimal security assumptions. In Bo-Yin Yang, editor, *Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011*, pages 117–129, Tapei, Taiwan, November 29 – December 2 2011. Springer, Heidelberg, Germany. (Cited on page 28.)

[BDK+21] Shi Bai, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Dilithium: Algorithm specifications and supporting documentation (version 3.1). NIST Post-Quantum Cryptography Standardization Round 3 Submission, 02 2021. https://pq-crystals.org/dilithium/index.shtml. (Cited on pages 17 and 19.)

[BFS11] Paul Baecher, Marc Fischlin, and Dominique Schröder. Expedient non-malleability notions for hash functions. In Aggelos Kiayias, editor, *Topics in Cryptology – CT-RSA 2011*, volume 6558 of *Lecture Notes in Computer Science*, pages 268–283, San Francisco, CA, USA, February 14–18, 2011. Springer, Heidelberg, Germany. (Cited on pages 7, 8, and 15.)

[BHAK15a] Richard Barnes, Jacob Hoffman-Andrews, and James Kasten. ACME Draft Barnes, 2015. (Cited on page 3.)

[BHAK15b] Richard Barnes, Jacob Hoffman-Andrews, and James Kasten. Automatic Certificate Management Environment (ACME), 2015. (Cited on page 3.)

[BHK+19] Daniel J. Bernstein, Andreas Hülsing, Stefan Kölbl, Ruben Niederhagen, Joost Rijneveld, and Peter Schwabe. The SPHINCS+ signature framework. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019: 26th Conference on Computer and Communications Security*, pages 2129–2146. ACM Press, November 11–15, 2019. (Cited on pages 28 and 29.)

[BK00] Joonsang Baek and Kwangjo Kim. Remarks on the unknown key share attacks. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 83(12):2766–2769, 2000. (Cited on page 8.)

[BL16]   Karthikeyan Bhargavan and Gaëtan Leurent. Transcript collision attacks: Breaking authentication in TLS, IKE and SSH. In *23rd Annual Network and Distributed System Security Symposium, NDSS 2016*. The Internet Society, 2016. (Cited on page 4.)

[BRS06]   Jens-Matthias Bohli, Stefan Röhrich, and Rainer Steinwandt. Key substitution attacks revisited: Taking into account malicious signers. *Int. J. Inf. Sec.*, 5(1):30–36, 2006. (Cited on page 8.)

[BWM99]   Simon Blake-Wilson and Alfred Menezes. Unknown key-share attacks on the station-to-station (STS) protocol. In Hideki Imai and Yuliang Zheng, editors, *PKC'99: 2nd International Workshop on Theory and Practice in Public Key Cryptography*, volume 1560 of *Lecture Notes in Computer Science*, pages 154–170, Kamakura, Japan, March 1–3, 1999. Springer, Heidelberg, Germany. (Cited on pages 3 and 8.)

[CDG⁺17]   Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, and Greg Zaverucha. Post-quantum zero-knowledge and signatures from symmetric-key primitives. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017: 24th Conference on Computer and Communications Security*, pages 1825–1842, Dallas, TX, USA, October 31 – November 2, 2017. ACM Press. (Cited on page 27.)

[CDG⁺20]   Melissa Chase, David Derler, Steven Goldfeder, Jonathan Katz, Vladimir Kolesnikov, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, Xiao Wang, and Greg Zaverucha. The Picnic signature scheme: Design document version 3.0. NIST Post-Quantum Cryptography Standardization Round 3 Submission, 09 2020. `https://microsoft.github.io/Picnic/`. (Cited on pages 17, 27, and 28.)

[CFMR⁺20] Antoine Casanova, Jean-Charles Faugère, Gilles Macario-Rat, Jacques Patarin, Ludovic Perret, and Jocelyn Ryckeghem. GeMSS: A Great Multivariate Short Signature. NIST Post-Quantum Cryptography Standardization Round 3 Submission, 2020. `https://www-polsys.lip6.fr/Links/NIST/GeMSS.html`. (Cited on pages 17 and 24.)

[CGN20]   Konstantinos Chalkias, François Garillot, and Valeria Nikolaenko. Taming the Many EdDSAs. In Thyla van der Merwe, Chris J. Mitchell, and Maryam Mehrnezhad, editors, *Security Standardisation Research - 6th International Conference, SSR 2020, London, UK, November 30 - December 1, 2020, Proceedings*, volume 12529 of *Lecture Notes in Computer Science*, pages 67–90. Springer, 2020. (Cited on page 9.)

[DCK⁺20]   Jintai Ding, Ming-Shing Chen, Matthias Kannwischer, Jacques Patarin, Albrecht Petzoldt, Dieter Schmidt, and Bo-Yin Yang. Rainbow: Algorithm specification and documentation the 3rd round proposal. NIST Post-Quantum Cryptography Standardization Round 3 Submission, 2020. `https://www.pqcrainbow.org/`. (Cited on pages 17 and 22.)

[DR09]   Thai Duong and Juliano Rizzo. Flickr's API Signature Forgery Vulnerability, 2009. (Retrieved November 2020). (Cited on page 4.)

[DRS04]   Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 523–540, Interlaken, Switzerland, May 2–6, 2004. Springer, Heidelberg, Germany. (Cited on page 6.)

[FHK+20]   Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Prest, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. Falcon: Fast-fourier lattice-based compact signatures over NTRU specifications v1.2. NIST Post-Quantum Cryptography Standardization Round 3 Submission, 10 2020. `https://falcon-sign.info/`. (Cited on pages 17 and 20.)

[FS87]   Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology – CRYPTO'86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194, Santa Barbara, CA, USA, August 1987. Springer, Heidelberg, Germany. (Cited on page 27.)

[GMO16]   Irene Giacomelli, Jesper Madsen, and Claudio Orlandi. ZKBoo: Faster zero-knowledge for Boolean circuits. In Thorsten Holz and Stefan Savage, editors, *USENIX Security 2016: 25th USENIX Security Symposium*, pages 1069–1083, Austin, TX, USA, August 10–12, 2016. USENIX Association. (Cited on page 27.)

[GMR88]   Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988. (Cited on page 3.)

[GPV08]   Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 197–206, Victoria, BC, Canada, May 17–20, 2008. ACM Press. (Cited on page 20.)

[HLR07]   Chun-Yuan Hsiao, Chi-Jen Lu, and Leonid Reyzin. Conditional computational entropy, or toward separating pseudoentropy from compressibility. In Moni Naor, editor, *Advances in Cryptology – EUROCRYPT 2007*, volume 4515 of *Lecture Notes in Computer Science*, pages 169–186, Barcelona, Spain, May 20–24, 2007. Springer, Heidelberg, Germany. (Cited on page 6.)

[IKOS09]   Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge proofs from secure multiparty computation. *SIAM J. Comput.*, 39(3):1121–1152, 2009. (Cited on page 27.)

[JCCS19]   Dennis Jackson, Cas Cremers, Katriel Cohn-Gordon, and Ralf Sasse. Seems legit: Automated analysis of subtle attacks on protocols that use signatures. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019: 26th Conference on Computer and Communications Security*, pages 2165–2180. ACM Press, November 11–15, 2019. (Cited on pages 3, 8, 9, and 11.)

[KKW18]   Jonathan Katz, Vladimir Kolesnikov, and Xiao Wang. Improved non-interactive zero knowledge with applications to post-quantum signatures. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018: 25th Conference on Computer and Communications Security*, pages 525–537, Toronto, ON, Canada, October 15–19, 2018. ACM Press. (Cited on page 27.)

[KZ20]   Daniel Kales and Greg Zaverucha. Improving the performance of the Picnic signature scheme. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(4):154–188, 2020. `https://tches.iacr.org/index.php/TCHES/article/view/8680`. (Cited on page 27.)

[Lyu09]   Vadim Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume

5912 of *Lecture Notes in Computer Science*, pages 598–616, Tokyo, Japan, December 6–10, 2009. Springer, Heidelberg, Germany. (Cited on page 17.)

[MS04]   Alfred Menezes and Nigel Smart. Security of signature schemes in a multi-user setting. In *Designs, Codes and Cryptography*, volume 33, pages 261–274. Springer, Heidelberg, Germany, 2004. (Cited on pages 3, 8, 14, and 27.)

[Nat15a]   National Institute of Standards and Technology (NIST). SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. `https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf`, Aug 2015. (Cited on page 4.)

[Nat15b]   National Institute of Standards and Technology (NIST). Post-quantum cryptography. `https://csrc.nist.gov/projects/post-quantum-cryptography`, Aug 19, 2015. (Cited on page 17.)

[PCG01]   Jacques Patarin, Nicolas T. Courtois, and Louis Goubin. Quartz, 128-bit long digital signatures. In David Naccache, editor, *Topics in Cryptology - CT-RSA 2001, The Cryptographer's Track at RSA Conference 2001, San Francisco, CA, USA, April 8-12, 2001, Proceedings*, volume 2020 of *Lecture Notes in Computer Science*, pages 282–297. Springer, 2001. (Cited on page 24.)

[PS05]   Thomas Pornin and Julien P. Stern. Digital signatures do not guarantee exclusive ownership. In John Ioannidis, Angelos Keromytis, and Moti Yung, editors, *ACNS 05: 3rd International Conference on Applied Cryptography and Network Security*, volume 3531 of *Lecture Notes in Computer Science*, pages 138–150, New York, NY, USA, June 7–10, 2005. Springer, Heidelberg, Germany. (Cited on pages 3, 8, 13, and 14.)

[SPMS02]   Jacques Stern, David Pointcheval, John Malone-Lee, and Nigel P. Smart. Flaws in applying proof methodologies to signature schemes. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 93–110, Santa Barbara, CA, USA, August 18–22, 2002. Springer, Heidelberg, Germany. (Cited on page 9.)

[Unr15]   Dominique Unruh. Non-interactive zero-knowledge proofs in the quantum random oracle model. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 755–784, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany. (Cited on page 27.)

[Zav20]   Greg Zaverucha. The Picnic Signature Scheme: Specification Version 3.0. NIST Post-Quantum Cryptography Standardization Round 3 Submission, 09 2020. `https://microsoft.github.io/Picnic/`. (Cited on pages 17 and 27.)

A   Auxiliary Definitions

A.1   Unforgeable Signature Schemes

Definition A.1. *Let Π be a digital signature scheme. We say that Π is* existentially unforgeable under chosen-message attack *if, for every* PPT *algorithm $\mathcal{A}$, there exists a negligible function $\mu\colon \mathbb{N} \to \mathbb{R}$ such that, for every $\lambda \in \mathbb{N}$, it holds that*

$$\Pr[\mathbf{Exp}_{\Pi,\mathcal{A}}^{\mathsf{EUF\text{-}CMA}}(\lambda)] \leq \mu(\lambda),$$

where $\mathbf{Exp}_{\Pi,\mathcal{A}}^{\mathsf{EUF\text{-}CMA}}(\lambda)$ is defined in Figure 13.

$$
\begin{array}{ll}
\textbf{Exp}_{\Pi,\mathcal{A}}^{\mathsf{EUF\text{-}CMA}}(\lambda): & \mathsf{Sig}(\mathsf{sk}, m): \\
\hline
11: \quad \mathcal{Q} \leftarrow \emptyset & 21: \quad \sigma \leftarrow\!\$\, \mathsf{Sig}(\mathsf{sk}, m) \\
12: \quad (\mathsf{sk}, \mathsf{pk}) \leftarrow\!\$\, \mathsf{KGen}(1^\lambda) & 22: \quad \mathcal{Q} \leftarrow \mathcal{Q} \cup \{m\} \\
13: \quad (m', \sigma') \leftarrow\!\$\, \mathcal{A}^{\mathsf{Sig}(\mathsf{sk}, \cdot)}(\mathsf{pk}) & 23: \quad \textbf{return } \sigma \\
14: \quad d \leftarrow \mathsf{Vf}(\mathsf{pk}, m', \sigma') & \\
15: \quad \textbf{return } [d = 1 \ \wedge \ m' \notin \mathcal{Q}] &
\end{array}
$$

Figure 13: Definition of the experiment $\textbf{Exp}_{\Pi,\mathcal{A}}^{\mathsf{EUF\text{-}CMA}}(\lambda)$ from Definition A.1.

A.2 Strong Universal Exclusive Ownership (S-UEO)

The union of our new variants of exclusive ownership, S-CEO and S-DEO, is strong universal exclusive ownership, as introduced by Brendel et al. in [BCJZ20].

$$
\begin{array}{ll}
\textbf{Exp}_{\Pi,\mathcal{A}}^{\mathsf{S\text{-}UEO}}(\lambda): & \mathsf{Sig}(\mathsf{sk}, m): \\
\hline
11: \quad \mathcal{Q} \leftarrow \emptyset & 21: \quad \sigma \leftarrow\!\$\, \mathsf{Sig}(\mathsf{sk}, m) \\
12: \quad (\mathsf{sk}, \mathsf{pk}) \leftarrow\!\$\, \mathsf{KGen}(1^\lambda) & 22: \quad \mathcal{Q} \leftarrow \mathcal{Q} \cup \{(m, \sigma)\} \\
13: \quad (m', \sigma', \mathsf{pk}') \leftarrow\!\$\, \mathcal{A}^{\mathsf{Sig}(\mathsf{sk}, \cdot)}(\mathsf{pk}) & 23: \quad \textbf{return } \sigma \\
14: \quad d \leftarrow \mathsf{Vf}(\mathsf{pk}', m', \sigma') & \\
15: \quad \textbf{return } \big[d = 1 \ \wedge \ (\exists m^*: (m^*, \sigma') \in \mathcal{Q}) \ \wedge \ \mathsf{pk}' \neq \mathsf{pk}\big] &
\end{array}
$$

Figure 14: Definition of the experiment $\textbf{Exp}_{\Pi,\mathcal{A}}^{\mathsf{S\text{-}UEO}}(\lambda)$ from Definition A.2.

Definition A.2. *Let Π be a digital signature scheme. We say that Π provides* strong universal exclusive ownership *(S-UEO) if, for every* PPT *algorithm $\mathcal{A}$, there exists a negligible function $\mu\colon \mathbb{N} \to \mathbb{R}$ such that, for every $\lambda \in \mathbb{N}$, it holds that*

$$\Pr[\textbf{Exp}_{\Pi,\mathcal{A}}^{\mathsf{S\text{-}UEO}}(\lambda)] \leq \mu(\lambda),$$

where $\textbf{Exp}_{\Pi,\mathcal{A}}^{\mathsf{S\text{-}UEO}}(\lambda)$ is defined in Figure 14.

B Further Details about the Relationships

In this part of the appendix, we present the remaining relationships between the notions as started in Section 4.3. However, before providing them we show that our new notions S-CEO and S-DEO are equivalent to S-UEO:

Proposition B.1. *A digital signature scheme Π is* S-CEO *and* S-DEO *if and only if it is* S-UEO.

Proof. Follows from inspecting the winning condition in S-UEO. Let m', σ', pk' be the adversary's choice in the S-UEO game. Then $\mathsf{pk}' \neq \mathsf{pk}$ and σ' must have been the result of some query m^* to the signing oracle. If $m^* = m'$ then we break S-CEO, if $m^* \neq m'$ then we break S-DEO. Hence security under S-CEO and S-DEO implies S-UEO-security. Vice versa, any successful attack against the more restrictive conditions in S-CEO and S-DEO means a successful attack against S-UEO. $\qquad\square$

Next, we show that M-S-UEO implies S-UEO:

Proposition B.2. *If a digital signature scheme* Π *is* M-S-UEO *it is also* S-UEO.

Proof. If there exists an efficient PPT algorithm $\mathcal{A}$ that breaks S-UEO of Π with non-negligible probability we can use it to construct an efficient PPT algorithm $\mathcal{B}$ that breaks M-S-UEO of Π with non-negligible probability. First, $\mathcal{B}$ samples a key pair $(\mathsf{pk}, \mathsf{sk})$ using the KGen algorithm and initiates $\mathcal{A}$ with input pk. For all queries to the signing oracle that $\mathcal{A}$ makes, $\mathcal{B}$ answers them as the oracle would with the knowledge of the secret key sk and saves the message signature pairs in $\mathcal{Q}$. When $\mathcal{A}$ outputs $(m', \sigma', \mathsf{pk}')$, algorithm $\mathcal{B}$ finds m such that (m, σ') is contained in $\mathcal{Q}$. Finally, $\mathcal{B}$ outputs $(m, m', \sigma', \mathsf{pk}, \mathsf{pk}')$. Using this strategy, $\mathcal{B}$ wins its game if $\mathcal{A}$ wins its own game. $\qquad\square$

We discuss below, in the more general context of the other properties, that M-S-UEO is strictly stronger than S-UEO (Proposition B.5).

The propositions below establish that except for the implications above all other properties are independent. To this end we start with a signature scheme having any of the properties $\mathcal{P}$ we are interested in, except for one, and show that we can modify the scheme into one which preserves all properties in $\mathcal{P}$ but is for sure vulnerable against the excluded property. It follows that the excluded property cannot follow from the other properties, no matter in which combination. The propositions below make the implicit assumption that the message space is non-trivial (for simplicity, assume that it is $\{0,1\}^*$), and that key generation when run twice outputs distinct public keys with non-negligible probability.

Proposition B.3. *If there is a digital signature scheme which has the properties* $\mathcal{P} \subseteq \{$EUF-CMA, S-CEO, MBS, NR$\}$, *then there is also one which has the same properties* $\mathcal{P}$ *but not* S-DEO.

Note that since M-S-UEO implies S-UEO and thus S-DEO, and we build a scheme where S-DEO does not hold, we cannot hope that the scheme preserves M-S-UEO.

Proof. Assume that we have a scheme $\Pi = (\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$ which has properties $\mathcal{P}$. Build the following modified signature scheme $\Pi^{\neg\mathsf{S\text{-}DEO}}$. In this scheme the public key carries a redundant bit b which is xored to the last message bit before signing (and where this is bit is also appended to the signature):

$\Pi^{\neg\mathsf{S\text{-}DEO}}.\mathsf{KGen}(1^\lambda)$:	$\Pi^{\neg\mathsf{S\text{-}DEO}}.\mathsf{Sig}(\mathsf{sk}\|b, m\|a)$:	$\Pi^{\neg\mathsf{S\text{-}DEO}}.\mathsf{Vf}(\mathsf{pk}\|b, m\|a, (\sigma, p, c))$:
11: $(\mathsf{sk}, \mathsf{pk}) \leftarrow\!\!{\scriptstyle\$}\ \Pi.\mathsf{KGen}(1^\lambda)$	21: $\sigma \leftarrow\!\!{\scriptstyle\$}\ \Pi.\mathsf{Sig}(\mathsf{sk}, m\|(a \oplus b))$	31: $d \leftarrow \Pi.\mathsf{Vf}(\mathsf{pk}, m\|(a \oplus b), \sigma)$
12: $b \leftarrow\!\!{\scriptstyle\$}\ \{0,1\}$	22: **return** $(\sigma, \mathsf{pk}, (a \oplus b))$	32: **return** $[d \wedge (c = a \oplus b) \wedge \mathsf{pk} = p]$
13: **return** $(\mathsf{sk}\|b, \mathsf{pk}\|b)$		

The scheme inherits correctness of the original scheme.

We first argue that the scheme $\Pi^{\neg\mathsf{S\text{-}DEO}}$ does *not* have property S-DEO. To see this note that an adversary, upon receiving a public key $\mathsf{pk}\|b$ queries its signing oracle about an arbitrary message $m\|a$ to get a signature (σ, pk, c). It then outputs $m' \leftarrow m\|\bar{a}$, $\sigma' \leftarrow (\sigma, \mathsf{pk}, c)$, and $\mathsf{pk}' \leftarrow \mathsf{pk}\|\bar{b}$. Note that since we flip both last bits in the message and the public key, the signature σ' is also valid for these two values. And yet m' is different from the signed message $m\|a$ which produced that signature, and so is the public key pk' from $\mathsf{pk}\|b$. Hence, our adversary breaks S-DEO with probability 1.

We next argue that the scheme $\Pi^{\neg\mathsf{S\text{-}DEO}}$ has the property S-CEO (unconditionally, even if Π does not have this property). Note that the adversary's task here is to find a new key pk' such that a signature-message pair (m', σ') obtained in a query for $\mathsf{pk}\|b$ is also valid under pk'. In our scheme all except for the last bit of the public key appear in a signature, such that this can only hold if $\mathsf{pk}' = \mathsf{pk}\|\bar{b}$. But for any signed message $m' = m\|a$ the bit $a \oplus b$ also becomes part of the signature σ' (and is checked for

verification), such that this bit can never match $a \oplus \bar{b}$ as required for pk'. Hence, none of the pairs (m, σ) can verify under pk' and thus S-CEO holds.

Property EUF-CMA is preserved because our modification for the signed message, $m\|a \mapsto m\| (a \oplus b)$, is an efficient bijection given the public key $\mathsf{pk}\|b$, and the other appended data in a signature are computable from the public information $m\|a$ and $\mathsf{pk}\|b$. Hence, we can easily give a black-box reduction from EUF-CMA of the modified scheme to the one of the original scheme.

If the original scheme obeys message-bound signatures MBS then so does the modified scheme $\Pi^{\neg\text{S-DEO}}$. If an adversary is able to find $\mathsf{pk}\|b$, $m\|a \neq m'\|a'$ and a signature (σ, pk, c) such that both messages verify under the public key, then the distinct messages $m\|(a \oplus b) \neq m'\|(a' \oplus b)$ must in particular both verify for σ under pk. It follows that we straightforwardly also get an adversary against MBS of the original scheme.

Finally, assume that Π has the non re-signability property NR. Since we can guess the last message bit for the unknown message with probability $\frac{1}{2}$ we can easily reduce an attacker pair $(\mathcal{D}, \mathcal{A})$ against the modified scheme into one $(\mathcal{D}, \mathcal{B})$ against the original scheme: Given a public key $\mathsf{pk}, \mathsf{aux}$ and signature σ for the original scheme algorithm $\mathcal{B}$ appends pk and a random bit \$ to σ, and then runs $\mathcal{A}$ on $\mathsf{pk}\|b$ for random b, the augmented signature, and aux. It strips off the augmented part from $\mathcal{A}$'s signature σ' and also the final bit of the key pk' to get a forgery for the original scheme. The advantage drops by a factor at most $\frac{1}{2}$ for guessing the message bit. $\qquad\square$

Proposition B.4. *If there is a digital signature scheme which has the properties $\mathcal{P} \subseteq \{\mathsf{EUF\text{-}CMA}, \mathsf{M\text{-}S\text{-}UEO},$ $\mathsf{S\text{-}CEO}, \mathsf{S\text{-}DEO}, \mathsf{NR}\}$, then there is also one which has the same properties $\mathcal{P}$ but not MBS.*

Proof. Take the scheme $\Pi = (\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$ which has properties $\mathcal{P}$ and transform it into the signature scheme $\Pi^{\neg\text{MBS}}$ where special signatures are accepted for some special public keys pk (but where the genuine key generation and signature algorithms never output such values):

$\Pi^{\neg\text{MBS}}.\mathsf{KGen}(1^\lambda)$:	$\Pi^{\neg\text{MBS}}.\mathsf{Sig}(\mathsf{sk}, m)$:	$\Pi^{\neg\text{MBS}}.\mathsf{Vf}(\mathsf{pk}\|b, m\|a, \sigma\|c)$:
11 : $(\mathsf{sk}, \mathsf{pk}) \leftarrow\!\!\$\, \Pi.\mathsf{KGen}(1^\lambda)$	21 : $\sigma \leftarrow\!\!\$\, \Pi.\mathsf{Sig}(\mathsf{sk}, m)$	31 : **if** $b = 0 \wedge c = 0$ **then**
12 : **return** $(\mathsf{sk}, \mathsf{pk}\|1)$	22 : **return** $\sigma\|1$	32 : $\quad$ **return** $[\sigma = m = \mathsf{pk}]$
		33 : **else**
		34 : $\quad d \leftarrow \Pi.\mathsf{Vf}(\mathsf{pk}, m\|a, \sigma)$
		35 : $\quad$ **return** $[d \wedge b = c]$

The scheme inherits correctness of the original scheme.

We first show how to break property MBS. For this the adversary outputs messages $m_1 \leftarrow 0$ and $m_2 \leftarrow 1$ together with $\sigma\|c \leftarrow 0$ and $\mathsf{pk}\|b \leftarrow 0$. Note that the modified verification algorithm accepts the signature ending with 0 for m_1 and m_2 for the public key ending with 0. The reason is that, pruning the last bit of σ, m_1, m_2, in all cases yields the empty string pk.

We next argue that EUF-CMA is preserved because this property only looks at "good" public keys, ending with bit 1. Since we can simulate modified signatures and the modified public key by appending 1, and verification only succeeds if the signature in the forgery also ends with the redundant bit 1, we immediately get a reduction to the corresponding security property of the underlying scheme.

For S-CEO and S-DEO note that any signature $\sigma\|1$ generated by the signing algorithm for genuine key $\mathsf{pk}\|1$ carries a 1-bit at the end. Hence, if the adversary against either of the two properties outputs a key $\mathsf{pk}'\|0$, then none of these signatures $\sigma\|1$ can make the verifier accept. This means that the adversary cannot win S-CEO nor S-DEO against the modified scheme for such values.

If, on the other hand, the S-CEO- or S-DEO-adversary chooses a public key $\mathsf{pk}'\|1$, then the adversary's signature $\sigma'\|1$ must end with 1, too, in order to succeed. In addition, the verifier in $\Pi^{\neg\text{MBS}}$ checks the data (without the extra bits) against the original scheme. We can therefore give a black-box reduction against

the original scheme which appends 1's to the public key and all the requested signatures, and prunes the attacker's outputs $\sigma\|1$ and $\mathsf{pk}'\|1$ by the final bits to win against the original scheme. This holds for either property S-CEO or S-DEO.

For M-S-UEO note that if the adversary picks two public keys ending with 0 then any signature $\sigma\|0$ can only be valid for one of the two keys (because σ must equal the public key without the trailing 0-bit in this case), and any signature $\sigma\|1$ is invalid for such keys. Moreover, for public keys with a different trailing bit b the signature $\sigma\|c$ cannot be valid for both keys since verification checks that $b = c$. In the remaining case that the public keys end with 1 and the signature $\sigma\|c$, too, one straightforwardly derives a contradiction to the M-S-UEO security of the underlying scheme.

The final step is to argue that NR is preserved. There are two cases: Either the adversary against the modified scheme uses a public key $\mathsf{pk}'\|0$ ending with 0, or it uses a "regular" key $\mathsf{pk}'\|1$. In the latter case the signature $\sigma'\|1$ must also end with 1 for the verifier to accept, and as in the case of S-CEO and S-DEO we immediately get a reduction to the NR-property of the original scheme Π (by appending and pruning the extra bits). Note that $\mathsf{pk}'\|1 \neq \mathsf{pk}\|1$ implies that also $\mathsf{pk}' \neq \mathsf{pk}$ such that the pruned key is still different from the given one.

Assume now that the NR-adversary uses $\mathsf{pk}'\|0$. Then for the verifier of scheme $\Pi^{\neg\mathsf{MBS}}$ to accept, the signature $\sigma'\|0$ must coincide with the (unknown) message m, except for the message's last bit. But then we immediately get a reduction to NR of the original scheme: Our reduction adds the extra bit 1 to the public key and the signature and hands it over to $\mathcal{A}$ against $\Pi^{\neg\mathsf{MBS}}$. If $\mathcal{A}$ then outputs $\mathsf{pk}'\|0$ and $\sigma'\|0$, then we can guess the missing bit of the message by setting it to a random bit \$, and can then recover the unknown message $\sigma'\|\$$ in clear. Our reduction can then generate another key pair of Π and sign this message with $\Pi.\mathsf{Sig}$. Its success probability is only a factor $\frac{1}{2}$ smaller than that of $\mathcal{A}$ (with the success strategy of using $\mathsf{pk}'\|0$). $\qquad\square$

Proposition B.5. *If there is a digital signature scheme which has the properties* $\mathcal{P} \subseteq \{\mathsf{EUF\text{-}CMA}, \mathsf{S\text{-}CEO},$ $\mathsf{S\text{-}DEO}, \mathsf{MBS}, \mathsf{NR}\}$, *then there is also one which has the same properties* $\mathcal{P}$ *but not* M-S-UEO.

Proof. Modify the original scheme $\Pi = (\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$ with properties $\mathcal{P}$ into the signature scheme $\Pi^{\neg\mathsf{M\text{-}S\text{-}UEO}}$:

$\Pi^{\neg\mathsf{M\text{-}S\text{-}UEO}}.\mathsf{KGen}(1^\lambda)$:	$\Pi^{\neg\mathsf{M\text{-}S\text{-}UEO}}.\mathsf{Sig}(\mathsf{sk}, m)$:	$\Pi^{\neg\mathsf{M\text{-}S\text{-}UEO}}.\mathsf{Vf}(\mathsf{pk}\|b, m, \sigma\|c)$:
11 : $(\mathsf{sk}, \mathsf{pk}) \leftarrow\!\!\$\ \Pi.\mathsf{KGen}(1^\lambda)$	21 : $\sigma \leftarrow\!\!\$\ \Pi.\mathsf{Sig}(\mathsf{sk}, m)$	31 : **if** $b = 0$ **then**
12 : **return** $(\mathsf{sk}, \mathsf{pk}\|1)$	22 : **return** $\sigma\|1$	32 : $\quad$ **return** $[c = 0 \wedge m = \mathsf{pk}]$
		33 : **else**
		34 : $\quad d \leftarrow \Pi.\mathsf{Vf}(\mathsf{pk}, m, \sigma)$
		35 : $\quad$ **return** $[d \wedge c = 1]$

The scheme inherits correctness of the original scheme.

We first show that the new scheme does not have property M-S-UEO. For this note that the adversary can simply output $\sigma \leftarrow 0$ and $\mathsf{pk}_1\|0$, $\mathsf{pk}_2\|0$ with $m_1 \leftarrow \mathsf{pk}_1$ and $m_2 \leftarrow \mathsf{pk}_2$ for $\mathsf{pk}_1 \neq \mathsf{pk}_2$. Verification then accepts both message-key pairs for the signature in the exceptional branch.

We observe that EUF-CMA is preserved because this property only works with genuine public keys ending with bit 1. That is, we can simulate the modified signature and key generation process by appending 1's, and verification only succeeds if the key and the signature in the forgery also end with the redundant bit 1. This easily gives a reduction to the corresponding security property of the underlying scheme.

As for S-CEO and S-DEO we remark that any signature $\sigma\|1$ generated by the signing algorithm for genuine key $\mathsf{pk}\|1$ has a trailing 1-bit. Hence, if the adversary against either of the properties uses a key $\mathsf{pk}'\|0$, then none of these signatures $\sigma\|1$ can make the verifier in the modified scheme accept. This means

that the adversary cannot win S-CEO nor S-DEO against the modified scheme for such values. If the adversary mounts an attack against either S-CEO or S-DEO and eventually outputs a key $pk'\|1$, then the signature (with a final 1-bit) is checked under the original scheme (without the trailing 1-bits). In this case we immediately get a reduction to the corresponding property of the underlying scheme.

We finally show that NR is preserved. There are again two cases: Either the adversary against the modified scheme uses an exceptional public key $pk'\|0$ ending with 0, or it uses a "regular" key $pk'\|1$. In the latter case the signature $\sigma'\|1$ must also end with 1 for the verifier to accept, and once more we can easily derive a reduction to the NR-property of the original scheme Π, appending and pruning the extra bits. Here we again use that $pk'\|1 \neq pk\|1$ implies $pk' \neq pk$ and therefore that the pruned key is different from the given one.

Assume next that the NR-adversary uses $pk'\|0$ in its attempt. Then the verifier only accepts if the signature is $\sigma'\|0$ and the unknown message m is of the form pk'. In other words, the NR-adversary must be able to predict m via pk'. Then again we get a reduction to NR of the original scheme. For this the reduction appends the bit 1 to the public key and the signature and hands it over to $\mathcal{A}$ against $\Pi^{\neg\text{M-S-UEO}}$. If $\mathcal{A}$ then outputs $pk'\|0$ and $\sigma'\|0$, then we can recover the hidden message as $m = pk'$. The reduction can then generate another key pair of Π and sign this message with $\Pi.\text{Sig}$. Its success probability is exactly that of $\mathcal{A}$ (with the success strategy of using $pk'\|0$).

Since the adversary $\mathcal{A}$ against NR must use either strategy in at least half of the successful attack executions we get that one of the two reductions above must succeed against the original scheme (with a loss in the success probability of at most $\frac{1}{2}$). □

Proposition B.6. *If there is a digital signature scheme which has the properties $\mathcal{P} \subseteq \{$EUF-CMA, M-S-UEO, S-CEO, S-DEO, MBS$\}$, then there is also one which has the same properties $\mathcal{P}$ but not NR.*

Proof. This claim is easy to achieve: append the message to the signature and let the verifier also check that the message has been appended correctly. Then the adversary can break NR by reading off the message from the signature, generating another key pair (sk', pk'), and signing the message under pk'.

All other properties are preserved, including correctness, since the security does not depend on the confidentiality of the message. Note that the modified scheme now has properties S-DEO and MBS unconditionally, even if Π did not have these properties before. But this, of course, does not violate our claim. □

Proposition B.7. *If there is a digital signature scheme which has the properties $\mathcal{P} \subseteq \{$M-S-UEO, S-CEO, S-DEO, MBS, NR$\}$, then there is also one which has the same properties $\mathcal{P}$ but not EUF-CMA.*

Proof. Transform the scheme $\Pi = (\text{KGen}, \text{Sig}, \text{Vf})$ with properties $\mathcal{P}$ into $\Pi^{\neg\text{EUF-CMA}}$:

$\Pi^{\neg\text{EUF-CMA}}.\text{KGen}(1^\lambda)$:	$\Pi^{\neg\text{EUF-CMA}}.\text{Sig}((sk, pk), m)$:	$\Pi^{\neg\text{EUF-CMA}}.\text{Vf}(pk, m, \sigma\|c)$:
11 : $(sk, pk) \leftarrow\$ \Pi.\text{KGen}(1^\lambda)$	21 : **if** $m = pk$ **then**	31 : **if** $m = pk$ **then**
12 : **return** $((sk, pk), pk)$	22 : **return** $pk\|0$	32 : **return** $[\sigma = pk$
	23 : **else**	$\wedge\, c = 0]$
	24 : $\sigma \leftarrow\$ \Pi.\text{Sig}(sk, m)$	33 : **else**
	25 : **return** $\sigma\|1$	34 : $d \leftarrow \Pi.\text{Vf}(m, \sigma, pk)$
		35 : **return** $[d \wedge c = 1]$

The scheme inherits correctness of the original scheme and because the signature $pk\|0$ for $m = pk$ is accepted by Vf by construction.

It is easy to break the unforgeability property EUF-CMA in the modified scheme. An adversary can immediately output $m = pk$ with signature $pk\|0$ as a successful forgery, after having received the public key and without making any signature queries.

The modified scheme preserves S-CEO and S-DEO, though. If the adversary uses the exceptional case $m' = \mathsf{pk}'$ and, then necessarily, $\sigma' = \mathsf{pk}'\|0$ for $\mathsf{pk}' \neq \mathsf{pk}$ in a S-CEO or S-DEO attack, then it is clear that a signature query about $m = \mathsf{pk}$ cannot yield this signature, and neither can any other query about $m \neq \mathsf{pk}$ because of the extra 1-bit at the end of those signatures. In any other case, if $m' \neq \mathsf{pk}'$ does not trigger the exceptional case, the signature must have a trailing 1-bit and in particular cannot have appeared in a signature query about $m = \mathsf{pk}$. But then it follows that the adversarial message-signature pair must have appeared in a "regular" signature query. But in this case it is again easy to give a reduction to the corresponding property of the starting scheme (by modifying only the signing reply for $m = \mathsf{pk}$ to $\mathsf{pk}\|0$, and appending 1 to each other signature).

Concerning M-S-UEO note that if the adversary outputs $\sigma\|0$ then a successful verification can only happen in the extra branch. But then the condition on $c = 0$ there stipulates that both message-key pairs must satisfy $m_1 = \mathsf{pk}_1$ and $m_2 = \mathsf{pk}_2$. Since one must additionally have that the signature part σ equals the public key in this case, but $\mathsf{pk}_1 \neq \mathsf{pk}_2$ for a successful attack against M-S-UEO, the adversary cannot succeed in this case. In case the adversary's signature is of the form $\sigma\|1$ then the adversary can only win if σ verifies for both m_1, pk_1 and m_2, pk_2, giving immediately an attack against the M-S-UEO property of the underlying scheme.

As for MBS, if the signature is of the form $\mathsf{pk}\|0$ for the chosen key pk, then there is only one unique message which verifies. Hence the adversary cannot break MBS for such signatures. In any other case, a signature of the form $\sigma\|1$ with two messages $m_1 \neq m_2$ for pk, also constitutes a successful attack against the underlying scheme, since $m_1 = \mathsf{pk}$ (or $m_2 = \mathsf{pk}$) cannot pass verification under the modified scheme for such signatures. We conclude that the signature σ under m_1 and m_2 must be also valid with respect to $\Pi.\mathsf{Vf}$.

Finally, for NR we note that, because of the min-entropy of $\mathcal{D}$, the chance that we sample $m = \mathsf{pk}$ is negligible. Hence, we can assume that this does not happen. Conditioning on this there are two cases: Either the adversary outputs $\sigma' = \mathsf{pk}'\|0$ and wins because $m = \mathsf{pk}'$, or the adversary outputs a signature ending with 1. In the latter case the adversary can only win if it already breaks the NR-property of Π, i.e., it is straightforward to turn this adversary into a reduction against scheme Π.

In the other case, if the adversary against NR succeeds with $\sigma' = \mathsf{pk}'\|0$ then it must have predicted $m = \mathsf{pk}'$ from σ and pk. In this case we can build an adversary against NR of the original scheme which takes the adversary's output, samples another key pair $(\mathsf{sk}^*, \mathsf{pk}^*)$, sets $m^* = \mathsf{pk}'$, and signs this message m^* under the key sk^*. It outputs m^*, pk^* and the derived signatures. This breaks NR of the underlying scheme if the adversary against the modified scheme wins with this strategy. $\qquad\square$

C   Summary of major changes

- **version 1.0 - December 2020**: Initial release

- **version 1.1 - March 2021**:

    – Differentiated exclusive ownership notions more precisely, now including S-CEO, S-DEO, UEO, S-UEO, and M-S-UEO

    – added M-S-UEO to relationships

    – discussion of S-CEO and S-DEO of Rainbow variants

    – proof details for attacks against NR and MBS of GeMSS added and corrected