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

Version 1.0, December 2020

Cas Cremers*, Samed Düzlü†, Rune Fiedler‡, Marc Fischlin‡, and Christian Janson‡
*CISPA Helmholtz Center for Information Security, Germany
†QPC, Technische Universität Darmstadt, Germany
‡Cryptoplexity, Technische Universität Darmstadt, Germany

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.

I. Introduction

For digital signature schemes, there are two classical security notions: EUF-CMA, existential unforgeability [1], 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 [2], [3], [4] 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** [2] (which generalizes earlier notions of Duplicate-Signature Key Selection (DSKS) attacks [5], [6]): the property that a signature only verifies under a single public key. For example, an early version of Let's Encrypt's ACME protocol [7], [8] 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 [9], [10] "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 [3] 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. In [3] 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.

The third property is **non re-signability** [3] meaning that one cannot produce a signature under another key given a signature for some unknown message m. For most signature schemes the short signature must lose information about the message, and one might therefore 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, which enables a similar attack on DRKey/OPT. This property was first proposed and defined in the symbolic model in [3]. 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 [4] 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 [11] and TLS, IKE, and SSH [12]. 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."* [13, 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 (CEO and 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 (*B*eyond *U*n*F*orgeability *F*eatures) 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 I. 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 II we introduce notation and further preliminaries. Section III overviews previous work on security properties of signatures beyond unforgeability. In Section IV we present our main theoretical results. In Section V 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 VI.

II. Preliminaries

A. 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

Table I: Several NIST PQ Signature scheme Round 3 candidates and alternate 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 refer to the detailed analyses in the table.
The "Conclusion" column summarizes for each scheme: ✓ indicates all properties hold. For schemes with ✗ or ✦, our generic transformation from Section IV-C provably provides all properties at the cost of a slight increase in signature size (see Figure 5).

	Scheme		exclusive ownership CEO (Def. III.1) & DEO (Def. III.2)		message-bound signatures MBS (Def. III.3)		no re-signing without message NR (Def. IV.1)		Conclusion
main	CRYSTALS-Dilithium		✓	Prop. V.1	✓	Prop. V.1	✓	Prop. V.1	✓
main	FALCON		✗	Prop. V.5	✓	Prop. V.3	✗	Prop. V.6	✗
main	Rainbow	Standard	✗	Prop. V.9	✓	Prop. V.7	✗	Prop. V.8	✗
main	Rainbow	CZ & Compressed	✦	Sec. V-C	✓	Prop. V.7	✗	Prop. V.8	✗
alternate	GeMSS		✗	Prop. V.10	✗	Prop. V.10	✗	Prop. V.10	✗
alternate	Picnic		✓	Prop. V.11	✓	Prop. V.11	✓	Prop. V.11	✓
alternate	SPHINCS$^+$		✦	Sec. V-D3	✓	Prop. V.12	✦	Sec. V-D3	✦

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 $\overline{b}$. 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 $s \xleftarrow{\$} S$ to denote that the string s was sampled uniformly at random from S. We also use this notation $y \xleftarrow{\$} A(x)$ 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 $\mathbf{A}^\mathbf{T}$ 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. [14] and define the average conditional min-entropy of random variables X and Y as $\tilde{\mathsf{H}}_\infty(X|Y) = -\log \mathbb{E}_{y \leftarrow Y}(\max_x \mathrm{Prob}[X = x \mid Y = y])$. 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 [15]. A random variable X has average conditional HILL entropy $\tilde{\mathsf{H}}^{\mathrm{HILL}}_\infty(X|Y) \geq k$ conditioned on Y, if there is a random variable X' which is computationally indistinguishable from X, and such that $\tilde{\mathsf{H}}_\infty(X|Y) \geq k$.

B. Digital Signature Schemes

We present the basic definition of a digital signature scheme.

Definition II.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:*

- $(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathsf{KGen}(1^\lambda)$*: On input the security parameter, this randomized algorithm returns a key pair* $(\mathsf{sk}, \mathsf{pk})$*;*
- $\sigma \xleftarrow{\$} \mathsf{Sig}(\mathsf{sk}, m)$*: 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 $(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathsf{KGen}(1^\lambda)$, every $m \in \mathcal{M}$, and random $\sigma \xleftarrow{\$} \mathsf{Sig}(\mathsf{sk}, m)$, it holds that $\Pr[\mathsf{Vf}(\mathsf{pk}, m, \sigma) = 1] = 1 - \mu(\lambda)$.

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-A.

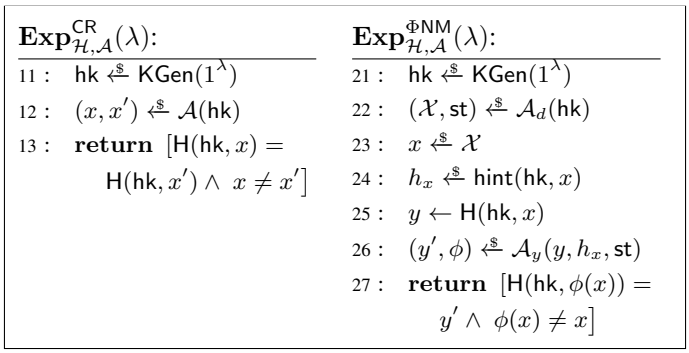

Figure 1: Definition of the security properties for a hash function. On the left: Definition of the experiment $\mathbf{Exp}^{\mathsf{CR}}_{\mathcal{H},\mathcal{A}}(\lambda)$ from Definition II.3. On the right: Definition of the experiment $\mathbf{Exp}^{\Phi\mathsf{NM}}_{\mathcal{H},\mathcal{A}}(\lambda)$ from Definition II.4.

C. 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 II.2. *A* hash function *is a pair of* PPT *algorithms* $\mathcal{H} = (\mathsf{KGen}, \mathsf{H})$ *with associated input space* $\mathcal{M}$ *such that:*

- $\mathsf{hk} \xleftarrow{\$} \mathsf{KGen}(1^\lambda)$*: On input the security parameter, this randomized algorithm generates a key* hk*;*
- $y \leftarrow \mathsf{H}(\mathsf{hk}, x)$*: 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 II.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* $\Pr[\mathbf{Exp}^{\mathsf{CR}}_{\mathcal{H},\mathcal{A}}(\lambda)] \leq \mu(\lambda)$*, where* $\mathbf{Exp}^{\mathsf{CR}}_{\mathcal{H},\mathcal{A}}(\lambda)$ *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. [16]. 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. [17] 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 II.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* $\Pr[\mathbf{Exp}^{\Phi\mathsf{NM}}_{\mathcal{H},\mathcal{A}}(\lambda)] \leq \mu(\lambda)$*, where* $\mathbf{Exp}^{\Phi\mathsf{NM}}_{\mathcal{H},\mathcal{A}}(\lambda)$ *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* $\tilde{\mathsf{H}}^{\mathit{HILL}}_{\infty}(\mathcal{X}|\mathsf{KGen}, \mathsf{hint}) \in \omega(\log \lambda)$*.*

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. [17] 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.

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

11 : $\mathcal{Q} \leftarrow \emptyset$
12 : $(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathsf{KGen}(1^\lambda)$
13 : $(m', \sigma', \mathsf{pk}') \xleftarrow{\$} \mathcal{A}^{\mathsf{Sig}(\mathsf{sk},\cdot)}(\mathsf{pk})$
14 : $d \leftarrow \mathsf{Vf}(m', \sigma', \mathsf{pk}')$
15 : **return** $[d = 1 \wedge (m', \sigma') \in \mathcal{Q} \wedge \mathsf{pk}' \neq \mathsf{pk}]$

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

11 : $\mathcal{Q} \leftarrow \emptyset$
12 : $(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathsf{KGen}(1^\lambda)$
13 : $(m', \sigma', \mathsf{pk}') \xleftarrow{\$} \mathcal{A}^{\mathsf{Sig}(\mathsf{sk},\cdot)}(\mathsf{pk})$
14 : $d \leftarrow \mathsf{Vf}(m', \sigma', \mathsf{pk}')$
15 : **return** $\left[d = 1 \wedge \left(\exists m^* \neq m' \colon (m^*, \sigma') \in \mathcal{Q}\right) \wedge \mathsf{pk}' \neq \mathsf{pk}\right]$

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

21 : $\sigma \xleftarrow{\$} \mathsf{Sig}(\mathsf{sk}, m)$
22 : $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(m, \sigma)\}$
23 : **return** σ

Figure 2: Definition of the experiments $\mathbf{Exp}^{\mathsf{CEO}}_{\Pi,\mathcal{A}}(\lambda)$ and $\mathbf{Exp}^{\mathsf{DEO}}_{\Pi,\mathcal{A}}(\lambda)$ from Definitions III.1 and III.2, respectively with access to the same signing oracle.

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.

III. Background on Security Notions beyond Unforgeability

In this section, we revisit security properties of signature schemes that go beyond unforgeability, namely exclusive ownership, non re-signability and message-bound signatures, 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 [2], [3], [5], [6], [18], [19], [20]. In [3], 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.

A. Notions of Exclusive Ownership

In the following we consider the two notions of exclusive ownership, namely conservative exclusive ownership and destructive exclusive ownership, as introduced by Pornin and Stern [2]. The underlying ideas go back to Blake-Wilson and Menezes' Duplicate-Signature Key Selection (DSKS) attacks [5] which were generalized by Menezes and Smart who termed this notion key substitution attack [6].

1) Conservative Exclusive Ownership (CEO): On an intuitive level, this notion captures that an adversary is given a legitimate public key pk as well as a signature σ on some message m. The adversary's goal is now to output a new public key pk' such that the signature σ verifies under pk', i.e. the adversary basically claims this signature to be its own. Menezes and Smart [6] denoted this as a *strong* key substitution attack.[1] We follow their exposition with updated syntax. In more formal detail, in the security game 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 III.1. *Let Π be a digital signature scheme. We say that Π provides* conservative exclusive ownership *(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 $\Pr[\mathbf{Exp}^{\mathsf{CEO}}_{\Pi,\mathcal{A}}(\lambda)] \leq \mu(\lambda)$, where $\mathbf{Exp}^{\mathsf{CEO}}_{\Pi,\mathcal{A}}(\lambda)$ is defined in Figure 2.*

2) Destructive Exclusive Ownership (DEO): This notion has been introduced by Pornin and Stern [2] and captures the following scenario. Intuitively, the adversary is given a legitimate public key pk as well as a signature σ on some message m. Here the adversary is asked to output a new public key pk' such that the signature σ verifies for a *different* message m'. Our formalization displayed in Figure 2 deviates from the one in [2] by providing the attacker with access to a signature oracle and requiring it to only output a new public key (and not the secret key). 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'.

[1] In [6], the authors have also introduced the notion of a *weak* key substitution attack where the attacker needs to also output the secret key corresponding to the public key. Note that the CEO formalization in [2] is basically a non-adaptive version of weak key substitution attacks.

$\mathbf{Exp}^{\mathsf{MBS}}_{\Pi,\mathcal{A}}(\lambda)$:

11 : $(m_1, m_2, \sigma, \mathsf{pk}) \xleftarrow{\$} \mathcal{A}()$
12 : $d_1 \leftarrow \mathsf{Vf}(\mathsf{pk}, m_1, \sigma)$
13 : $d_2 \leftarrow \mathsf{Vf}(\mathsf{pk}, m_2, \sigma)$
14 : **return** $[d_1 = 1 \wedge d_2 = 1 \wedge m_1 \neq m_2]$

$\mathbf{Exp}^{\mathsf{NR}}_{\Pi,\mathcal{A},\mathcal{D}}(\lambda)$:

21 : $(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathsf{KGen}(1^\lambda)$
22 : $(m, \mathsf{aux}) \xleftarrow{\$} \mathcal{D}(1^\lambda, \mathsf{pk})$
23 : $\sigma \xleftarrow{\$} \mathsf{Sig}(\mathsf{sk}, m)$
24 : $(\sigma', \mathsf{pk}') \xleftarrow{\$} \mathcal{A}(\mathsf{pk}, \sigma, \mathsf{aux})$
25 : $d \leftarrow \mathsf{Vf}(\mathsf{pk}', m, \sigma')$
26 : **return** $[d = 1 \wedge \mathsf{pk}' \neq \mathsf{pk}]$

Figure 3: Definition of the experiments $\mathbf{Exp}^{\mathsf{MBS}}_{\Pi,\mathcal{A}}(\lambda)$ and $\mathbf{Exp}^{\mathsf{NR}}_{\Pi,\mathcal{A}}(\lambda)$ from Definitions III.3 and IV.1, respectively.

Definition III.2. *Let Π be a digital signature scheme. We say that Π provides* destructive exclusive ownership *(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 $\Pr[\mathbf{Exp}^{\mathsf{DEO}}_{\Pi,\mathcal{A}}(\lambda)] \leq \mu(\lambda)$, where $\mathbf{Exp}^{\mathsf{DEO}}_{\Pi,\mathcal{A}}(\lambda)$ is defined in Figure 2.*

Pornin and Stern [2] point out that both notions can be combined into one notion, called universal exclusive ownership (UEO), such that UEO is equivalent to CEO and DEO. We discuss UEO in more detail in the full version of the paper.

B. 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. [21] with the name *duplicate signature* where they provide a particular example for ECDSA, and later by Jackson et al. [3] as *non-colliding signatures*.

The first game-based formalization of this notion was provided by Brendel et al. [4], who introduced the term message-bound signatures. We provide the formal details in Figure 3. 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 III.3. *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 $\Pr[\mathbf{Exp}^{\mathsf{MBS}}_{\Pi,\mathcal{A}}(\lambda)] \leq \mu(\lambda)$, where $\mathbf{Exp}^{\mathsf{MBS}}_{\Pi,\mathcal{A}}(\lambda)$ is defined on the left-hand side in Figure 3.*

Chalkias et al. [22] call MBS signatures *binding signatures* and define *strongly binding signatures* as the conjunction of the MBS and M-S-UEO notions from [4].

IV. 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 IV-A we provide the first formal security definition for non re-signability. We establish relations among the security properties in Section IV-B, before giving a generic transformation that efficiently and provably achieves our security properties beyond unforgeability in Section IV-C.

A. Non Re-signability

Jackson et al. [3] 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 of 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 on the right-hand side of Figure 3 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 from m, and the adversary is then given the legitimate public key pk, the generated signature, 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 correctly for 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 IV.1. *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 on the right-hand side in Figure 3. It is required that the* PPT *algorithm $\mathcal{D}$ outputs a pair (m, aux) such that the conditional min-entropy $\tilde{\mathsf{H}}^{HILL}_{\infty}(m|\mathsf{aux}) \in \omega(\log \lambda)$.*

B. 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 other properties except for a particular one. This holds for each property from CEO, DEO, MBS, NR and EUF-CMA and in the following we exemplify the separations only for the CEO property. The remaining relationships and respective proofs can be found in Appendix B.

Proposition IV.2. *If there is a digital signature scheme which has properties $\mathcal{P} \subseteq \{\mathsf{EUF\text{-}CMA}, \mathsf{DEO}, \mathsf{NR}, \mathsf{MBS}\}$, then there is also one which has the same properties $\mathcal{P}$ but not* CEO.

Proof. Modify the scheme $\Pi = (\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$ with properties $\mathcal{P}$ to scheme $\Pi^{\neg\mathsf{CEO}}$ 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):

$\Pi^{\neg\mathsf{CEO}}.\mathsf{KGen}(1^\lambda)$:
11 : $(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \Pi.\mathsf{KGen}(1^\lambda)$
12 : **return** $(\mathsf{sk}, \mathsf{pk}\|1)$

$\Pi^{\neg\mathsf{CEO}}.\mathsf{Sig}(\mathsf{sk}, m)$:
21 : $\sigma \xleftarrow{\$} \Pi.\mathsf{Sig}(\mathsf{sk}, m)$
22 : **if** $m = 0$ **then**
23 : **return** $\sigma\|0$
24 : **else**
25 : **return** $\sigma\|1$

$\Pi^{\neg\mathsf{CEO}}.\mathsf{Vf}(\mathsf{pk}\|b, m, \sigma\|c)$:
31 : **if** $b = 0$ **then**
32 : **return** $[c = 0 \wedge m = 0]$
33 : **else**
34 : $d \leftarrow \Pi.\mathsf{Vf}(m, \sigma, \mathsf{pk})$
35 : **return** d

The scheme inherits correctness of the original scheme.

To break property 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 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 $\Pi^{\neg\mathsf{CEO}}$ preserves the property 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 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. □

C. BUFF transformation: A generic transformation for provably achieving CEO/DEO, MBS, and NR

We construct a generic transformation that ensures that the resulting signature scheme achieves Beyond UnForgeability Features (i.e., CEO/DEO, MBS, and NR): The BUFF transformation. Before we present the details, we first revisit known transformations for some individual properties. Pornin and Stern [2] provided three transformations to add the (different) properties 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. 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 by appending the hash function evaluation of the message to the signature. 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 as $\mathsf{Sig}^*(\mathsf{sk}, m) = (\mathsf{Sig}(\mathsf{sk}, m), \mathsf{H}(m))$. 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.

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 to any signature scheme by appending the hash of the public key to the signature. The construction itself works similar to the previous one with the difference that when generating the signature one appends the hash of the public key, 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 CEO and DEO since the signature cannot be reused with any *other* public key.

However, this transformation does not 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 [6], who prepended the message with the public key in an unambiguous way to achieve a security notion that is equivalent to CEO. Without assuming the above mentioned property, the transformation achieves none of the four security properties, since a signature scheme may have a public key under which the verify algorithm unconditionally accepts.

Table II: 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	CEO	DEO	MBS	NR
[2]-1	$\mathsf{Sig}(\mathsf{sk}, m)$, $\mathsf{H}(m)$	✗	✓	✓	✗
[2]-2	$\mathsf{Sig}(\mathsf{sk}, m)$, $\mathsf{H}(\mathsf{pk})$	✓	✓	✗	✗
[2]-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 four properties (CEO, DEO, 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 5 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 4. 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

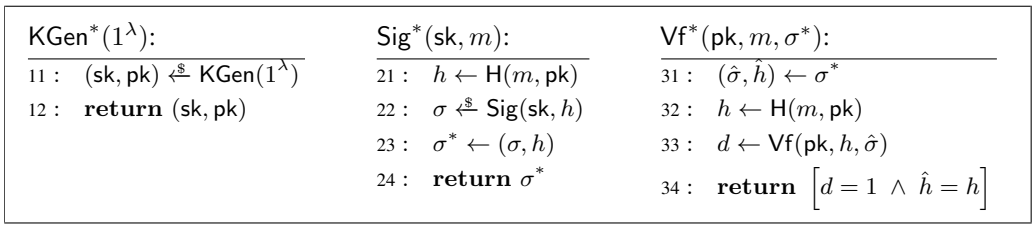

Figure 4: The BUFF (Beyond UnForgeability Features) transformation, which turns any EUF-CMA-secure signature scheme Π into an EUF-CMA-secure scheme Π^* that also achieves CEO/DEO, MBS, and NR, even if there are weak keys.

signature scheme details. However, it is known that at least for some schemes (e.g., [4]) 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 signatures. In this case the transformation does not even require a hash function invocation nor does it bear the size penalty.

Theorem IV.3. *Let* Π *be a* EUF-CMA*-secure signature scheme. Then the application of the BUFF transformation in Figure 4 produces an* EUF-CMA*-secure signature scheme* Π^* *that additionally also provides the properties of* CEO*,* DEO*,* 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 [17], 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. The proofs for those properties can be found in Appendix C. We provide the proof of non re-signability below. 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.

Proof. In this proof we show that the signature scheme Π^* obtained from transforming Π according to Figure 4 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.

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. □

V. Analyzing NIST's Round 3 Signature Schemes

In this section, we analyze the six signature schemes in round 3 of NIST's call to standardize quantum-resistant schemes [23]. Our goal is to check whether these signature schemes achieve the security properties beyond unforgeability as presented in Sections III and IV. We expand on the three finalists CRYSTALS-Dilithium [24], FALCON [25], and Rainbow [26], and summarize our analysis of the three alternate candidates GeMSS [27], Picnic [28], and SPHINCS$^+$ [29]. We give full details for the alternates in Appendix D.

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 I. We visualize the relative cost for signature size of provably achieving all three properties using our transformation in Figure 5.

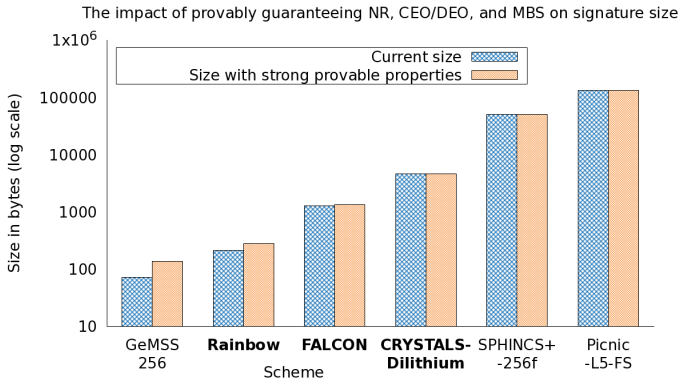

Figure 5: Provably achieving security properties beyond unforgeability for the NIST round 3 candidates: for candidates that do not provably offer these properties yet, our BUFF transformation slightly increases signature size. Since the additional size is constant (64 bytes), the largest relative increase 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 BUFF involves only a single hash, the additional computational cost is in all cases negligible compared to the signature generation and verification.

A. CRYSTALS-Dilithium

Dilithium [24] 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 [30]. Figure 6 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 $\mathbf{t} = \mathbf{A}\mathbf{s_1} + \mathbf{s_2}$ 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 $\tilde{c} \leftarrow \mathsf{H}'(\mathsf{pk}, m, \mathsf{HighBits}(\mathbf{A}\mathbf{y}))$ 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 $\mathbf{z} \leftarrow \mathbf{y} + \mathbf{c}\mathbf{s_1}$, where $\mathsf{SampleInBall}$ produces a short vector. If the resulting $\mathbf{z}$ is not short or $\mathsf{HighBits}(\mathbf{A}\mathbf{y}) \neq \mathsf{HighBits}(\mathbf{A}\mathbf{z} - \mathbf{c}\mathbf{t})$ 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 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.

$\mathsf{KGen}(1^\lambda)$

$\zeta \xleftarrow{\$} \{0,1\}^{256}$
$(\rho, \varsigma, K) \leftarrow \mathsf{H}(\zeta)$
$(\mathbf{s_1}, \mathbf{s_2}) \leftarrow \mathsf{H}(\varsigma)$ // s_1, s_2 are short vectors
$\mathbf{A} \leftarrow \mathsf{ExpandA}(\rho), \mathbf{t} \leftarrow (\mathbf{As_1} + \mathbf{s_2})$
$(\mathbf{t_0}, \mathbf{t_1}) \leftarrow (\mathsf{LowBits}(\mathbf{t}), \mathsf{HighBits}(\mathbf{t}))$
$tr \leftarrow \mathsf{CRH}(\rho, \mathbf{t_1})$
$\mathsf{sk} \leftarrow (K, tr, \mathbf{s_1}, \mathbf{s_2}, \mathbf{t_0}, \rho), \mathsf{pk} \leftarrow (\rho, \mathbf{t_1})$
return $(\mathsf{sk}, \mathsf{pk})$

$\mathsf{Vf}(\mathsf{pk}, m, \sigma)$

$\mathbf{A} \leftarrow \mathsf{ExpandA}(\rho)$
$\mu \leftarrow \mathsf{CRH}(\mathsf{CRH}(\rho, \mathbf{t_1}), m)$
$\mathbf{c} \leftarrow \mathsf{SampleInBall}(\tilde{c})$
$\mathbf{w_1}' \leftarrow \mathsf{UseHint}(h, \mathbf{Az} - \mathbf{ct})$
return $\left[\mathbf{z} \text{ short} \wedge \tilde{c} = \mathsf{H}(\mu, \mathbf{w_1}') \wedge h \text{ well-formed}\right]$

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

$\mathbf{A} \leftarrow \mathsf{ExpandA}(\rho)$
$\mu \leftarrow \mathsf{CRH}(tr, m), \rho' \leftarrow \mathsf{CRH}(K, \mu)$
$\kappa \leftarrow 0, \mathbf{z} \leftarrow \perp$
while $\mathbf{z} = \perp$
  $\mathbf{y} \xleftarrow{\$} \mathsf{ExpandMask}(\rho', \kappa)$
  $\mathbf{w_1} \leftarrow \mathsf{HighBits}(\mathbf{Ay})$
  $\tilde{c} \leftarrow \mathsf{H}(\mu, \mathbf{w_1})$
  $\mathbf{c} \leftarrow \mathsf{SampleInBall}(\tilde{c})$
  $\mathbf{z} \leftarrow \mathbf{y} + \mathbf{cs_1}$
  if $\mathbf{z}$ not short $\vee\ \mathbf{w_1} \neq \mathsf{HighBits}(\mathbf{Az} - \mathbf{ct})$ **then**
    $\mathbf{z} \leftarrow \perp$
  else
    $h \leftarrow \mathsf{MakeHint}(\mathbf{Ay}, \mathsf{sk})$
  $\kappa \leftarrow \kappa + \mathsf{dim}(\mathbf{y})$
$\sigma \leftarrow (\mathbf{z}, h, \tilde{c})$
return σ

Figure 6: Algorithmic description of Dilithium based on Figure 4 in [24].

Proposition V.1. *The signature scheme Dilithium as described in Figure 6 provides* CEO, DEO, 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 IV.3, 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.

Proof. By inspecting the details of Dilithium in Figure 6, 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 4 and therefore Theorem IV.3 applies to Dilithium. From this we can conclude that Dilithium provides CEO, DEO, 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. □

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 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.

$\mathsf{KGen}(1^\lambda)$

$(f, g, F, G) \xleftarrow{\$} \mathsf{NTRUGen}(\phi, q)$

$\mathbf{B} \leftarrow \begin{bmatrix} g & -f \\ G & -F \end{bmatrix}$

$\hat{\mathbf{B}} \leftarrow \mathsf{FFT}(\mathbf{B})$

$T \leftarrow \mathsf{FalconTree}(\hat{\mathbf{B}})$

$\mathsf{sk} \leftarrow (\hat{\mathbf{B}}, T)$

$h \leftarrow gf^{-1}$

$\mathsf{pk} \leftarrow h$

return $(\mathsf{sk}, \mathsf{pk})$

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

$r \xleftarrow{\$} \{0,1\}^{320}$

$c \leftarrow \mathsf{H}(r, m)$

$\mathbf{t} \leftarrow (\mathsf{FFT}(c), \mathsf{FFT}(0)) \cdot \hat{\mathbf{B}}^{-1}$

$\mathbf{s} \xleftarrow{\$} \mathsf{FFSampling}(\mathbf{t}, T, \lfloor \beta^2 \rfloor)$

$(s_1, s_2) \leftarrow \mathsf{FFT}^{-1}(\mathbf{s})$

$s \leftarrow \mathsf{Compress}(s_2)$

$\sigma \leftarrow (r, s)$

return σ

$\mathsf{Vf}(\mathsf{pk}, m, \sigma)$

$(r, s) \leftarrow \sigma$

$c \leftarrow \mathsf{H}(r, m)$

$s_2 \leftarrow \mathsf{Decompress}(s)$

$s_1 \leftarrow c - s_2 h$

return $\left[\|(s_1, s_2)\|^2 \leq \lfloor \beta^2 \rfloor\right]$

Figure 7: Algorithmic description of FALCON.

B. FALCON

The FALCON [25] scheme is a hash-and-sign lattice-based signature scheme based on the GPV framework [31]. The proposed scheme uses the class of NTRU lattices and a new trapdoor sampler called Fast Fourier Sampler. The security of FALCON is based on the shortest integer solution (SIS) problem.

In Figure 7 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 $h \leftarrow gf^{-1}$. Here, 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. The scheme only gives the square of β (which is also used in the verification); 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 V.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 in FALCON should indeed be hard.

Proposition V.3. *The signature scheme FALCON as described in Figure 7 provides MBS under the near-collision resistance assumption V.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. □

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

Assumption V.4 ((Non-)Invertibility Assumption for s_2)**.** *We assume that s_2 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 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 V.5. *The signature scheme FALCON as described in Figure 7 does not provide* CEO*, and under the (non-)invertibility assumption for s_2, neither* DEO*.*

Proof. An attacker against constructive exclusive ownership 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, $s_1' \leftarrow c - s_2h' = c - s_2s_2^{-1}c = c - c = 0$ 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, $s_1' \leftarrow c - s_2h' = c - s_2(h + \alpha) = c - s_2h - s_2\alpha = c - s_2h$. Thus, $s_1' = s_1$ and the bound is satisfied trivially.

An attacker against destructive exclusive ownership of FALCON can proceed in a similar fashion if s_2 is invertible, which it is with non-negligible probability according to our Assumption V.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 $s_1' \leftarrow c' - s_2h' = c' - s_2(s_2^{-1}c') = c' - c' = 0$ and therefore $\|(s_1', s_2)\|^2 \leq \|(s_1, s_2)\|^2 \leq \lfloor\beta^2\rfloor$. □

Proposition V.6. *The signature scheme FALCON as described in Figure 7 does not provide* NR *under the (non-)invertibility assumption for s_2.*

Proof. An attacker against non re-signability 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 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 V.4 the probability of s_2 not being invertible is non-negligible, such that the attacker succeeds with non-negligible probability as well. □

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.

C. Rainbow

The signature scheme Rainbow [26] 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 8.

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 $\mathbf{z}$.

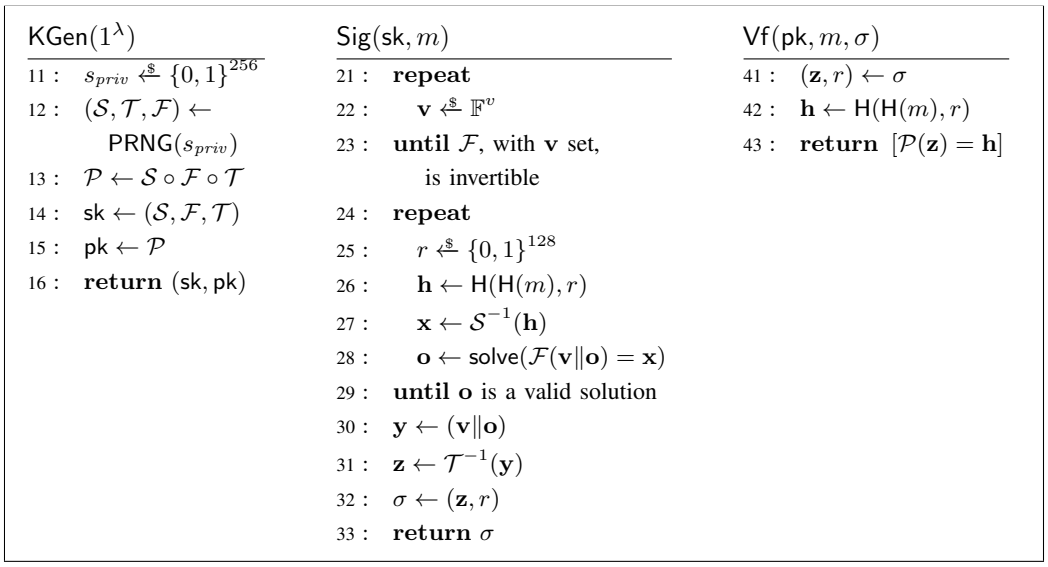

Figure 8: Algorithmic description of Rainbow.

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 ownership.

Proposition V.7. *The signature scheme Rainbow as described in Figure 8 (and its two variants) achieve* MBS *if the hash function* H *is collision resistant.*

Proof. A successful attacker against the message-bound signatures property 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 $\mathbf{h_1} \neq \mathbf{h_2}$. □

Proposition V.8. *The signature scheme Rainbow as described in Figure 8 (and its two variants) do not provide* NR.

Proof. An attacker against non re-signability 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 $(\mathsf{sk}', \mathsf{pk}')$ and then executes the signing algorithm (cf. Figure 8) 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. □

Proposition V.9. *The signature scheme Rainbow as described in Figure 8 does not provide* CEO *and, assuming collision resistance of the hash function, neither* DEO.

Proof. An attacker against 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 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 CEO and 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. We provide a more detailed discussion on this issue in the full version.

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.

D. Alternate NIST Candidates

In the following, we provide an overview of the main results of whether the alternate NIST candidates achieve any of the security properties beyond unforgeability.

1) GeMSS: The signature scheme GeMSS [27] is built from multivariate cryptography and relies on hidden field equations with vinegar specialization. We give an algorithmic description of GeMSS in Figure 10. GeMSS does not achieve any of the properties beyond unforgeability. We give the proofs of the following proposition in Appendix D-A.

Proposition V.10. *The signature scheme GeMSS as described in Figure 10 does not provide* CEO, DEO, MBS *nor* NR.

2) Picnic: The signature scheme Picnic [28], [32] is a family of digital signature algorithms based on multi-party computation, zero-knowledge proofs, and symmetric key primitives such as a hash function and a block cipher. We give an algorithmic description of Picnic in Figure 11. The scheme deploys a hash function to create the challenge in a Fiat-Shamir proof, applied to a commitment a of the multi-party computation, the public key pk, and the message m. Picnic achieves all presented properties beyond unforgeability which is summarized in the following proposition. We give the proof in Appendix D-B.

Proposition V.11. *The signature scheme Picnic as described in Figure 11 achieves* CEO, DEO, 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.

3) SPHINCS$^+$: SPHINCS$^+$ [29] is a hash-based signature scheme based on Merkle trees and employs both a one-time signature scheme (OTS) and a few-time signature scheme (FTS). We provide an algorithmic description of SPHINCS$^+$ in Figure 12. It uses two hash functions, H_{msg} for hashing the message, and H for building the hash three. Both are instantiated from one hash function like SHAKE-256. SPHINCS$^+$ can provably achieve the property of message-bound signatures which is captured in the following proposition. We give the proof details in Appendix D-C.

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

For the remaining properties of CEO, DEO and NR, we cannot provide formal proofs showing that SPHINCS$^+$ achieves them under standard assumptions. We provide some discussion arguing that we intuitively expect these notions to hold in Appendix D-C.

VI. 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: We thank Thomas Pornin and Thomas Prest for providing their insight on whether s_2 as computed by the signing algorithm in FALCON is invertible. This research work has been funded by the German Federal Ministry of Education and Research and the Hessian Ministry of Higher Education, Research, Science and the Arts within their joint support of the National Research Center for Applied Cybersecurity ATHENE. Funded also by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – SFB 1119 – 236615297 and the German Ministry of Education, Research and Technology in the context of the project Aquorypt (grant number 16KIS1022).

REFERENCES

[1] S. Goldwasser, S. Micali, and R. L. Rivest, "A digital signature scheme secure against adaptive chosen-message attacks," *SIAM Journal on Computing*, vol. 17, no. 2, pp. 281–308, Apr. 1988.
[2] T. Pornin and J. P. Stern, "Digital signatures do not guarantee exclusive ownership," in *ACNS 05*, ser. LNCS, J. Ioannidis, A. Keromytis, and M. Yung, Eds., vol. 3531. Springer, Heidelberg, Jun. 2005, pp. 138–150.
[3] D. Jackson, C. Cremers, K. Cohn-Gordon, and R. Sasse, "Seems legit: Automated analysis of subtle attacks on protocols that use signatures," in *ACM CCS 2019*, L. Cavallaro, J. Kinder, X. Wang, and J. Katz, Eds. ACM Press, Nov. 2019, pp. 2165–2180.
[4] J. Brendel, C. Cremers, D. Jackson, and M. Zhao, "The provable security of Ed25519: Theory and practice," *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 823, 2020. [Online]. Available: https://eprint.iacr.org/2020/823
[5] S. Blake-Wilson and A. Menezes, "Unknown key-share attacks on the station-to-station (STS) protocol," in *PKC'99*, ser. LNCS, H. Imai and Y. Zheng, Eds., vol. 1560. Springer, Heidelberg, Mar. 1999, pp. 154–170.
[6] A. Menezes and N. Smart, "Security of signature schemes in a multi-user setting," in *Designs, Codes and Cryptography*, vol. 33. Springer, Heidelberg, 2004, pp. 261–274.
[7] Automatic Certificate Management Environment (ACME). [Online]. Available: https://tools.ietf.org/html/draft-ietf-acme-acme-00
[8] ACME Draft Barnes. [Online]. Available: https://datatracker.ietf.org/doc/draft-barnes-acme/04/
[9] A. Ayer. ACME signature misuse vulnerability in draft-barnes-acme-04. [Online]. Available: https://www.ietf.org/mail-archive/web/acme/current/msg00484.html
[10] ——. Duplicate Signature Key Selection Attack in Let's Encrypt. [Online]. Available: https://www.agwa.name/blog/post/duplicate_signature_key_selection_attack_in_lets_encrypt
[11] T. Duong and J. Rizzo. (2009) Flickr's API Signature Forgery Vulnerability. (Retrieved November 2020). [Online]. Available: http://netifera.com/research/flickr_api_signature_forgery.pdf
[12] K. Bhargavan and G. 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.
[13] 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.
[14] Y. Dodis, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," in *EUROCRYPT 2004*, ser. LNCS, C. Cachin and J. Camenisch, Eds., vol. 3027. Springer, Heidelberg, May 2004, pp. 523–540.
[15] C.-Y. Hsiao, C.-J. Lu, and L. Reyzin, "Conditional computational entropy, or toward separating pseudoentropy from compressibility," in *EUROCRYPT 2007*, ser. LNCS, M. Naor, Ed., vol. 4515. Springer, Heidelberg, May 2007, pp. 169–186.
[16] A. Boldyreva, D. Cash, M. Fischlin, and B. Warinschi, "Foundations of non-malleable hash and one-way functions," in *ASIACRYPT 2009*, ser. LNCS, M. Matsui, Ed., vol. 5912. Springer, Heidelberg, Dec. 2009, pp. 524–541.
[17] P. Baecher, M. Fischlin, and D. Schröder, "Expedient non-malleability notions for hash functions," in *CT-RSA 2011*, ser. LNCS, A. Kiayias, Ed., vol. 6558. Springer, Heidelberg, Feb. 2011, pp. 268–283.
[18] J. Bohli, S. Röhrich, and R. Steinwandt, "Key substitution attacks revisited: Taking into account malicious signers," *Int. J. Inf. Sec.*, vol. 5, no. 1, pp. 30–36, 2006. [Online]. Available: https://doi.org/10.1007/s10207-005-0071-2
[19] C. Decker and R. Wattenhofer, "Bitcoin transaction malleability and MtGox," in *ESORICS 2014, Part II*, ser. LNCS, M. Kutylowski and J. Vaidya, Eds., vol. 8713. Springer, Heidelberg, Sep. 2014, pp. 313–326.
[20] J. Baek and K. Kim, "Remarks on the unknown key share attacks," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 83, no. 12, pp. 2766–2769, 2000.
[21] J. Stern, D. Pointcheval, J. Malone-Lee, and N. P. Smart, "Flaws in applying proof methodologies to signature schemes," in *CRYPTO 2002*, ser. LNCS, M. Yung, Ed., vol. 2442. Springer, Heidelberg, Aug. 2002, pp. 93–110.
[22] K. Chalkias, F. Garillot, and V. Nikolaenko, "Taming the Many EdDSAs," in *Security Standardisation Research - 6th International Conference, SSR 2020, London, UK, November 30 - December 1, 2020, Proceedings*, ser. Lecture Notes in Computer Science, T. van der Merwe, C. J. Mitchell, and M. Mehrnezhad, Eds., vol. 12529. Springer, 2020, pp. 67–90.
[23] National Institute of Standards and Technology (NIST), "Post-quantum cryptography," https://csrc.nist.gov/projects/post-quantum-cryptography, Aug 19, 2015.
[24] S. Bai, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé, "CRYSTALS-Dilithium: Algorithm specifications and supporting documentation," NIST Post-Quantum Cryptography Standardization Round 3 Submission, 10 2020, https://pq-crystals.org/dilithium/index.shtml.
[25] P.-A. Fouque, J. Hoffstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Prest, T. Ricosset, G. Seiler, W. Whyte, and Z. Zhang, "Falcon: Fast-fourier lattice-based compact signatures over NTRU specifications v1.2," NIST Post-Quantum Cryptography Standardization Round 3 Submission, 2020.
[26] J. Ding, M.-S. Chen, M. Kannwischer, J. Patarin, A. Petzoldt, D. Schmidt, and B.-Y. Yang, "Rainbow: Algorithm specification and documentation the 3rd round proposal," NIST Post-Quantum Cryptography Standardization Round 3 Submission, 2020.
[27] A. Casanova, J.-C. Faugère, G. Macario-Rat, J. Patarin, L. Perret, and J. Ryckeghem, "GeMSS: A Great Multivariate Short Signature," NIST Post-Quantum Cryptography Standardization Round 3 Submission, 2020.
[28] M. Chase, D. Derler, S. Goldfeder, J. Katz, V. Kolesnikov, C. Orlandi, S. Ramacher, C. Rechberger, D. Slamanig, X. Wang, and G. 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/.
[29] J.-P. Aumasson, D. J. Bernstein, W. Beullens, C. Dobraunig, M. Eichlseder, S. Fluhrer, S.-L. Gazdag, A. Hülsing, P. Kampanakis, S. Kölbl, T. Lange, M. M. Lauridsen, F. Mendel, R. Niederhagen, C. Rechberger, J. Rijneveld, P. Schwabe, and B. 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.

[30] V. Lyubashevsky, "Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures," in *ASIACRYPT 2009*, ser. LNCS, M. Matsui, Ed., vol. 5912. Springer, Heidelberg, Dec. 2009, pp. 598–616.
[31] C. Gentry, C. Peikert, and V. Vaikuntanathan, "Trapdoors for hard lattices and new cryptographic constructions," in *40th ACM STOC*, R. E. Ladner and C. Dwork, Eds. ACM Press, May 2008, pp. 197–206.
[32] G. Zaverucha, "The picnic signature scheme: Specification version 3.0," NIST Post-Quantum Cryptography Standardization Round 3 Submission, 09 2020, https://microsoft.github.io/Picnic/.
[33] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *CRYPTO'86*, ser. LNCS, A. M. Odlyzko, Ed., vol. 263. Springer, Heidelberg, Aug. 1987, pp. 186–194.
[34] D. Unruh, "Non-interactive zero-knowledge proofs in the quantum random oracle model," in *EUROCRYPT 2015, Part II*, ser. LNCS, E. Oswald and M. Fischlin, Eds., vol. 9057. Springer, Heidelberg, Apr. 2015, pp. 755–784.
[35] I. Giacomelli, J. Madsen, and C. Orlandi, "ZKBoo: Faster zero-knowledge for Boolean circuits," in *USENIX Security 2016*, T. Holz and S. Savage, Eds. USENIX Association, Aug. 2016, pp. 1069–1083.
[36] M. Chase, D. Derler, S. Goldfeder, C. Orlandi, S. Ramacher, C. Rechberger, D. Slamanig, and G. Zaverucha, "Post-quantum zero-knowledge and signatures from symmetric-key primitives," in *ACM CCS 2017*, B. M. Thuraisingham, D. Evans, T. Malkin, and D. Xu, Eds. ACM Press, Oct. / Nov. 2017, pp. 1825–1842.
[37] J. Katz, V. Kolesnikov, and X. Wang, "Improved non-interactive zero knowledge with applications to post-quantum signatures," in *ACM CCS 2018*, D. Lie, M. Mannan, M. Backes, and X. Wang, Eds. ACM Press, Oct. 2018, pp. 525–537.
[38] D. Kales and G. Zaverucha, "Improving the performance of the Picnic signature scheme," *IACR TCHES*, vol. 2020, no. 4, pp. 154–188, 2020, https://tches.iacr.org/index.php/TCHES/article/view/8680.
[39] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai, "Zero-knowledge proofs from secure multiparty computation," *SIAM J. Comput.*, vol. 39, no. 3, pp. 1121–1152, 2009. [Online]. Available: https://doi.org/10.1137/080725398
[40] M. R. Albrecht, C. Rechberger, T. Schneider, T. Tiessen, and M. Zohner, "Ciphers for MPC and FHE," in *EUROCRYPT 2015, Part I*, ser. LNCS, E. Oswald and M. Fischlin, Eds., vol. 9056. Springer, Heidelberg, Apr. 2015, pp. 430–454.
[41] J. A. Buchmann, E. Dahmen, and A. Hülsing, "XMSS - A practical forward secure signature scheme based on minimal security assumptions," in *Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011*, B.-Y. Yang, Ed. Springer, Heidelberg, Nov. / Dec. 2011, pp. 117–129.
[42] D. J. Bernstein, A. Hülsing, S. Kölbl, R. Niederhagen, J. Rijneveld, and P. Schwabe, "The SPHINCS$^+$ signature framework," in *ACM CCS 2019*, L. Cavallaro, J. Kinder, X. Wang, and J. Katz, Eds. ACM Press, Nov. 2019, pp. 2129–2146.

APPENDIX A
AUXILIARY DEFINITIONS

A. 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* $\Pr[\mathbf{Exp}^{\mathsf{EUF\text{-}CMA}}_{\Pi,\mathcal{A}}(\lambda)] \leq \mu(\lambda)$*, where* $\mathbf{Exp}^{\mathsf{EUF\text{-}CMA}}_{\Pi,\mathcal{A}}(\lambda)$ *is defined in Figure 9.*

$\mathbf{Exp}^{\mathsf{EUF\text{-}CMA}}_{\Pi,\mathcal{A}}(\lambda)$:
11 : $\mathcal{Q} \leftarrow \emptyset$
12 : $(\mathsf{sk},\mathsf{pk}) \xleftarrow{\$} \mathsf{KGen}(1^\lambda)$
13 : $(m',\sigma') \xleftarrow{\$} \mathcal{A}^{\mathsf{Sig}(\mathsf{sk},\cdot)}(\mathsf{pk})$
14 : $d \leftarrow \mathsf{Vf}(\mathsf{pk},m',\sigma')$
15 : **return** $\left[d = 1 \ \wedge \ m' \notin \mathcal{Q}\right]$

$\mathsf{Sig}(\mathsf{sk},m)$:
21 : $\sigma \xleftarrow{\$} \mathsf{Sig}(\mathsf{sk},m)$
22 : $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{m\}$
23 : **return** σ

Figure 9: Definition of the experiment $\mathbf{Exp}^{\mathsf{EUF\text{-}CMA}}_{\Pi,\mathcal{A}}(\lambda)$ from Definition A.1.

APPENDIX B
FURTHER DETAILS ABOUT THE RELATIONSHIPS

In this part of the appendix, we present the remaining relationships between the notions as started in Section IV-B.

Proposition B.1. *If there is a digital signature scheme which has any of the properties* $\mathcal{P} \subseteq \{\mathsf{EUF\text{-}CMA}, \mathsf{CEO}, \mathsf{MBS}, \mathsf{NR}\}$*, then there is also one which has the same properties* $\mathcal{P}$ *but not* DEO.

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 $\Pi^{\neg\mathsf{DEO}}$. 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):

$\Pi^{\neg\mathsf{DEO}}.\mathsf{KGen}(1^\lambda)$:
11 : $(\mathsf{sk},\mathsf{pk}) \xleftarrow{\$} \Pi.\mathsf{KGen}(1^\lambda)$
12 : $b \xleftarrow{\$} \{0,1\}$
13 : **return** $(\mathsf{sk}\|b, \mathsf{pk}\|b)$

$\Pi^{\neg\mathsf{DEO}}.\mathsf{Sig}(\mathsf{sk}\|b, m\|a)$:
21 : $\sigma \xleftarrow{\$} \Pi.\mathsf{Sig}(\mathsf{sk}, m\|\ (a \oplus b))$
22 : **return** $(\sigma, \mathsf{pk}, (a \oplus b))$

$\Pi^{\neg\mathsf{DEO}}.\mathsf{Vf}(\mathsf{pk}\|b, m\|a, (\sigma, p, c))$:
31 : $d \leftarrow \Pi.\mathsf{Vf}(\mathsf{pk}, m\|\ (a \oplus b), \sigma)$
32 : **return** $[d \wedge (c = a \oplus b) \wedge \mathsf{pk} = p]$

The scheme inherits correctness of the original scheme.

We first argue that the scheme $\Pi^{\neg\mathsf{DEO}}$ does *not* have property 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 $m' \leftarrow m\|\overline{a}$, $\sigma' \leftarrow (\sigma, \mathsf{pk}, c)$, and $\mathsf{pk}' \leftarrow \mathsf{pk}\|\overline{b}$. 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 DEO with probability 1.

We next argue that the scheme $\Pi^{\neg\mathsf{DEO}}$ has the property 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 $\mathsf{pk}' = \mathsf{pk}\|\overline{b}$. 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 $a \oplus \overline{b}$ as required for pk'. Hence, none of the pairs (m, σ) can verify under pk' and thus 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 $\Pi^{\neg\mathsf{DEO}}$. 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. □

Proposition B.2. *If there is a digital signature scheme which has any of the properties $\mathcal{P} \subseteq \{\mathsf{EUF\text{-}CMA}, \mathsf{CEO}, \mathsf{DEO}, \mathsf{NR}\}$, 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 $\Pi^{\neg\mathsf{MBS}}$ where any signature is accepted for some special public keys pk (but where the genuine key generation and signature algorithms never output such values):

$\Pi^{\neg\mathsf{MBS}}.\mathsf{KGen}(1^\lambda)$:

11 : $(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \Pi.\mathsf{KGen}(1^\lambda)$

12 : **return** $(\mathsf{sk}, \mathsf{pk}\|1)$

$\Pi^{\neg\mathsf{MBS}}.\mathsf{Sig}(\mathsf{sk}, m)$:

21 : $\sigma \xleftarrow{\$} \Pi.\mathsf{Sig}(\mathsf{sk}, m)$

22 : **return** $\sigma\|1$

$\Pi^{\neg\mathsf{MBS}}.\mathsf{Vf}(\mathsf{pk}\|b, m\|a, \sigma\|c)$:

31 : **if** $b = 0 \wedge c = 0$ **then**

32 : **return** $[\sigma = m]$

33 : **else**

34 : $d \leftarrow \Pi.\mathsf{Vf}(\mathsf{pk}, m\|a, \sigma)$

35 : **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.

We next argue that 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 CEO and 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 CEO nor DEO against the modified scheme for such values.

If, on the other hand, the CEO- or 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 $\Pi^{\neg\mathsf{MBS}}$ 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 CEO or DEO.

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 CEO and 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 $\mathsf{pk}'\|0$). □

Proposition B.3. *If there is a digital signature scheme which has any of the properties $\mathcal{P} \subseteq \{\mathsf{EUF\text{-}CMA}, \mathsf{CEO}, \mathsf{DEO}, \mathsf{MBS}\}$, 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 $(\mathsf{sk}', \mathsf{pk}')$, and signing the message under pk'.

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

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

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

$\Pi^{\neg\mathsf{EUF\text{-}CMA}}.\mathsf{KGen}(1^\lambda)$:

11 : $(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \Pi.\mathsf{KGen}(1^\lambda)$
12 : **return** $((\mathsf{sk}, \mathsf{pk}), \mathsf{pk})$

$\Pi^{\neg\mathsf{EUF\text{-}CMA}}.\mathsf{Sig}((\mathsf{sk}, \mathsf{pk}), m)$:

21 : **if** $m = \mathsf{pk}$ **then**
22 : **return** $\mathsf{pk}\|0$
23 : **else**
24 : $\sigma \xleftarrow{\$} \Pi.\mathsf{Sig}(\mathsf{sk}, m)$
25 : **return** $\sigma\|1$

$\Pi^{\neg\mathsf{EUF\text{-}CMA}}.\mathsf{Vf}(\mathsf{pk}, m, \sigma\|c)$:

31 : **if** $m = \mathsf{pk}$ **then**
32 : **return** $[\sigma = \mathsf{pk} \wedge c = 0]$
33 : **else**
34 : $d \leftarrow \Pi.\mathsf{Vf}(m, \sigma, \mathsf{pk})$
35 : **return** $[d \wedge c = 1]$

The scheme inherits correctness of the original scheme and because the signature $\mathsf{pk}\|0$ for $m = \mathsf{pk}$ 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 $m = \mathsf{pk}$ with signature $\mathsf{pk}\|0$ as a successful forgery, after having received the public key and without making any signature queries.

The modified scheme preserves CEO and 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 CEO or 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).

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. □

APPENDIX C
PROOF OF THEOREM IV.3

Proof of Theorem IV.3. We start with EUF-CMA security. 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.

Next we argue about CEO and DEO security. Let us assume a successful attacker against CEO of Π^* that outputs $(m', \sigma', \mathsf{pk}')$. Let pk denote the public key corresponding to the secret key sk used by the signing oracle and m^* the message for which the signature σ' was generated by the oracle. Since $\mathsf{Vf}^*(\mathsf{pk}, m^*, \sigma')$ and $\mathsf{Vf}^*(\mathsf{pk}', m', \sigma')$ both yield true, it must hold that $\mathsf{H}(m^*, \mathsf{pk}) = h = \mathsf{H}(m', \mathsf{pk}')$ where $\mathsf{pk} \neq \mathsf{pk}'$. 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. This argument holds analoguously for DEO.

We provide details about MBS. Let us assume a successful attacker against the message-bound signatures property 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. □

APPENDIX D
ALTERNATE NIST CANDIDATES

In the following, we provide more details of the alternate NIST candidates and their respective proofs.

A. GeMSS

The signature scheme GeMSS [27] is built from multivariate cryptography 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.

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 $\mathsf{Inv}_\mathsf{p}(\cdot, \mathsf{sk})$ 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 Inv_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 i^{th} 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.

Proof of Proposition V.10. An *attacker against constructive exclusive ownership* 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 destructive exclusive ownership* 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'.

An *attacker against message-bound signatures* 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

$\mathsf{KGen}(1^\lambda)$

11 : $(\mathcal{S}, \mathcal{T}) \xleftarrow{\$} (\mathrm{GL}_{n+v}(\mathbb{F}_2) \times \mathrm{GL}_n(\mathbb{F}_2))$
12 : $F \xleftarrow{\$} \mathbb{F}_{2^n}[X, v_1, ..., v_v]$ // with HFEv-shape
13 : $\mathsf{sk} \leftarrow (F, \mathcal{S}, \mathcal{T})$
14 : $(p_1, ..., p_n) \leftarrow \mathsf{Eval}(F, \mathcal{S}, \mathcal{T})$
15 : $\mathbf{p} \leftarrow$ first k polynomials of $(p_1, ..., p_n)$
16 : $\mathsf{pk} \leftarrow \mathbf{p}$
17 : **return** $(\mathsf{sk}, \mathsf{pk})$

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

21 : $h \leftarrow \mathsf{H}(m)$
22 : $\mathbf{s}_0 \leftarrow \mathbf{0} \in \mathbb{F}_2^k$
23 : **for** i from 1 to nb_ite **do**
24 : $\quad \mathbf{d}_i \leftarrow$ first k bits of h
25 : $\quad (\mathbf{s}_i, \mathbf{x}_i) \leftarrow \mathsf{Inv}_\mathsf{p}(\mathbf{d}_i \oplus \mathbf{s}_{i-1}, \mathsf{sk})$
26 : $\quad h \leftarrow \mathsf{H}(h)$
27 : $\sigma \leftarrow (\mathbf{s}_{nb\_ite}, \mathbf{x}_{nb\_ite}, ..., \mathbf{x}_1)$
28 : **return** σ

$\mathsf{Vf}(\mathsf{pk}, m, \sigma)$

31 : $(\mathbf{s}_{nb\_ite}, \mathbf{x}_{nb\_ite}, ..., \mathbf{x}_1) \leftarrow \sigma$
32 : $h \leftarrow \mathsf{H}(m)$
33 : **for** i from 1 to nb_ite **do**
34 : $\quad \mathbf{d}_i \leftarrow$ first k bits of h
35 : $\quad h \leftarrow \mathsf{H}(h)$
36 : **for** i from $nb\_ite - 1$ to 0 **do**
37 : $\quad \mathbf{s}_i \leftarrow \mathbf{p}(\mathbf{s}_{i+1}, \mathbf{x}_{i+1}) \oplus \mathbf{d}_{i+1}$
38 : **return** $[\mathbf{s}_0 = \mathbf{0}]$

Figure 10: Algorithmic description of GeMSS.

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 $\mathbf{p}(\mathbf{s}_1^{(2)}, \mathbf{x}_1) \oplus \mathbf{d}_1^{(2)} = \mathbf{0}$. We assume here that $\mathbf{s}_1^{(2)}$ is distinct from $\mathbf{s}_i$ for $i \in [1, nb\_ite]$ and $\mathbf{s}_i^{(2)}$ for $i \in [2, nb\_ite]$. This is reasonable to assume since 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. For each $j \in [1, k]$, there exists a polynomial $q_j \neq 0$ that satisfies $q_j(\mathbf{s}_i, \mathbf{x}_i) = 0$ for $i \in [1, nb\_ite]$ and $q_j(\mathbf{s}_i^{(2)}, \mathbf{x}_i) = 0$ for $i \in [2, nb\_ite]$ and $q_j(\mathbf{s}_1^{(2)}, \mathbf{x}_1) = p_j(\mathbf{s}_1^{(2)}, \mathbf{x}_1) \oplus d_{1,j}^{(2)}$. The full version details how to construct q_j efficiently. Then replacing $\mathbf{p}$ with $\mathbf{p}' \leftarrow \mathbf{p} + (q_1, \ldots, q_k)$ yields a new public polynomial map under which σ verifies both m_1 and m_2. Hence, the attacker succeeds by returning $(\mathbf{p}', m_1, m_2, \sigma)$.

An *attacker against non re-signability* 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. The attacker picks the polynomial $q \in \mathbb{F}_2[z_1, \ldots, z_{n+v}]$ as $\prod_{i=1}^{nb\_ite}(z_{n+v} - \mathbf{x}_{i,n+v-k}) \neq 0$, for which $q(\mathbf{s}_i, \mathbf{x}_i) = 0$. It outputs $\mathbf{p} + q\mathbf{e}_j$ for some j as new public key and the same signature. Here, $\mathbf{e}_j$ denotes the vector in $\mathbb{F}_2[z_1, \ldots, z_{n+v}]^n$ which has a 1 in the j-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.

B. Picnic

The signature scheme Picnic [28], [32] 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 [33] or Unruh transform [34]. Currently, all variants of Picnic basically follow the same design principle, however relying on different variants of the proof of knowledge protocol ZKBoo [35], 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++ [36] 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 [37], [38] 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 [39]. 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 [40], 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

$\mathsf{KGen}(1^\lambda)$

11 : $p \xleftarrow{\$} \{0,1\}^\lambda$
12 : $\mathsf{sk} \xleftarrow{\$} \{0,1\}^\lambda$
13 : $C \leftarrow \mathsf{Enc}_{LowMC}(\mathsf{sk}, p)$
14 : $\mathsf{pk} \leftarrow (C, p)$
15 : **return** $(\mathsf{sk}, \mathsf{pk})$

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

21 : $(a, salt) \xleftarrow{\$} \mathsf{simulate\_circuit}(\mathsf{sk}, \mathsf{pk}, m)$
22 : $c \leftarrow \mathsf{H}(a, \mathsf{pk}, m)$
23 : $z \leftarrow \mathsf{Prove}(c, a, salt)$
24 : $\sigma \leftarrow (c, z, salt)$
25 : **return** σ

$\mathsf{Vf}(\mathsf{pk}, m, \sigma)$

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 [28].

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 [6]. They argue that all Picnic variants provide CEO in [28, Section 7.3].

Proof of Proposition V.11. 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 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 4 and therefore Theorem IV.3 applies to Picnic. Thus, it follows directly that Picnic achieves CEO, DEO, 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 IV.3 for our BUFF transformation. □

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.

C. SPHINCS$^+$

The signature scheme SPHINCS$^+$ [29] is a hash-based signature scheme based on Merkle trees, in particular on XMSS [41]. 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 [29] and [42].

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 $\mathsf{pkFromSig}$ 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 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$^+$.

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,

$\mathsf{KGen}(1^\lambda)$

11 : $\mathsf{sk}.seed \xleftarrow{\$} \{0,1\}^{8\lambda}, \mathsf{sk}.prf \xleftarrow{\$} \{0,1\}^{8\lambda}$
12 : $\mathsf{pk}.seed \xleftarrow{\$} \{0,1\}^{8\lambda}, \mathsf{pk}.root \leftarrow$ hypertree root
13 : $\mathsf{sk} \leftarrow (\mathsf{sk}.seed, \mathsf{sk}.prf)$
14 : $\mathsf{pk} \leftarrow (\mathsf{pk}.root, \mathsf{pk}.seed)$
15 : **return** $(\mathsf{sk}, \mathsf{pk})$

$\mathsf{Vf}(\mathsf{pk}, m, \sigma)$

31 : $(r, \sigma_{FORS}, \sigma_{HT}) \leftarrow \sigma$
32 : $(md \| idx) \leftarrow \mathsf{H}_{msg}(r, \mathsf{pk}, m)$
33 : $\mathsf{pk}_{FORS} \leftarrow \mathsf{pkFromSig}_{FORS}(\sigma_{FORS}, md, \mathsf{pk}.seed, idx)$
34 : **return** $\mathsf{Vf}_{HT}(\mathsf{pk}_{FORS}, \sigma_{HT}, idx, \mathsf{pk}.root)$

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

21 : $r \leftarrow \mathrm{PRF}_{msg}(\mathsf{sk}.prf, OptRand, m)$
22 : $(md \| idx) \leftarrow \mathsf{H}_{msg}(r, \mathsf{pk}, m)$
23 : $\sigma_{FORS} \leftarrow \mathsf{Sig}_{FORS}(md, \mathsf{sk}.seed, \mathsf{pk}.seed, idx)$
24 : $\mathsf{pk}_{FORS} \leftarrow \mathsf{pkFromSig}_{FORS}(\sigma_{FORS}, md, \mathsf{pk}.seed, idx)$
25 : $\sigma_{HT} \leftarrow \mathsf{Sig}_{HT}(\mathsf{pk}_{FORS}, \mathsf{sk}.seed, \mathsf{pk}.seed, idx)$
26 : $\sigma \leftarrow (r, \sigma_{FORS}, \sigma_{HT})$
27 : **return** σ

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

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$^+$ [42] 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.

Proof of Proposition V.12. 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 $(md_1 \| idx_1) \leftarrow \mathsf{H}_{msg}(r_1, \mathsf{pk}, m_1)$ and $(md_2 \| idx_2) \leftarrow \mathsf{H}_{msg}(r_2, \mathsf{pk}, m_2)$, 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, $node \leftarrow \mathsf{H}(\mathsf{pk}.seed, \mathrm{ADRS}, \mathrm{leftChild}, \mathrm{rightChild})$ 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. □

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

Let us start with 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. $\mathsf{pk}'.seed \neq \mathsf{pk}.seed$ or $\mathsf{pk}'.root \neq \mathsf{pk}.root$. 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 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 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 $\mathsf{H}_{msg}(r, \mathsf{pk}', m)$ must be related to the original hash value $\mathsf{H}_{msg}(r, \mathsf{pk}, m)$ 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.