A Framework for Advanced Signature Notions

Patrick Struck[1] and Maximiliane Weishäupl[2]

[1] Universität Konstanz, Germany
patrick.struck@uni-konstanz.de
[2] Universität Regensburg, Germany
maximiliane.weishaeupl@ur.de

Abstract. The beyond unforgeability features formalize additional security properties for signature schemes. We develop a general framework of binding properties for signature schemes that encompasses existing beyond unforgeability features and reveals new notions. Furthermore, we give new results regarding various transforms: We show that the transform by Cremers et al. (SP'21) achieves all of our security notions and provide requirements such that this is also the case for the transform by Pornin and Stern (ACNS'05). Finally, we connect our framework to unforgeability notions.

1 Introduction

Signature schemes, key-encapsulation mechanisms, and authenticated encryption are among the most basic cryptographic primitives, achieving fundamental security goals such as confidentiality, integrity, and authenticity. These security goals are attained by showing constructions to achieve standard security notions, such as *existential unforgeability under chosen message attack* (EUF-CMA) in case of signature schemes. In practice, though, the aforementioned primitives are seldom used as standalone primitives. They are commonly embedded into larger, more complex protocols. The way a protocol makes use of a primitive can open new attack vectors that are beyond what standard security, like, say, EUF-CMA security, guarantees. The recent literature provides numerous examples for such attacks with severe effects: the "Facebook message franking attack" [DGRW18], the "partitioning oracle attack" [LGR21], the "subscribe with Google attack" [ADG+22], the "Let's Encrypt attack" [Aye15], the "Dynamically Recreatable Key attack" [JCCS19], and the "PQXDH re-encapsulation attack" [BJKS24]. The list certainly does not end here as more attacks, against other existing or future protocols are sure to be found. All of these attacks were possible *even* for cryptographic primitives that were proven secure in the standard sense, e.g., EUF-CMA security.

To deal with the problem, new security notions for authenticated encryption, key-encapsulation mechanisms, and signature schemes were developed, with the

* This work was funded by the DFG – SFB 1119 – 236615297, the BMFTR under the projects QUDIS (16KIS2091) and Quant-ID (16KISQ111), as well as the Hector Foundation II.

goal of preventing attacks like in the aforementioned list. For authenticated encryption and key-encapsulation, so-called committing and binding security, respectively, prevent attacks like the ones described above. For signature schemes, the so-called *beyond unforgeability features* or *BUFF security notions* were introduced for this purpose. As of now, there are six BUFF security notions: S-CEO, S-DEO, S-UEO, M-S-UEO, MBS, and NR. The first four are variants of *exclusive ownership* notions, which, roughly speaking, ask if one can find a public key that verifies a signature from a different public key—the different variants cover various subtleties of this core idea. The notion MBS is related to the non-repudiation property of signatures: it asks to find a public key and *one* signature that verifies *two distinct* messages. The notion NR asks an adversary to morph a signature of an *unknown* message for one public key, into a signature for the same message but a different public key.

As mentioned earlier, there are no guarantees that the existing list of attacks is complete—in fact, it seems even likely that more attacks will be found in the future. Ideally, we would like to design and analyze cryptographic primitives today, such that they withstand not just the known attacks but also the currently unknown attacks. To achieve that, we need security notions that cover all different attack vectors that might be exploited in such (potentially still unknown) attacks.[3] For authenticated encryption and key-encapsulation mechanisms, we do have these security notions: Menda et al. [MLGR23] developed a complete framework of security notions for committing security while Cremers et al. [CDM24] did the same for key-encapsulation mechanisms. For signature schemes, on the other hand, there is no formal framework covering the various subtleties that might go wrong. While the BUFF security notions already catch a number of these cases, the following question remains open:

Are the BUFF security notions complete?

1.1 Contribution

In this work, we address the aforementioned question. We apply the blueprint of binding properties (as done in [CDM24] for key-encapsulation mechanisms) to signature schemes. More precisely, the generic structure of each notion is Am-B-T, which can be read as "B binds T in attack model Am". Here, B and T are subsets of $\{\mathrm{S}, \mathrm{M}, \mathrm{P}\}$, which represent the signature, message, and public key, respectively, while $\mathrm{Am} \in \{\textsc{Mal}, \textsc{Leak}, \textsc{Hon}\}$ represents one of three attack models. By this, we obtain a total of 36 security notions. Removing notions that are not meaningful in the context of signatures, leaves us with 15 security notions—5 notion classes (on which we elaborate below) each of which can be considered in the 3 different attack models (Mal, Leak, and Hon) which differ in how the keys are generated. Our 5 notion classes can be distinguished as follows. There are 2 *generalized notions*: Am-S-M (does a signature bind

[3] This means of course only security in a black-box sense, where the adversary has no access to the internal workings—unlike, say, side-channel attacks.

a message?) and AM-S-P (does a signature bind a public key?). There are 3 *restricted notions*: AM-[S,M]-P (do signature and message *together* bind a public key?), AM-S-[M,P] (does a signature bind *the pair* of message and public key?), and AM-[S,P]-M (do signature and public key *together* bind a message?). These notions cover the existing BUFF security notions S-CEO, S-DEO, S-UEO, M-S-UEO, and MBS.[4] The notions S-CEO, S-DEO, and MBS correspond to the restricted notion AM-[S,M]-P, AM-S-[M,P], and AM-[S,P]-M, respectively; an interesting observation here, is that MBS is in a different attack model (MAL) than S-CEO and S-DEO (which are HON). The notions S-UEO and M-S-UEO belong to the notion class AM-S-P, differing only in their attack model (HON and MAL).

Next to encompassing the existing BUFF notions, our approach aids the identification of gaps: Out of our 15 notions, the prior BUFF notions S-CEO, S-DEO, UEO, M-S-UEO, and MBS account for 5, while the remaining 10 have not been considered before. Our new framework completes the picture in two ways: Firstly, it introduces new variants of the existing BUFF notions in different attack modes (e.g., MBS in the HON and LEAK model), and secondly, it reveals the new notion class AM-S-M, that has not been considered before.

Alongside the introduction of the notions, we analyze their relations by showing implications as well as separations. The known connection between S-CEO, S-DEO, and S-UEO in the HON model, transfers also to the other attack modes, i.e., more generally we prove that AM-S-P decomposes into AM-[S,M]-P and AM-S-[M,P]. Moreover, the new notions class AM-S-M exhibits such a relation as well: namely AM-S-M holds if and only if AM-[S,P]-M and AM-S-[M,P] hold. This reveals a connection between "MBS-like" and "S-DEO-like" notions that is symmetric to the one between S-CEO and S-DEO. In particular, AM-S-[M,P] occupies a special position in this context as it is part of both relations. An illustration of this is provided in Fig. 1. This results in a hierarchy between the notions, where AM-S-P and AM-S-M imply all other notion classes. Taking into account the natural hierarchy MAL $\rightarrow$ LEAK $\rightarrow$ HON (in decreasing order of strength), we observe that the two notions MAL-S-P and MAL-S-M yield all others. When compared to the hierarchy in other frameworks for advanced security notions, we observe the following: For authenticated encryption, committing security notions form a hierarchy [MLGR23], where it suffices to target the strongest form as done, for instance, in [KSW24b, NSS23], in the context of the NIST lightweight cryptography finalists [NIST15]. For key-encapsulation mechanisms, the notions by Cremers et al. [CDM24] do not form such a hierarchy with one notion implying all others. Instead, the picture is similar to what we observe in this work for signatures, that is, a few notions together imply all others. Thus, to be certain, one needs to show security with respect to all of these notions, e.g., as done in [KSW24a], which developed a variant of the Fujisaki-Okamoto transform that achieves the required notions.

[4] Note that NR is not included in the list. This is to be expected as it is a very different notion with the concept of a message that is unknown to the adversary.

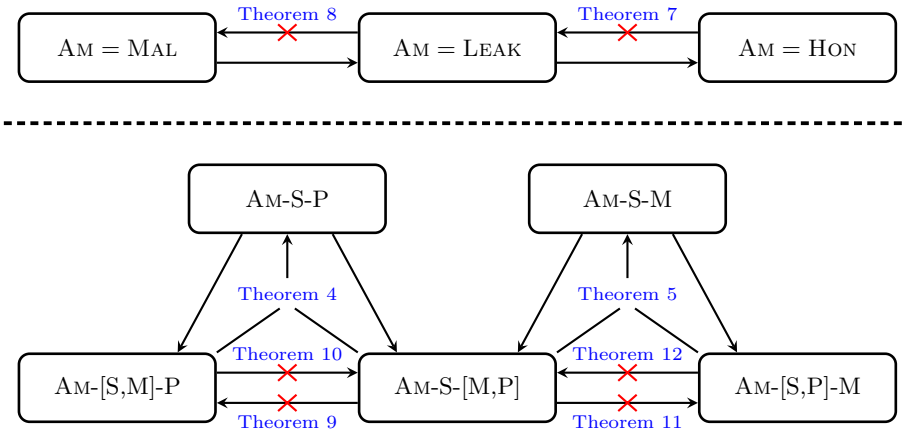

Fig. 1: Implications (arrows) and separations (crossed arrows) between the different attack models (upper part) and the different notion classes (lower part) with the theorems that establish the results (or none for trivial implications). The two notions AM-S-P and AM-S-M are the generalized notions while the remaining three notions AM-[S,M]-P, AM-S-[M,P], and AM-[S,P]-M, are the restricted notions.

After having established the framework with its various relations, we turn towards achievability of the notions: For this we analyze the existing transforms for obtaining BUFF security in the context of our new framework. We consider the PS-1, PS-2, and PS-3 transforms from [PS05], as well as the BUFF-lite and BUFF transforms from [CDF+21]. The PS-3 transform has a unique advantage as it is the only one among the transforms that does not impose an increase in the signature size. However, its security is tied to the absence of so-called weak keys, which have appeared in previous works [PS05,CDF+21,DS24] with varying characterizations. As of now, it is known that the PS-3 transform fulfills HON-S-P (i.e., S-UEO) security under the assumption that no weak keys exist. We address these gaps as follows: We provide a new formalization of weak keys and prove that a signature schemes for which these weak keys can be excluded achieves *all* notions from our framework. Regarding the remaining transforms, we show that the PS-1 and PS-2 transform yield a total of nine notations each—note that they overlap in three notions and together cover the complete framework. Lastly, we prove that the BUFF-lite (and thus in particular the BUFF) transform attain all notions from our framework. This reveals that we do not require a new transformation to achieve our notions as the BUFF(-lite) transform already fulfills the task.

Finally, we demonstrate that a slight modification of our notions allows to model the unforgeability notions—both existential and strong unforgeability.

1.2 Related Work

The beyond unforgeability features *message-bound signatures*, *exclusive ownership* (in its different forms), and *non-resignability*, were formalized in [CDF+21]. Message-bound signatures originate in [SPMS02], where the property was introduced using the term "duplicate signatures". The exclusive ownership notions were introduced in [PS05] but the idea can be traced back further [BWM99, MS04]. Non-resignability was first formalized in [CDF+21], based on an attack from [JCCS19], but identified as flawed in [DFHS24]. The initial claims regarding the non-resignability of the BUFF transform were restored in [DFH+24].

Alongside the introduction of the beyond unforgeability features, Cremers et al. [CDF+21] analyzed the round-3 candidates in the NIST PQC standardization process [NIST17]: Dilithium, Falcon, Rainbow, GeMSS, Sphincs+, and Picnic. Later, Düzlü et al. [DFF24] and Düzlü and Struck [DS24] provided new results for FALCON and SPHINCS*, respectively. Aulbach et al. [ADM+24] analyzed the signature schemes based on lattices, codes, isogenies, and multivariate equations, in the additional call for PQC-signatures by NIST [NIST22]. Kulkarni and Xagawa [KX24] did a similar analysis for the MPC-in-the-Head signatures. Emura [Emu24] analyzed ECDSA while Fischlin et al. [FMT25] considered BUFF security for threshold signature schemes.

2 Preliminaries

In this section, we give the relevant background for this work. This entails the definition of signature schemes, its standard security notions—existential and strong unforgeability—and the BUFF notions[5]. By adversary, we mean an efficient, i.e., probabilistic polynomial-time algorithm with respect to a security parameter λ. We typically consider the security parameter to be implicit and do not state it explicitly. Furthermore, we write H to denote a random oracle that is used in the transformations later.

Definition 1. *A signature scheme* $\mathtt{S}$ *consists of three efficient algorithms:*

$\mathtt{KeyGen}(1^\lambda) \to (\mathtt{pk}, \mathtt{sk})$**:** *The key generation algorithm takes as input a security parameter and outputs a key pair* $(\mathtt{pk}, \mathtt{sk})$.

$\mathtt{Sign}(\mathtt{sk}, \mathtt{msg}) \to \mathtt{sig}$**:** *The signing algorithm takes as input a secret key* $\mathtt{sk}$ *and a message* $\mathtt{msg}$*, and outputs a signature* $\mathtt{sig}$.

$\mathtt{Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig}) \to \mathtt{v}$**:** *The verification algorithm takes as input a public key* $\mathtt{pk}$*, a message* $\mathtt{msg}$*, and a signature* $\mathtt{sig}$*, and it outputs a bit* $\mathtt{v}$.

A signature scheme is said to be correct, if, for any $(\mathtt{pk}, \mathtt{sk}) \leftarrow\!\!\$\ \mathtt{KeyGen}(1^\lambda)$ *and any message* $\mathtt{msg}$*,* $\mathtt{Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{Sign}(\mathtt{sk}, \mathtt{msg})) = 1$ *holds with overwhelming probability.*

Definition 2. *A signature scheme* $\mathtt{S} = (\mathtt{KeyGen}, \mathtt{Sign}, \mathtt{Verify})$ *fulfills* EUF-CMA *and* SUF-CMA *if for any efficient adversary* $\mathcal{A}$*, its probability in winning the corresponding game shown in Fig. 2 is negligible.*

[5] Except for non-resignability, which is not considered in this work.

Game EUF-CMA	Game SUF-CMA	Oracle $\mathsf{Sign}(\mathtt{sk}, \mathtt{msg})$
$(\mathtt{pk}, \mathtt{sk}) \leftarrow\!\!\$\ \mathtt{KeyGen}()$	$(\mathtt{pk}, \mathtt{sk}) \leftarrow\!\!\$\ \mathtt{KeyGen}()$	$\mathtt{sig} \leftarrow \mathtt{Sign}(\mathtt{sk}, \mathtt{msg})$
$\mathcal{Q} \leftarrow \emptyset$	$\mathcal{Q} \leftarrow \emptyset$	$\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(\mathtt{msg}, \mathtt{sig})\}$
$(\mathtt{msg}, \mathtt{sig}) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathtt{sk}, \cdot)}(\mathtt{pk})$	$(\mathtt{msg}, \mathtt{sig}) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathtt{sk}, \cdot)}(\mathtt{pk})$	**return** $\mathtt{sig}$
if $(\mathtt{msg}, \cdot) \in \mathcal{Q}$	**if** $(\mathtt{msg}, \mathtt{sig}) \in \mathcal{Q}$	
return 0	**return** 0	
return $\mathtt{Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig})$	**return** $\mathtt{Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig})$	

Fig. 2: The notions EUF-CMA and SUF-CMA.

Definition 3. *A signature scheme* $\mathtt{S} = (\mathtt{KeyGen}, \mathtt{Sign}, \mathtt{Verify})$ *fulfills* S-CEO, S-DEO, S-UEO, M-S-UEO, *and* MBS *if for any efficient adversary* $\mathcal{A}$*, its probability in winning the corresponding game shown in Fig. 3 is negligible.*

3 Framework

In this section, we give a framework for binding properties of signature schemes. We first introduce the various notions and subsequently show implications and separations between them.

3.1 Notions

We deploy the systematic approach from [CDM24], i.e., we consider Am-B-T as a generic structure for each notion. The generic notions are displayed in Fig. 4.

The components B and T describe which component(s)—represented by B—bind which component(s)—represented by T. Since we are concerned with signature schemes, the relevant elements are the signature (S), the message (M), and the public key (P), i.e., $\mathrm{B}, \mathrm{T} \subseteq \{\mathrm{S}, \mathrm{M}, \mathrm{P}\}$. For instance, Am-S-P, formalizes that the signature binds the public key. Speaking differently, it should be infeasible to find a signature (the binding component B) that verfies under two different public keys (the target component T). If B or T contain more than one element, we use square brackets for clarification, e.g., Am-[S,M]-P and Am-S-[M,P].

The component Am describes the attack model and can take any value from $\{\textsc{Hon}, \textsc{Leak}, \textsc{Mal}\}$. The attack models describes the kind of access the adversary has to the key pairs. For $\textsc{Am} = \textsc{Mal}$ (malicious setting), the adversary can generate both key pairs (or the one key pair for notions which only allow for one public key, i.e., notions with $\mathrm{P} \in \mathrm{B}$) itself. For $\textsc{Am} = \textsc{Hon}$ (honest setting), the adversary receives an honestly generated public key and can request signatures via the signing oracle; breaking the binding property then needs to involve a signature received from the signing oracle, i.e., an honestly generated signature. For $\textsc{Am} = \textsc{Leak}$ (leakage setting), the key pair is honestly generated, as in the honest setting, but the adversary receives both public key *and* secret key—in

Game S-CEO

$\mathcal{Q} \leftarrow \emptyset$
$(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathtt{KeyGen}()$
$(\mathtt{sig}, \mathtt{msg}, \overline{\mathtt{pk}}) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathtt{sk},\cdot)}(\mathtt{pk})$
if $(\mathtt{msg}, \mathtt{sig}) \notin \mathcal{Q}$
  return 0
$\mathtt{v} \leftarrow \mathtt{Verify}(\overline{\mathtt{pk}}, \mathtt{msg}, \mathtt{sig})$
return $[\![\mathtt{v} = 1 \wedge \overline{\mathtt{pk}} \neq \mathtt{pk}]\!]$

Game S-DEO

$\mathcal{Q} \leftarrow \emptyset$
$(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathtt{KeyGen}()$
$(\mathtt{sig}, \mathtt{msg}, \overline{\mathtt{pk}}) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathtt{sk},\cdot)}(\mathtt{pk})$
if $\exists \overline{\mathtt{msg}} \neq \mathtt{msg}$ **s.t.** $(\mathtt{msg}, \mathtt{sig}) \in \mathcal{Q}$
  $\mathtt{v}_0 \leftarrow 1$
$\mathtt{v}_1 \leftarrow \mathtt{Verify}(\overline{\mathtt{pk}}, \mathtt{msg}, \mathtt{sig})$
return $[\![\mathtt{v}_0 = 1 \wedge \mathtt{v}_1 = 1 \wedge \overline{\mathtt{pk}} \neq \mathtt{pk}]\!]$

Game S-UEO

$\mathcal{Q} \leftarrow \emptyset$
$(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathtt{KeyGen}()$
$(\mathtt{sig}, \mathtt{msg}, \overline{\mathtt{pk}}) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathtt{sk},\cdot)}(\mathtt{pk})$
if $(\cdot, \mathtt{sig}) \notin \mathcal{Q}$
  return 0
$\mathtt{v} \leftarrow \mathtt{S.Verify}(\overline{\mathtt{pk}}, \mathtt{msg}, \mathtt{sig})$
return $[\![\mathtt{v} = 1 \wedge \overline{\mathtt{pk}} \neq \mathtt{pk}]\!]$

Oracle $\mathsf{Sign}(\mathtt{sk}, \mathtt{msg})$

$\mathtt{sig} \leftarrow \mathtt{Sign}(\mathtt{sk}, \mathtt{msg})$
$\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(\mathtt{msg}, \mathtt{sig})\}$
return sig

Game M-S-UEO

$(\mathtt{sig}, \mathtt{msg}, \overline{\mathtt{msg}}, \mathtt{pk}, \overline{\mathtt{pk}}) \leftarrow \mathcal{A}()$
$\mathtt{v}_0 \leftarrow \mathtt{S.Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig})$
$\mathtt{v}_1 \leftarrow \mathtt{S.Verify}(\overline{\mathtt{pk}}, \overline{\mathtt{msg}}, \mathtt{sig})$
return $[\![\mathtt{v}_0 = 1 \wedge \mathtt{v}_1 = 1 \wedge \overline{\mathtt{pk}} \neq \mathtt{pk}]\!]$

Game MBS

$(\mathtt{sig}, \mathtt{msg}, \overline{\mathtt{msg}}, \mathtt{pk}) \leftarrow \mathcal{A}()$
$\mathtt{v}_0 \leftarrow \mathtt{Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig})$
$\mathtt{v}_1 \leftarrow \mathtt{Verify}(\mathtt{pk}, \overline{\mathtt{msg}}, \mathtt{sig})$
return $[\![\mathtt{msg} \neq \overline{\mathtt{msg}} \wedge \mathtt{v}_0 = 1 \wedge \mathtt{v}_1 = 1]\!]$

Fig. 3: Security games S-CEO, S-DEO, S-UEO, M-S-UEO, and MBS.

particular the signing oracle is omitted as it becomes obsolete. We want to emphasize, that in our notions the attack models differ from [CDM24] in how the target key pair is generated; the second key pair—if a notion asks for one—is *always* chosen by the adversary, allowing it to be maliciously generated.[6] This is in contrast to the binding notions for key-encapsulation mechanisms [CDM24], where the attack model is a directive for both keys. This difference stems from the fact that for key-encapsulation mechanisms, the adversary tries to disrupt the communication between two parties, whereas for signatures, the adversary wants to claim a signature for itself.

Depending on the attack model, the notion Am-S-P corresponds to two existing notions: For Am = Mal, it equals the notion M-S-UEO, while for Am = Hon, it equals the notion S-UEO. On the other hand, for, Am = Leak, the notion does not correspond to any existing notion from prior literature. Sim-

[6] However, note that the attacker only ever has to output the public key—never the secret one. In contrast, the initial notions from [PS05] required the adversary to also output the secret key.

Table 1: Overview of all possible binding notions. The rows correspond to different notion classes with the respective column describing the specific attack model. Cells containing ✗ are notions that are not achievable. For the remaining cells, a ⋆ denotes that the corresponding notion is new, while otherwise the existing name of the notion displayed.

Am-B-T	Mal	Leak	Hon
Am-S-M	⋆	⋆	⋆
Am-S-P	M-S-UEO	⋆	UEO
Am-M-S	✗	✗	✗
Am-M-P	✗	✗	✗
Am-P-S	✗	✗	✗
Am-P-M	✗	✗	✗
Am-[S,P]-M	MBS	⋆	⋆
Am-S-[M,P]	⋆	⋆	DEO
Am-[S,M]-P	⋆	⋆	CEO
Am-M-[S,P]	✗	✗	✗
Am-[M,P]-S	✗	✗	✗
Am-P-[S,M]	✗	✗	✗

ilarly, our framework captures the remaining BUFF security notions.[7] That is, S-CEO and S-DEO correspond to Hon-[S,M]-P and Hon-S-[M,P], respectively, while MBS equals Mal-[S,P]-M. When looking at prior literature, one can get the impression that MBS is "on the same level" as S-CEO, S-DEO, and S-UEO, while M-S-UEO corresponds to a stronger attack model. In light of our framework, however, MBS is actually closer to M-S-UEO (as both are in the malicious model) while the remaining notions are all in the honest model.

In total, one can consider a variety of 36 security notions: 12 combinations for B and T (6 containing all S, M, and P, and 6 containing only two out of those three) times the 3 attack models.

3.2 Unachievable Notions

Several of the notions turn out to be unachievable which we discuss here. We describe simple attacks for Am = Hon, which restrict the adversary to using the provided signing oracle. The attacks easily extend to Am ∈ {Leak, Mal}: for Leak, the adversary can emulate the oracle with the secret key it receives as input, whereas for Mal, the adversary can simply generate the key pair(s) honestly and then proceed like in the Leak setting.

Messages are not Binding. The message itself cannot bind anything, making the notions Am-M-P, Am-M-S, and Am-M-[S,P] unachievable. The following attack

[7] Note that our framework does not encompass the BUFF notion NR, which differs significantly from the others due to the concept of an unknown message.

Game MAL-B-T

$(\mathtt{sig}, \overline{\mathtt{sig}}, \mathtt{msg}, \overline{\mathtt{msg}}, \mathtt{pk}, \overline{\mathtt{pk}}) \leftarrow \mathcal{A}()$
if $\mathtt{Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig}) = \bot$
  return 0
if $\mathtt{Verify}(\overline{\mathtt{pk}}, \overline{\mathtt{msg}}, \overline{\mathtt{sig}}) = \bot$
  return 0
return $[\![(\forall x \in \mathrm{B}.x = \overline{x}) \wedge (\exists y \in \mathrm{T}.y \neq \overline{y})]\!]$

Game LEAK-B-T

$(\mathtt{pk}, \mathtt{sk}) \leftarrow\!\!\$\ \mathtt{KeyGen}()$
$(\mathtt{sig}, \overline{\mathtt{sig}}, \mathtt{msg}, \overline{\mathtt{msg}}, \overline{\mathtt{pk}}) \leftarrow \mathcal{A}(\mathtt{pk}, \mathtt{sk})$
if $\mathtt{Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig}) = \bot$
  return 0
if $\mathtt{Verify}(\overline{\mathtt{pk}}, \overline{\mathtt{msg}}, \overline{\mathtt{sig}}) = \bot$
  return 0
return $[\![(\forall x \in \mathrm{B}.x = \overline{x}) \wedge (\exists y \in \mathrm{T}.y \neq \overline{y})]\!]$

Game HON-B-T

$(\mathtt{pk}, \mathtt{sk}) \leftarrow\!\!\$\ \mathtt{KeyGen}()$
$(\mathtt{sig}, \overline{\mathtt{sig}}, \mathtt{msg}, \overline{\mathtt{msg}}, \overline{\mathtt{pk}}) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathtt{sk}, \cdot)}(\mathtt{pk})$
if $(\mathtt{msg}, \mathtt{sig}) \notin \mathcal{Q}$
  return 0
if $\mathtt{Verify}(\overline{\mathtt{pk}}, \overline{\mathtt{msg}}, \overline{\mathtt{sig}}) = \bot$
  return 0
return $[\![(\forall x \in \mathrm{B}.x = \overline{x}) \wedge (\exists y \in \mathrm{T}.y \neq \overline{y})]\!]$

Oracle $\mathsf{Sign}(\mathtt{sk}, \mathtt{msg})$

$\mathtt{sig} \leftarrow \mathtt{S.Sign}(\mathtt{sk}, \mathtt{msg})$
$\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(\mathtt{msg}, \mathtt{sig})\}$
return sig

Fig. 4: Generic game for the three attack modes MAL, LEAK, and HON.

works against all three notions: the adversary is given a public key $\mathtt{pk}$ and a matching sign oracle. Then, it generates a second key pair $(\overline{\mathtt{pk}}, \overline{\mathtt{sk}})$ using $\mathtt{KeyGen}$ and picks an arbitrary message $\mathtt{msg}$. The adversary, queries $\mathtt{msg}$ to the sign oracle resulting in a signature $\mathtt{sig}$, and computes $\overline{\mathtt{sig}}$ as the signature of $\mathtt{msg}$ under $\overline{\mathtt{sk}}$. These signatures will most likely differ and by correctness, they verify under the respective public keys.

Public Keys are not Binding. Similarly to the message, also the public key itself cannot bind anything. This makes the following notions unachievable: AM-P-M, AM-P-S, and AM-P-[S,M]. An adversary receiving a public key $\mathtt{pk}$, can simply query two distinct messages $\mathtt{msg}$ and $\overline{\mathtt{msg}}$ to the signing oracle receiving two (likely to be different) signatures $\mathtt{sig}$ and $\overline{\mathtt{sig}}$.

Messages and Public Keys together are not Binding. Finally, messages and public keys together also do not bind the signature. This shows that the notion AM-[M,P]-S is unachievable. An adversary receiving a public key can query an arbitrary message $\mathtt{msg}$ twice to the signing oracle to receive two different signatures $\mathtt{sig}$ and $\overline{\mathtt{sig}}$.

The above results show that, in the context of signature schemes, only the signature can bind other values. Thus, we can restrict our focus to notions where

S ∈ B. Ignoring the attack model for now, this leaves us with 5 notion classes: AM-S-[M,P], AM-[S,P]-M, AM-[S,M]-P, AM-S-M, and AM-S-P. Considering the three attack models, we end with 15 binding notions (which cover the existing BUFF security notions MBS, S-CEO, S-DEO, S-UEO, and M-S-UEO as discussed above). An overview is given in Table 1. This table shows which notions are not achievable, which correspond to existing notions from the literature, and which notions are new. Implications and separations between the different notion classes and attack models are illustrated in Fig. 1. The generic notions MAL-B-T, LEAK-B-T, and HON-B-T are depicted in Fig. 4, while all 15 individual notions are given in Appendix B.

3.3 Implications

In this section we discuss the implications between the different notion classes AM-S-P, AM-S-M, AM-[S,M]-P, AM-S-[M,P], and AM-[S,P]-M. In the following, we will refer to the former two as the *generalized notions*, in the sense that they are less restricted by only involving two out of the three components (signature, message, and public key), and the latter three as the *restricted notions*.

There are some trivial implications from generalized notions to restricted notions. For instance, if the signature binds the public key, then signature and message *together* bind the public key. This yields the implication AM-S-P ⇒ AM-[S,M]-P. At the same time, if the signature bind any of the other two values (message or signature), it also bind the pair of both, e.g., AM-S-P ⇒ AM-S-[M,P]. Overall, we obtain the following trivial implications:

$$\text{AM-S-P} \Rightarrow \text{AM-[S,M]-P} \qquad \text{AM-S-M} \Rightarrow \text{AM-[S,P]-M}$$
$$\text{AM-S-P} \Rightarrow \text{AM-S-[M,P]} \qquad \text{AM-S-M} \Rightarrow \text{AM-S-[M,P]}$$

An interesting aspect is that each generalized notion implies two restricted notions; AM-[S,M]-P and AM-[S,P]-M are each implied by one of the two generalized notions while AM-S-[M,P] is implied by both. It turns out, that the implications also hold in the other direction, meaning that two specialized notions together imply the generalized notion that implies them. This is formalized in the following two theorems. Note that the first one was already shown in [CDF+21], as it corresponds to the relation that S-CEO and S-DEO together imply S-UEO. The second one is new, since the generalized notion AM-S-M is novel.

Theorem 4. *For* AM ∈ {HON, LEAK, MAL}*, a signature scheme* S *is* AM-S-P *secure if and only if it is both* AM-[S,M]-P *and* AM-S-[M,P] *secure.*

Proof. The statement directly follows from the definitions of the notions: in AM-S-P, the public keys have to differ, while the signatures must agree; the same is true in AM-[S,M]-P and AM-S-[M,P], however, for the former the messages have to be the same, and for the latter they have to be different. Since in AM-S-P, no requirement is made for the messages, the combination of AM-[S,M]-P and AM-S-[M,P] corresponds exactly to AM-S-P. □

Theorem 5. *For* AM $\in \{\text{HON}, \text{LEAK}, \text{MAL}\}$*, a signature scheme* S *is* AM-S-M *secure if and only if it is both* AM-S-[M,P] *and* AM-[S,P]-M *secure.*

Proof. The statement directly follows from the definitions of the notions using the same reasoning as done in the previous proof. □

Remark 6. Theorem 5 shows an interesting connection, namely that the existing BUFF security notions MBS (AM-[S,P]-M) and DEO (AM-S-[M,P]) together imply our new notion AM-S-M. This means that MBS (AM-[S,P]-M) and DEO (AM-S-[M,P]) exhibit the same relation to AM-S-M as CEO (AM-[S,M]-P) and DEO (AM-S-[M,P]) exhibit towards UEO (AM-S-P). In prior works, the nomenclature clearly indicates a connection between CEO (AM-[S,M]-P) and DEO (AM-S-[M,P]). On the other hand, MBS (AM-[S,P]-M) typically seems to be a different security notion. Our framework reveals that MBS and DEO in fact are related as well, which also shows that DEO plays a central role as it is connected to both other restricted notions CEO and MBS—which itself are not directly related.

3.4 Separations

The following two theorems show separations for notions of the same notion class but different attack models (first separating honest and leakage setting, followed by separating leakage and malicious setting). The first theorem shows that security in the honest setting does not imply security in the leakage setting while the second theorem shows that security in the leakage setting does not yield security in the malicious setting. Clearly, to make the statements of the two theorems not vacuous, we need schemes that satisfy the requirements. Looking ahead, we will later see how to achieve all security notions, which thus shows that there are schemes satisfying the requirements of Theorem 7 and Theorem 8.

The theorem below states that security for HON notions does not imply security for LEAK notions. It relies on the signature scheme shown in Fig. 5. Here, a random value x is added to the secret key while the output y of x under a one-way function is added to the public key. Signatures have an additional component z such that if z is a preimage of y, the signature will always be accepted. This change does not affect security for HON notions as honest signatures will always have $z = \bot$. For LEAK notions, the adversary can easily obtain a preimage from the secret key.

Theorem 7. *Let* S *be a signature scheme,* F *be a one-way function, and* $\mathtt{S}^*$ *be the signature scheme displayed in Fig. 5. Then the following statements hold:*

1. *if* S *is* HON-[S,P]-M*, then* $\mathtt{S}^*$ *is* HON-[S,P]-M *but not* LEAK-[S,P]-M
2. *if* S *is* HON-S-[M,P]*, then* $\mathtt{S}^*$ *is* HON-S-[M,P] *but not* LEAK-S-[M,P]
3. *if* S *is* HON-[S,M]-P*, then* $\mathtt{S}^*$ *is* HON-[S,M]-P *but not* LEAK-[S,M]-P
4. *if* S *is* HON-S-M*, then* $\mathtt{S}^*$ *is* HON-S-M *but not* LEAK-S-M
5. *if* S *is* HON-S-P*, then* $\mathtt{S}^*$ *is* HON-S-P *but not* LEAK-S-P

Proof. Observe that $\mathtt{S}^*$ is identical to $\mathtt{S}$ unless a signature contains the preimage x of the value y appended to the public key in which case the special case for verification is triggered which accepts the signature for any message. Note further that honestly generated signatures never trigger the special case for verification.

Regarding the Hon notions, observe that an adversary $\mathcal{A}$ is restricted to honestly generated signatures and hence every signature will be of the form $(\mathtt{sig}, \bot)$. Hence the check $z \neq \bot$ will never succeed, i.e., $\mathcal{A}$ cannot trigger the special verification case. This establishes that $\mathtt{S}^*$ inherits Hon-[S,P]-M, Hon-S-[M,P], and Hon-[S,M]-P security from the underlying signature scheme $\mathtt{S}$.

Regarding Leak-[S,P]-M, the following attack is possible. Given an honestly generated key pair $(\mathtt{pk}^*, \mathtt{sk}^*)$, with $\mathtt{pk}^* = (\mathtt{pk}, y)$ and $\mathtt{sk}^* = (\mathtt{sk}, x)$, an adversary $\mathcal{A}$ can pick an arbitrary signature $\mathtt{sig}$, set $\mathtt{sig}^* \leftarrow (\mathtt{sig}, x)$, and output $(\mathtt{sig}^*, \mathtt{msg}, \overline{\mathtt{msg}})$ for arbitrary messages $\mathtt{msg} \neq \overline{\mathtt{msg}}$. Since $\mathtt{F}(x) = y$, $\mathtt{sig}^*$ triggers the special case for verification which accepts irrespectively of the message, i.e., $\mathtt{S}^*.\mathtt{Verify}(\mathtt{pk}^*, \cdot, \mathtt{sig}^*) = 1$.

Regarding Leak-S-[M,P], the following attack is possible. Given an honestly generated key pair $(\mathtt{pk}^*, \mathtt{sk}^*)$, with $\mathtt{pk}^* = (\mathtt{pk}, y)$ and $\mathtt{sk}^* = (\mathtt{sk}, x)$, an adversary $\mathcal{A}$ can pick an arbitrary signature $\mathtt{sig}$, set $\mathtt{sig}^* \leftarrow (\mathtt{sig}, x)$. Furthmore, $\mathcal{A}$ sets $\overline{\mathtt{pk}}^* \leftarrow (\overline{\mathtt{pk}}, y)$, for $\overline{\mathtt{pk}} \neq \mathtt{pk}$ and outputs $(\mathtt{sig}^*, \mathtt{msg}, \overline{\mathtt{msg}}, \overline{\mathtt{pk}}^*)$. By construction, the public keys and messages are different, and, since $\mathtt{F}(x) = y$, $\mathtt{sig}^*$ triggers the special case for verification which accepts irrespectively of the message, i.e., $\mathtt{S}^*.\mathtt{Verify}(\mathtt{pk}^*, \mathtt{msg}, \mathtt{sig}^*) = \mathtt{S}^*.\mathtt{Verify}(\mathtt{pk}^*, \overline{\mathtt{msg}}, \mathtt{sig}^*) = 1$.

Regarding Leak-[S,M]-P, the above attack for Leak-S-[M,P] applies, with the mere difference that $\mathcal{A}$ only picks a single message that it outputs.

Lastly, the above results together with Theorem 4 and Theorem 5 imply that Leak-S-P and Leak-S-M are not fulfilled. □

Note that the theorem also holds without using a one-way function to "hide" the value x in the public key. By using a one-way function, the transformed scheme remains unforgeable. While not strictly necessary, one might wonder how relevant the separation is, if the scheme does not achieve the minimum security though, which is why we opted to make use of it.

The next theorem establishes the separation between the leakage setting and the malicious setting and relies on the signature scheme shown in Fig. 6. Here, both public keys and signatures have a trailing bit. For honestly generated keys and signature, the bits are 1. Honestly generated public keys can only verify signatures with a trailing 1, in which case the normal verification algorithm is done. Malicious public keys (having a trailing 0) will accept any malicious signature (also having a trailing 0). If the trailing bits of public key and signature are different, verification will always fail. In the leakage setting, the adversary receives an honestly generated key pair, which will have a trailing 1 and thus can never trigger the special case of verification. However, in the malicious setting, the adversary can simply choose public key and signature with trailing 0s to trigger the special verification check.

Theorem 8. *Let* $\mathtt{S}$ *be a signature scheme and* $\mathtt{S}^*$ *be the signature scheme displayed in Fig. 6. Then the following statements hold:*

1. *if* $\mathtt{S}$ *is* Leak-[S,P]-M*, then* $\mathtt{S}^*$ *is* Leak-[S,P]-M *but not* Mal-[S,P]-M
2. *if* $\mathtt{S}$ *is* Leak-S-[M,P]*, then* $\mathtt{S}^*$ *is* Leak-S-[M,P] *but not* Mal-S-[M,P]
3. *if* $\mathtt{S}$ *is* Leak-[S,M]-P*, then* $\mathtt{S}^*$ *is* Leak-[S,M]-P *but not* Mal-[S,M]-P
4. *if* $\mathtt{S}$ *is* Leak-S-M*, then* $\mathtt{S}^*$ *is* Leak-S-M *but not* Mal-S-M
5. *if* $\mathtt{S}$ *is* Leak-S-P*, then* $\mathtt{S}^*$ *is* Leak-S-P *but not* Mal-S-P

Proof. Regarding the Leak notions, observe that the honestly generated public key that an adversary $\mathcal{A}$ obtains will have a trailing 1. By construction, this public key can only verify signatures that have a trailing 1 as well. In this case, however, $\mathtt{S}^*$ is identical to $\mathtt{S}$, hence Leak-[S,P]-M, Leak-S-[M,P], and Leak-[S,M]-P security of $\mathtt{S}^*$ is inherited from the respective security of $\mathtt{S}$.

Regarding the Mal notions, observe that $\mathtt{S}^*$ accepts everything if both public key and signature have a trailing 0. An adversary $\mathcal{A}$ can pick two different public keys $\mathtt{pk}^* \leftarrow (\mathtt{pk}, 0)$ and $\overline{\mathtt{pk}}^* \leftarrow (\overline{\mathtt{pk}}, 0)$ and an arbitrary signature $\mathtt{sig}^* \leftarrow (\mathtt{sig}, 0)$, all with a trailing 0. By construction, $\mathtt{S}^*.\mathtt{Verify}(\mathtt{pk}^*, \cdot, \mathtt{sig}^*) = \mathtt{S}^*.\mathtt{Verify}(\overline{\mathtt{pk}}^*, \cdot, \mathtt{sig}^*) = 1$, irrespective of the message. Depending on the exact notion, $\mathcal{A}$ outputs $(\mathtt{sig}^*, \mathtt{msg}, \overline{\mathtt{msg}}, \mathtt{pk}^*)$ (Mal-[S,P]-M), $(\mathtt{sig}^*, \mathtt{msg}, \overline{\mathtt{msg}}, \mathtt{pk}^*, \overline{\mathtt{pk}}^*)$ (Mal-S-[M,P]), or $(\mathtt{sig}^*, \mathtt{msg}, \mathtt{pk}^*, \overline{\mathtt{pk}}^*)$ (Mal-[S,M]-P) to break the corresponding security notion.

The above results together with Theorem 4 and Theorem 5 imply that Leak-S-P and Leak-S-M are not fulfilled. □

The following theorems show separations between the various restricted notions. In particular, we show the following separation:

$$\text{Am-S-[M,P]} \not\Rightarrow \text{Am-[S,M]-P} \qquad \text{Am-S-[M,P]} \not\Rightarrow \text{Am-[S,P]-M}$$
$$\text{Am-[S,M]-P} \not\Rightarrow \text{Am-S-[M,P]} \qquad \text{Am-[S,P]-M} \not\Rightarrow \text{Am-S-[M,P]}$$

The proofs are deferred to Appendix A. The first two separations (Am-S-[M,P] $\not\Rightarrow$ Am-[S,M]-P and Am-[S,M]-P $\not\Rightarrow$ Am-S-[M,P]) are due to [CDF+21]; we merely recast them in our framework and argue that they hold for all three attack models (the corresponding results in [CDF+21] consider only the Hon case where Am = Hon). For the third separation, we give a new construction. For the forth separation, we do not use an artificial signature scheme but UOV, leveraging the results by Aulbach et al. [ADM+24]. For all separations, the proofs hold regardless of the attack model.

Theorem 9 (Am-S-[M,P] $\not\Rightarrow$ Am-[S,M]-P). *There exists a signature scheme that is* Am-S-[M,P] *but not* Am-[S,M]-P.

Theorem 10 (Am-[S,M]-P $\not\Rightarrow$ Am-S-[M,P]). *There exists a signature scheme that is* Am-[S,M]-P *but not* Am-S-[M,P].

Theorem 11 (Am-S-[M,P] $\not\Rightarrow$ Am-[S,P]-M). *There exists a signature scheme that is* Am-S-[M,P] *but not* Am-[S,P]-M.

Theorem 12 (Am-[S,P]-M $\not\Rightarrow$ Am-S-[M,P]). *There exists a signature scheme that is* Am-[S,P]-M *but not* Am-S-[M,P].

$\mathtt{KeyGen}^*()$	$\mathtt{Sign}^*(\mathtt{sk}^*, \mathtt{msg})$	$\mathtt{Verify}^*(\mathtt{pk}^*, \mathtt{msg}, \mathtt{sig}^*)$
$(\mathtt{pk}, \mathtt{sk}) \leftarrow\!\!\$\ \mathtt{KeyGen}()$	$(\mathtt{sk}, x) \leftarrow \mathtt{sk}^*$	$(\mathtt{pk}, y) \leftarrow \mathtt{pk}^*$
$x \leftarrow\!\!\$\ \mathcal{X}$	$\mathtt{sig} \leftarrow\!\!\$\ \mathtt{Sign}(\mathtt{sk}, \mathtt{msg})$	$(\mathtt{sig}, z) \leftarrow \mathtt{sig}^*$
$y \leftarrow \mathtt{F}(x)$	$\mathtt{sig}^* \leftarrow (\mathtt{sig}, \bot)$	**if** $[\![z \neq \bot \wedge \mathtt{F}(z) = y]\!]$
$\mathtt{sk}^* \leftarrow (\mathtt{sk}, x)$	**return** $\mathtt{sig}^*$	**return** 1
$\mathtt{pk}^* \leftarrow (\mathtt{pk}, y)$		**return** $\mathtt{Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig})$
return $(\mathtt{pk}^*, \mathtt{sk}^*)$		

Fig. 5: Separation example for HON-B-T $\not\Rightarrow$ LEAK-B-T (Theorem 7).

$\mathtt{KeyGen}^*()$	$\mathtt{Sign}^*(\mathtt{sk}, \mathtt{msg})$	$\mathtt{Verify}^*(\mathtt{pk}^*, \mathtt{msg}, \mathtt{sig}^*)$
$(\mathtt{pk}, \mathtt{sk}) \leftarrow\!\!\$\ \mathtt{KeyGen}()$	$\mathtt{sig} \leftarrow\!\!\$\ \mathtt{Sign}(\mathtt{sk}, \mathtt{msg})$	$(\mathtt{pk}, b) \leftarrow \mathtt{pk}^*$
$\mathtt{pk}^* \leftarrow (\mathtt{pk}, 1)$	$\mathtt{sig}^* \leftarrow (\mathtt{sig}, 1)$	$(\mathtt{sig}, d) \leftarrow \mathtt{sig}^*$
return $(\mathtt{pk}^*, \mathtt{sk})$	**return** $\mathtt{sig}^*$	**if** $[\![b \neq d]\!]$
		return 0
		if $[\![b = 0 \wedge d = 0]\!]$
		return 1
		if $[\![b = 1 \wedge d = 1]\!]$
		return $\mathtt{Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig})$

Fig. 6: Separation example for LEAK-B-T $\not\Rightarrow$ MAL-B-T (Theorem 8).

4 Analysis of Transformations

In this section, we analyze the existing transformations that achieve (partial) BUFF security with respect to our expanded framework. There are a total of five transformations: PS-1, PS-2, PS-3, BUFF-lite, and BUFF. We start with a description of these transforms and the known results regarding the BUFF notions they achieve in Section 4.1. Out of these transforms, only one—the PS-3 transform—does not increase the signature size, which offers a significant benefit. Thus, we start with the analysis of the PS-3 transform in Section 4.2, and analyze the signature-increasing transforms, i.e., the PS-1, PS-2, BUFF-lite, and BUFF transform in Section 4.3. Lastly, in Section 4.4, we give an updated overview of the results for the transformations with respect to our new framework.

Whenever we give an overview of existing results in the following, we will use the BUFF nomenclature from prior works and assign our new notation in brackets. Otherwise, we will stick to our new naming convention.

4.1 Overview of the existing Transformations

Firstly, there are three transformations that were introduced in [PS05] and further analyzed in [CDF+21]: PS-1, PS-2, and PS-3. For a signature scheme $\mathtt{S}$,

$\underline{\mathtt{Sign}^*(\mathtt{sk},\mathtt{msg})}$
$\mathtt{sig} \leftarrow \mathtt{Sign}(\mathtt{sk},\mathtt{msg})$
$\mathtt{h} \leftarrow \mathsf{H}(\mathtt{msg})$
$\mathtt{sig}^* \leftarrow (\mathtt{sig},\mathtt{h})$
return $\mathtt{sig}^*$

$\underline{\mathtt{Sign}^*(\mathtt{sk},\mathtt{msg})}$
$\mathtt{sig} \leftarrow \mathtt{Sign}(\mathtt{sk},\mathtt{msg})$
$\mathtt{h} \leftarrow \mathsf{H}(\mathtt{pk})$
$\mathtt{sig}^* \leftarrow (\mathtt{sig},\mathtt{h})$
return $\mathtt{sig}^*$

$\underline{\mathtt{Sign}^*(\mathtt{sk},\mathtt{msg})}$
$\mathtt{h} \leftarrow \mathsf{H}(\mathtt{pk},\mathtt{msg})$
$\mathtt{sig} \leftarrow \mathtt{Sign}(\mathtt{sk},\mathtt{h})$
return $\mathtt{sig}$

$\underline{\mathtt{Verify}^*(\mathtt{pk},\mathtt{msg},\mathtt{sig}^*)}$
$(\mathtt{sig},\mathtt{h}) \leftarrow \mathtt{sig}^*$
$\overline{\mathtt{h}} \leftarrow \mathsf{H}(\mathtt{msg})$
$\mathtt{v} \leftarrow \mathtt{Verify}(\mathtt{pk},\mathtt{msg},\mathtt{sig})$
return $[\![\mathtt{v} = 1 \wedge \overline{\mathtt{h}} = \mathtt{h}]\!]$

$\underline{\mathtt{Verify}^*(\mathtt{pk},\mathtt{msg},\mathtt{sig}^*)}$
$(\mathtt{sig},\mathtt{h}) \leftarrow \mathtt{sig}^*$
$\overline{\mathtt{h}} \leftarrow \mathsf{H}(\mathtt{pk})$
$\mathtt{v} \leftarrow \mathtt{Verify}(\mathtt{pk},\mathtt{msg},\mathtt{sig})$
return $[\![\mathtt{v} = 1 \wedge \overline{\mathtt{h}} = \mathtt{h}]\!]$

$\underline{\mathtt{Verify}^*(\mathtt{pk},\mathtt{msg},\mathtt{sig})}$
$\mathtt{h} \leftarrow \mathsf{H}(\mathtt{pk},\mathtt{msg})$
$\mathtt{v} \leftarrow \mathtt{Verify}(\mathtt{pk},\mathtt{h},\mathtt{sig})$
return $[\![\mathtt{v} = 1]\!]$

Fig. 7: Transforms PS-1 (left), PS-2 (middle), and PS-3 (right).

the signature after application of the PS-1 transform (shown in Fig. 7) is $\mathtt{sig} = \mathtt{S.Sign}(\mathtt{sk},\mathtt{msg})||\mathsf{H}(\mathtt{msg})$ and it was proven that—out of the BUFF notions—the resulting scheme achieves S-DEO (Hon-S-[M,P]) and MBS (Mal-[S,P]-M). For the PS-2 transform (shown in Fig. 7), the signature is $\mathtt{sig} = \mathtt{S.Sign}(\mathtt{sk},\mathtt{msg})||\mathsf{H}(\mathtt{pk})$ and a transformed scheme achieves M-S-UEO (Mal-S-P). For the PS-3 transform (shown in Fig. 7), the signature is computed as $\mathtt{sig} = \mathtt{S.Sign}(\mathtt{sk},\mathsf{H}(\mathtt{msg},\mathtt{pk}))$ and the transform achieves S-UEO (Hon-S-P) security requiring some additional assumption related to weak keys.

Next to this, [CDF+21] introduced two further transforms: the BUFF-lite and the BUFF transform. For the BUFF-lite transform (shown in Fig. 8), signatures are of the form $\mathtt{sig} = \mathtt{S.Sign}(\mathtt{sk},\mathtt{msg})||\mathsf{H}(\mathtt{msg},\mathtt{pk})$ and M-S-UEO (Mal-S-P) and MBS (Mal-[S,P]-M) are achieved. The BUFF transform (shown in Fig. 8) computes signatures as $\mathtt{sig} = \mathtt{S.Sign}(\mathtt{sk},\mathsf{H}(\mathtt{msg},\mathtt{pk}))||\mathsf{H}(\mathtt{msg},\mathtt{pk})$ and achieves M-S-UEO (Mal-S-P), MBS (Mal-[S,P]-M) and NR. An overview of the existing positive results for the different transforms is given in Table 2.

To summarize, all of the transforms either append a hash value to the signature (PS-1, PS-2, and BUFF-lite) or sign not the message but the hash of message and public key (PS-3)—or do both (BUFF). In particular, for all transforms, the signing time increases while for all except PS-3 also the signature size increases.

4.2 Analysis of the PS-3 Transform

The PS-3 transform is of special interest, as it is the only transform that does not increase the signature size. At the same time, its analysis is the most involved, as so-called weak keys have to be taken into account. This observation has been made in various prior works, starting with the very first paper [PS05]. Here, a property for signatures schemes is defined, which requires that for each public key $\mathtt{pk}$ and signature $\mathtt{sig}$—that have passed some basic correctness tests implemented by the verifier—the fraction of messages $\mathtt{msg}$ in the message space

$\underline{\mathtt{Sign}^*(\mathtt{sk},\mathtt{msg})}$
$\mathtt{sig} \leftarrow \mathtt{Sign}(\mathtt{sk},\mathtt{msg})$
$\mathtt{h} \leftarrow \mathsf{H}(\mathtt{pk},\mathtt{msg})$
$\mathtt{sig}^* \leftarrow (\mathtt{sig},\mathtt{h})$
return $\mathtt{sig}^*$

$\underline{\mathtt{Sign}^*(\mathtt{sk},\mathtt{msg})}$
$\mathtt{h} \leftarrow \mathsf{H}(\mathtt{pk},\mathtt{msg})$
$\mathtt{sig} \leftarrow \mathtt{Sign}(\mathtt{sk},\mathtt{h})$
$\mathtt{sig}^* \leftarrow (\mathtt{sig},\mathtt{h})$
return $\mathtt{sig}^*$

$\underline{\mathtt{Verify}^*(\mathtt{pk},\mathtt{msg},\mathtt{sig}^*)}$
$(\mathtt{sig},\mathtt{h}) \leftarrow \mathtt{sig}^*$
$\bar{\mathtt{h}} \leftarrow \mathsf{H}(\mathtt{pk},\mathtt{msg})$
$\mathtt{v} \leftarrow \mathtt{Verify}(\mathtt{pk},\mathtt{msg},\mathtt{sig})$
return $[\![\mathtt{v} = 1 \wedge \bar{\mathtt{h}} = \mathtt{h}]\!]$

$\underline{\mathtt{Verify}^*(\mathtt{pk},\mathtt{msg},\mathtt{sig}^*)}$
$(\mathtt{sig},\mathtt{h}) \leftarrow \mathtt{sig}^*$
$\bar{\mathtt{h}} \leftarrow \mathsf{H}(\mathtt{pk},\mathtt{msg})$
$\mathtt{v} \leftarrow \mathtt{Verify}(\mathtt{pk},\bar{\mathtt{h}},\mathtt{sig})$
return $[\![\mathtt{v} = 1 \wedge \bar{\mathtt{h}} = \mathtt{h}]\!]$

Fig. 8: Transforms BUFF-lite (left) and BUFF (right).

Table 2: Positive results from prior works [PS05, CDF+21] regarding the five existing transforms. The asterisks in the PS-3 column indicate that the results are tied to an assumption (absence of weak keys).

	PS-1			PS-2			PS-3			BUFF-lite			BUFF		
	M	L	H	M	L	H	M	L	H	M	L	H	M	L	H
AM-S-M															
AM-[S,P]-M	✓									✓			✓		
AM-S-[M,P]			✓			✓			✓⋆			✓			✓
AM-[S,M]-P						✓			✓⋆			✓			✓
AM-S-P					✓	✓			✓⋆	✓	✓	✓	✓	✓	✓

for which $\mathtt{Verify}(\mathtt{pk},\mathtt{msg},\mathtt{sig}) = 1$, is negligible. Note that this property is not restricted to honestly generated keys, but also comprises ones outside the image of key generation—as such can also be chosen by an adversary in the S-UEO (HON-S-P) game. Under this assumption, the PS-3 transform suffices to achieve S-UEO (HON-S-P)[8] security.

In [CDF+21], weak keys are informally described as keys that verify multiple or even all messages and it is shown that—contrary to the intuition—weak keys cannot only occur outside of $\mathcal{HGK}$. This means that it does not suffice for the verifier to check whether a key is in the image of `KeyGen` to obtain S-UEO (HON-S-P) just from the PS-3 transform. In particular, it seems that one cannot

[8] In [PS05] the slightly weaker notion UEO is considered, where the adversary is not provided access to a signing oracle but only a single pair of message and signature instead. However, in [CDF+21] it was shown that all results from [PS05] regarding UEO security transfer to the stronger notion S-UEO.

get around weak keys when analyzing the PS-3 transform. Next to the S-UEO (Hon-S-P) results, [CDF+21] also describe that the remaining BUFF properties M-S-UEO (Mal-S-P), MBS (Mal-[S,P]-M), and NR[9] cannot be achieved by the PS-3 transform without any additional assumptions.

In [DS24], the connection between MBS (Mal-[S,P]-M) and weak keys (as they are characterized in [CDF+21]) is brought to attention: as MBS (Mal-[S,P]-M) excludes that an adversary can find a public key that verifies two messages for the same signature, it especially prevents the adversary from finding a public key that verifies many messages. Under the assumption that the underlying signature scheme, i.e., *before* applying PS-3, achieves MBS (Mal-[S,P]-M), [DS24] shows that the PS-3 transform suffices to achieve achieve S-UEO (Hon-S-P) and wNR while maintaining MBS (Mal-[S,P]-M). However, there are two disadvantages: First, the resulting bound for S-UEO (Hon-S-P) is quite loose, which leaves room for improvement. Second, the result comes with an explicit counter example that PS-3 does *not* achieve M-S-UEO (Mal-S-P) security.

While MBS (Hon-[S,P]-M) excludes weak keys as defined in [CDF+21], it does not exclude the existence of public keys $\mathtt{pk}$ and $\overline{\mathtt{pk}}$ for which, given two random messages $\mathtt{msg}$ and $\overline{\mathtt{msg}}$, one can find a signature $\mathtt{sig}$ such that $\mathtt{Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig}) = 1$ and $\mathtt{Verify}(\overline{\mathtt{pk}}, \overline{\mathtt{msg}}, \mathtt{sig}) = 1$ hold with non-negligible probability. Then the following M-S-UEO (Mal-S-P) attack against a PS-3-transformed scheme is possible—even if the underlying scheme fulfilled MBS (Hon-[S,P]-M) security—: Firstly, pick two keys $\mathtt{pk}$ and $\overline{\mathtt{pk}}$ with the described property; secondly choose arbitrary $\mathtt{msg}$ and $\overline{\mathtt{msg}}$ and compute $\mathtt{h} = \mathsf{H}(\mathtt{pk}, \mathtt{msg})$ and $\overline{\mathtt{h}} = \mathsf{H}(\overline{\mathtt{pk}}, \overline{\mathtt{msg}})$; thirdly, determine $\mathtt{sig}$ such that $\mathtt{Verify}(\mathtt{pk}, \mathtt{h}, \mathtt{sig}) = 1$ and $\mathtt{Verify}(\overline{\mathtt{pk}}, \overline{\mathtt{h}}, \mathtt{sig}) = 1$. In [DS24], it is shown that such keys exist for the multivariate signature scheme UOV [BCD+23].

In summary, the security of PS-3-transformed schemes regarding the various notions we consider (and introduce) in this work, is quite unclear. The PS-3 transform achieves Hon-S-P (under some conditions related to weak keys) while it generally does not achieve Mal-S-P. Furthermore, it is known that the PS-3 transform maintains Mal-[S,P]-M security of the underlying signature scheme (which then also yields the required conditions to achieve Hon-S-P). The situation for several other notions (Mal-[S,M]-P, Mal-S-[M,P], Leak-S-P, Leak-[S,M]-P, Leak-S-[M,P]) is unclear at the moment.

In the following we will give an updated formal weak key definition and prove that for a scheme without such weak keys, the PS-3 transform suffices to achieve *all* notions from our framework introduced in Section 3. In particular, this definition covers also the kind of keys that allow the Mal-S-P attack against UOV in [DS24]. Also, while our result requires a scheme-specific analysis to check for weak keys, it can yield better bounds.

In the following we give a rigorous definition of weak keys.

[9] Note that this result is with respect to the old non-resignability definition, which has been been proven faulty [DFHS24].

Definition 13. *For a signature scheme* $\mathtt{S}$*, we call a public key* $\mathtt{pk} \in \mathcal{PK}$ *of this scheme weak if, there is an adversary* $\mathcal{A}$ *such that the probability*

$$\Pr[\mathtt{Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig}) = 1 \mid \mathtt{sig} \leftarrow \mathcal{A}(\mathtt{pk}, \mathtt{msg}),\ \mathtt{msg} \leftarrow\!\!\$\ \mathcal{M}]$$

is non-negligible.

Compared to the literature, our weak key definition is closest to the one from [PS05]. Here, a scheme is said to have no weak keys if for each public key $\mathtt{pk}$ and signature $\mathtt{sig}$, the fraction of messages $\mathtt{msg}$ in the message space for which $\mathtt{Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig}) = 1$, is negligible. However, this definition does not involve an adversary, but is a general property of the signature scheme.

Regarding the relation between the notions, one easily observes that weak keys by definition of [PS05] are also weak in our setting, while we will prove the converse to be false. In the following, we show that, while being harder to fulfill, our definition allows to achieve all notions from our framework. In contrast, we prove that the weak key definition following [PS05] is *not* sufficient. We consider the transform T described in Fig. 9 that adapts signing by appending $\perp$ to each signature. Further, verification of a message $\mathtt{msg}$ and a signature $\mathtt{sig}^* = (\mathtt{sig}, b)$ works as follows: if $b = \perp$, $\mathtt{Verify}$ of the underlying schemes is called, if $b \neq \perp$, output 1 if and only if $b = \mathtt{msg}$ and 0 otherwise. We consider a signature scheme $\mathtt{S}$ that has no weak keys according to the definition of [PS05] and observe that this property is preserved after application of the transform from Fig. 9. This is due to the fact that $T(\mathtt{S})$ verifies only exactly one message more for each pair of public key and signature. In particular, if the fraction of messages verifying a pair of public key and signature was negligible before application of the transform, that will also be the case after. In contrast, for the transformed scheme $T(\mathtt{S})$ all keys are weak by our definition: Given a public key and a random message $\mathtt{msg}$, the adversary can always output $\mathtt{sig}^* = (\mathtt{sig}, \mathtt{msg})$ for an arbitrarily chosen $\mathtt{sig}$. By definition of the transform, $\mathtt{sig}^*$ will verify for the message $\mathtt{msg}$ via the special verification case. In summary, this shows that weak keys in our definition are not necessarily weak in the definition by [PS05] and thus our definition is strictly stronger.

Next, we argue why this stronger definition is necessary for achieving all notions from the new framework. For this, we consider the above scheme after additionally applying the PS-3 transform. Note that this scheme still has no weak keys as defined in [PS05]. The following Leak-[S,M]-P attack against PS-3[T[S]] is possible: Given $(\mathtt{pk}, \mathtt{sk})$, the adversary chooses a random message $\mathtt{msg}$, signature $\mathtt{sig}$, and public key $\overline{\mathtt{pk}} \neq \mathtt{pk}$. Then, $\mathtt{sig}^* = (\mathtt{sig}, \mathtt{msg})$ will verify under both $\mathtt{pk}$ and $\overline{\mathtt{pk}}$.[10]

Remark 14. Definition 13 could also be formulated in a way that does not completely exclude the existence of weak keys, but instead just requires that it is computationally hard to find such keys. While the latter variant would be more

[10] Note that this attack does not apply against Hon-S-[M,P], as the signature has to stem from an query to the sign oracle (and hence will be of the form $\mathtt{sig}^* = (\mathtt{sig}, \perp)$).

general, we opt for the former one, as in all examples we are aware of, there are either no weak keys or the ones that exist are easy to find.

Note that, our weak key definition is not restricted to keys in the image of $\texttt{KeyGen}$ as an adversary in the MAL games can choose also maliciously generated keys. In the following we give a number of examples for schemes with weak keys: The SCHNORR signature scheme [Sch91] has two weak keys that lie outside the image of $\texttt{KeyGen}$, while for Lyubashevsky's signature scheme [Lyu12] and the multivariate signature scheme UOV [BCD+24], also weak keys inside the image exist.

Weak Keys of the SCHNORR *Signature Scheme.* The SCHNORR signature scheme is displayed in Fig. 10. For a weak key $\texttt{pk} = Z$ of the SCHNORR signature scheme, there has to be an adversary $\mathcal{A}$, such that, given $\texttt{pk}$ and an arbitrary message $\texttt{msg}$, $\mathcal{A}$ finds a signature $\texttt{sig} = (e, s)$ with $\texttt{Verify}(\texttt{pk}, \texttt{msg}, \texttt{sig}) = 1$, i.e., $\mathsf{H}(\texttt{msg}, g^s Z^{-e}) = e$ with non-negligible probability. Firstly note that by varying e also the target value of the hash computation is changed (i.e., this is a moving target). On the other hand, for some fixed e, the problem of finding a matching s requires solving a discrete logarithm problem.

However, the cases $\texttt{pk} = Z \in \{0, 1\}$ represent exceptions, in which the moving target problem can be bypassed—note that these values never occur during honest key generation as honestly generated keys are of the form $\texttt{pk} = Z = g^z$ for $z \leftarrow\!\!\$\, [q-1]$ and g a generator of order q. Simply put, for $\texttt{pk} = Z \in \{0, 1\}$, e does not influence the value of $g^s Z^{-e}$. Given a random message $\texttt{msg}$, the adversary chooses a random value for s, computes $\mathsf{H}(\texttt{msg}, g^s)$ and sets e to be the result of the latter computation. Then $\mathsf{H}(\texttt{msg}, g^s Z^{-e}) = e$ will hold, i.e., the signature (s, e) verifies.

So in conclusion, the SCHNORR signature scheme has two weak keys, namely $\texttt{pk} = Z \in \{0, 1\}$, which are not in the image of $\texttt{KeyGen}$.

Weak Keys in Lyubashevsky's Signature Scheme. We consider Lyubashevsky's signature scheme [Lyu12] as described in Fig. 11 based on the SIS problem. For a weak key $\texttt{pk} = (A, T)$ of this signature scheme, there has to be an adversary $\mathcal{A}$, such that, given $\texttt{pk}$ and an arbitrary message $\texttt{msg}$, $\mathcal{A}$ finds a signature $\texttt{sig} = (z, c)$ with $\texttt{Verify}(\texttt{pk}, \texttt{msg}, \texttt{sig}) = 1$, i.e., $||z|| \leq \eta\sigma\sqrt{m}$ and $c = \mathsf{H}(Az - Tc, \texttt{msg})$. As for the SCHNORR signature scheme, varying c changes the target of the hash computation (i.e., this is again a moving target problem) and for a fixed c, finding a matching *short* z requires solving an SIS problem.

Thus, there are two types of weak keys: Those for which we can circumvent the moving target problem and those for which the SIS problem is not hard. For the former, all public keys of the form $(A, 0)$ for $A \in \mathbb{Z}_q^{n \times m}$ act as weak keys: for $T = 0$, c does not influence the value of $Az - Tc$ and thus c can be chosen *after* the hash is computed. Note that this works analogously to the attack described for the SCHNORR signature scheme. Further, we observe that these keys lie in the image of $\texttt{KeyGen}$ as for $S = 0$ all $(A, 0)$ with $A \in \mathbb{Z}_q^{n \times m}$ can occur as public keys. However, it is extremely unlikely for such a key to be the result of an honest

key generation and these cases are easily excluded by checking whether $T = 0$ during verification.

The other type of weak keys cannot be described as concretely: It comprises all public keys (A, T) for which A cannot yield hard SIS instances. Then, one can choose arbitrary $x \in \mathbb{Z}^m$ and $\mathtt{msg}$, compute $c = \mathsf{H}(x, \mathtt{msg})$, and solve $Az = x - Tc$ for z. Note that these type of weak keys do not exist for the SCHNORR signature scheme, where the generator g is a public parameter and not part of the public key (as is the case for the matrix A). Similarly, if A was a public parameter in Lyubashevsky's scheme, these weak keys could be prevented.

Weak Keys of the UOV *Signature Scheme.* UOV is a multivariate signature scheme that can be seen to have weak keys as defined in Definition 13 using the results from [DS24]. They show that for UOV, one can construct two public keys $\mathtt{pk}$ and $\overline{\mathtt{pk}}$, such that for randomly chosen messages $\mathtt{msg}$ and $\overline{\mathtt{msg}}$, the probability of finding a signature $\mathtt{sig}$ such that $\mathtt{Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig}) = 1$ and $\mathtt{Verify}(\overline{\mathtt{pk}}, \overline{\mathtt{msg}}, \mathtt{sig}) = 1$ is non-negligible.

On a high-level, a UOV secret key is a matrix $O \in K^{(n-m)\times m}$ (for K a finite field), that defines the so-called oil space as its image. The public key consists of n quadratic polynomials $p_1, \ldots, p_n$ which map the oil space to zero. Note that to each p_i, one can associate a matrix

$$P_i = \begin{pmatrix} P_i^{(1)} & P_i^{(2)} \\ 0 & P_i^{(3)} \end{pmatrix} \tag{1}$$

with $P_i^{(1)} \in K^{(n-m)\times(n-m)}, P_i^{(2)} \in K^{(n-m)\times m}$, and $P_i^{(3)} \in K^{m\times m}$ such that $p_i(x) = x^\top P_i x$ for any $x \in K^n$, and $P_i^{(1)}$ and $P_i^{(3)}$ are upper triangular matrices. To sign a message $\mathtt{msg}$, a vector $s \in K^n$ such that $s^\top P_i s = \mathsf{H}(\mathtt{msg})$ is determined using the oil space. Verification then checks whether this equation holds.

The key idea of finding $\mathtt{pk}$ and $\overline{\mathtt{pk}}$ as described above is then to pick a very large oil space (more precisely, $O \in K^{(n-2m)\times 2m}$) and two different public keys which map this oil space to zero. Note that for the associated matrices (as shown in Eq. (1)), the dimensions of the submatrices will be different, namely $P_i^{(1)} \in K^{(n-2m)\times(n-2m)}, P_i^{(2)} \in K^{(n-2m)\times 2m}$, and $P_i^{(3)} \in K^{2m\times 2m}$. Then, given two messages $\mathtt{msg}, \overline{\mathtt{msg}}$ and the corresponding targets $\mathsf{H}(\mathtt{msg}), \mathsf{H}(\overline{\mathtt{msg}})$, this choice of public keys allows to find a signature $\mathtt{sig}$ that verifies both targets with non-negligible probability.

In particular, public keys that are obtained like this, are weak by our definition. Note that large oil spaces as described above can never result from honestly generated secret keys as the matrix O is always chosen of dimension $(n-m)\times m$ during $\mathtt{KeyGen}$. However, it is possible for an honestly generated public key to map a larger space to zero than the oil space defined by the corresponding secret key. In particular, the public keys we describe above can be the result of $\mathtt{KeyGen}$ (though, it is unlikely).

So in conclusion, this shows that UOV has weak keys in the image of $\mathtt{KeyGen}$. One should be aware that, compared to the analysis we give for SCHNORR and

$\mathtt{KeyGen}^*()$	$\mathtt{Sign}^*(\mathtt{sk},\mathtt{msg})$	$\mathtt{Verify}^*(\mathtt{pk},\mathtt{msg},\mathtt{sig}^*)$
$(\mathtt{pk},\mathtt{sk}) \leftarrow\!\!\$\ \mathtt{KeyGen}()$	$\mathtt{sig} \leftarrow\!\!\$\ \mathtt{Sign}(\mathtt{sk},\mathtt{msg})$	$(\mathtt{sig}, z) \leftarrow \mathtt{sig}^*$
return $(\mathtt{pk},\mathtt{sk})$	$\mathtt{sig}^* \leftarrow (\mathtt{sig}, \bot)$	**if** $z = \bot$
	return $\mathtt{sig}^*$	**return** $[\![\mathtt{Verify}(\mathtt{pk},\mathtt{msg},\mathtt{sig})]\!]$
		if $z = \mathtt{msg}$
		return 1
		else return 0

Fig. 9: Transform T used to separate our weak key definition from [PS05].

$\mathtt{KeyGen}()$	$\mathtt{Sign}(\mathtt{sk},\mathtt{msg})$	$\mathtt{Verify}(\mathtt{pk},\mathtt{msg},\mathtt{sig})$
$z \leftarrow\!\!\$\ [q-1]$	$z \leftarrow \mathtt{sk}$	$(e, s) \leftarrow \mathtt{sig}$
$Z \leftarrow g^z$	$k \leftarrow\!\!\$\ [q-1]$	$Z \leftarrow \mathtt{pk}$
$\mathtt{sk} \leftarrow z$	$r \leftarrow g^k$	$r \leftarrow g^s Z^{-e}$
$\mathtt{pk} \leftarrow Z$	$e \leftarrow \mathsf{H}(\mathtt{msg}, r)$	**if** $\mathsf{H}(\mathtt{msg}, r) \neq e$
return $(\mathtt{pk},\mathtt{sk})$	$s \leftarrow k + ze$	**return** 0
	return $\mathtt{sig} \leftarrow (e, s)$	**return** 1

Fig. 10: Schnorr signature scheme. Here, q is a prime and g is a generator of the cyclic group of order q underlying the signature scheme.

Lyubashevsky's scheme, this does not provide a complete description of UOV's weak keys—there might be more inside as well as outside the image of `KeyGen`.

Weak-Key-Checking. Depending on the scheme, efficient checks to exclude the existence of weak keys can be possible.[11] We formalize this by defining a function `WK` that takes as input a public key `pk` and outputs 1 if `pk` is a weak key and 0 otherwise. For a signature scheme `S`, we say that `S` deploys weak-key-checking if the verification algorithm is modified to additionally run `WK` on the given public key and reject if the output is 0.

From our examples, the Schnorr signature scheme allows such checking. In the case of Schnorr signatures, none of the keys in the image of `KeyGen` are weak, i.e., it suffices to check whether a key is honestly generated to exclude weak keys. The latter can be done by defining $\mathtt{WK}(\mathtt{pk})$ to be 1 if $\mathtt{pk} \in \{0, 1\}$ and 0 otherwise.

The following theorem shows that a PS-3-transformed signature schemes that deploys weak-key-checking, achieves all of the notions from our framework.

Theorem 15. *If applied to a signature scheme that deploys weak-key-checking, the* PS-3 *transform produces a signature scheme that fulfills all 15 notions shown in Table 1.*

[11] Note that [PS05] also mentions "basic correctness checks" which are implemented by the verifier.

$\texttt{KeyGen}()$	$\texttt{Sign}(\texttt{sk}, \texttt{msg})$	$\texttt{Verify}(\texttt{pk}, \texttt{msg}, \texttt{sig})$
$S \leftarrow\!\!\$\ \{-d, \ldots, 0, \ldots, d\}^{m \times k}$	$S \leftarrow \texttt{sk}$	$(z, c) \leftarrow \texttt{sig}$
$A \leftarrow\!\!\$\ \mathbb{Z}_q^{n \times m}$	$y \leftarrow\!\!\$\ D_\sigma^m$	**if** $\|\|z\|\| > \eta\sigma\sqrt{m}$
$T \leftarrow AS$	$c \leftarrow \mathsf{H}(Ay, \texttt{msg})$	**return** 0
$\texttt{sk} \leftarrow S$	$z \leftarrow Sc + y$	$c' \leftarrow \mathsf{H}(Az - Tc, \texttt{msg})$
$\texttt{pk} \leftarrow (A, T)$	With probability p:	**return** $[\![c = c']\!]$
return $(\texttt{pk}, \texttt{sk})$	**return** $\texttt{sig} \leftarrow (z, c)$	
	Otherwise restart	

Fig. 11: Lyubashevsky's signature scheme based on the SIS problem as described in [Lyu12]. Here, $d, n, m, k, q, \sigma \in \mathbb{N}$ and $\eta, M \in \mathbb{R}$ are parameters of the scheme. Further, D_σ^m and $D_{Sc,\sigma}^m$ denote the discrete Gaussian distribution over $\mathbb{Z}^m$ with standard deviation σ, centered in 0 and Sc, respectively. Lastly, the probability p used in $\texttt{Sign}$ is computed as $p = \min\left(D_\sigma^m(z)(MD_{Sc,\sigma}^m(z))^{-1}, 1\right)$.

Proof. Denote by S the signature scheme after the PS-3 transform is applied. We show that S achieves Mal-[S,P]-M and Mal-S-P security, which then implies all other notions by our prior results.

In order to break Mal-[S,P]-M, the adversary has to find two different messages $\texttt{msg} \neq \overline{\texttt{msg}}$, a signature $\texttt{sig}$, and a public key $\texttt{pk}$ such that both $\mathsf{S}.\texttt{Verify}(\texttt{pk}, \mathsf{H}(\texttt{pk}, \texttt{msg}), \texttt{sig}) = 1$ and $\mathsf{S}.\texttt{Verify}(\texttt{pk}, \mathsf{H}(\texttt{pk}, \overline{\texttt{msg}}), \texttt{sig}) = 1$. In order to compute $\mathsf{H}(\texttt{pk}, \texttt{msg})$ and $\mathsf{H}(\texttt{pk}, \overline{\texttt{msg}})$, the adversary has to fix the public key and messages, which leaves only the signature free to choose. Due to the weak-key-checking, a weak public key will never verify, i.e., we can assume that $\texttt{pk}$ is not a weak key. In particular, even given only one of the random messages (e.g., $\mathsf{H}(\texttt{pk}, \texttt{msg})$), the probability that an adversary finds $\texttt{sig}$ such that $\mathsf{S}.\texttt{Verify}(\texttt{pk}, \mathsf{H}(\texttt{pk}, \texttt{msg}), \texttt{sig}) = 1$ is negligible—which is then also true for the Mal-[S,P]-M advantage.

In the Mal-S-P game, the adversary has to find two different public keys $\texttt{pk} \neq \overline{\texttt{pk}}$, two messages $\texttt{msg}, \overline{\texttt{msg}}$ (not required to differ) and a signature $\texttt{sig}$ such that $\mathsf{S}.\texttt{Verify}(\texttt{pk}, \mathsf{H}(\texttt{pk}, \texttt{msg}), \texttt{sig}) = 1$ and $\mathsf{S}.\texttt{Verify}(\texttt{pk}, \mathsf{H}(\overline{\texttt{pk}}, \overline{\texttt{msg}}), \texttt{sig}) = 1$. Then, again using weak-key-checking, the same reasoning as above applies. □

Remark 16. Note that the assumptions of the above theorem also yield wNR security. More precisely, for a signature scheme that deploys weak-key-checking, its PS-3-transformed version fulfills wNR. In the wNR the adversary is given a public key $\texttt{pk}$ and a signature $\texttt{sig}$ of a randomly chosen message $\texttt{msg}$, which the adversary does *not* receive. The adversary has to find a different public key $\overline{\texttt{pk}}$ and a (potentially different) signature $\overline{\texttt{sig}}$ such that $\texttt{Verify}(\overline{\texttt{pk}}, \mathsf{H}(\overline{\texttt{pk}}, \texttt{msg}), \overline{\texttt{sig}}) = 1$. If the adversary chooses $\overline{\texttt{pk}}$ to be a weak key, verify will never succeed due to the weak-key checking—hence, we can assume that $\overline{\texttt{pk}}$ is not a weak key. Then, the probability that the adversary finds $\overline{\texttt{sig}}$ with $\texttt{Verify}(\overline{\texttt{pk}}, \mathsf{H}(\overline{\texttt{pk}}, \texttt{msg}), \overline{\texttt{sig}}) = 1$ for $\mathsf{H}(\overline{\texttt{pk}}, \texttt{msg})$ random (and unknown), is negligible.

Out of our 15 binding notions, two restrict the adversary two honestly generated public keys: LEAK-[S,P]-M and HON-[S,P]-M. The following corollary states that the PS-3 transform satisfies these notions if the underlying signature scheme is unforgeable. The reason is that unforgeability effectively rules out that the target public key is a weak key.

Corollary 17. *Applying the* PS-3 *transform to an* EUF-CMA *secure signature scheme, yields a scheme that fulfills* LEAK-[S,P]-M *and* HON-[S,P]-M.

Proof. Firstly note, that a signature scheme that fulfills EUF-CMA security can have only negligible many weak keys. Since the key pairs in the notions LEAK-[S,P]-M and HON-[S,P]-M are honestly generated, the probability of the adversary playing against a weak key is negligible. Then the same reasoning as in the proof of Theorem 15 yields security. □

4.3 Analysis of Signature-increasing Transformations

Analysis of the PS-1 *and* PS-2 *Transforms.* Since the PS-1 and the PS-2 transform append the hash of the message $\mathsf{H}(\mathtt{msg})$ or the hash of the public key $\mathsf{H}(\mathtt{pk})$, respectively, to the signature, they achieve all notions for which the adversary has to choose two different messages or public keys, respectively. In particular, this does not depend on the adversarial model. The corresponding theorems are written below.

Theorem 18. *Application of the* PS-1 *transform yields a signature scheme that fulfills* AM-S-M*,* AM-[S,P]-M*, and* AM-S-[M,P] *for* AM $\in \{$HON, LEAK, MAL$\}$.

Theorem 19. *Application of the* PS-2 *transform yields a signature scheme that fulfills* AM-S-P*,* AM-S-[M,P]*, and* AM-[S,M]-P *for* AM $\in \{$HON, LEAK, MAL$\}$.

Analysis of the BUFF-lite *and* BUFF *Transform.* The following theorem shows that the BUFF-lite transform achieves all the new notion in our framework. From this, we can follow that the BUFF transform achieves all these notions as well. Thus, the only difference between the transformations still lies in whether NR is achieved—as was the case for the original BUFF notions.

Theorem 20. *Application of the* BUFF-lite *transformation produces a signature scheme that fulfills all 15 notions shown in Table 1.*

Proof. By [CDF+21, Theorem 5.5], a BUFF-lite transformed signature scheme fulfills M-S-UEO and MBS, i.e., MAL-S-P and MAL-[S,P]-M security in our notation. Using Theorem 4, MAL-S-P security implies that MAL-S-[M,P] and MAL-[S,M]-P hold. Furthermore, Theorem 5 shows that MAL-S-[M,P] together with MAL-[S,P]-M yields MAL-S-M security. Thus, all five of the MAL notions hold after the BUFF-lite transform is applied and hence also all of the corresponding weaker HON and LEAK notions are fulfilled. □

Remark 21. Note that the above proof boils down to the fact that MAL-S-P and MAL-[S,P]-M imply all other notions.

Corollary 22. *Application of the* BUFF *transformation produces a signature scheme that fulfills all 15 notions shown in Table 1.*

4.4 Updated Overview of Transformations

After the analysis conducted in the prior two sections, we can now give an updated overview of the properties achieved by the different transformations. Starting with the PS-3 transform, we gave a new formalization of weak keys. If these keys do not exist for a signature scheme or can be excluded by additional check during verification, we showed that the PS-3 transform suffices to achieve *all* notions from our framework. An overview of the existing and new results regarding the PS-3 transform is given in Fig. 12.

Then, for the signature-increasing transforms, i.e., PS-1, PS-2, BUFF-lite, and BUFF, we also gave an updated analysis: Next to the particular BUFF notions that were already known to be achieved by the transforms, a number of new notions from our framework can be achieved. More precisely, the PS-1 transform achieves all notions Am-B-T for which M $\in$ T, i.e., a total of nine notions. Analogously, the PS-2 transform yields all notions Am-B-T for which P $\in$ T, i.e., again a total of nine notions. This might seem surprising at first glance, as for the original BUFF notions PS-2 achieved more notions (CEO, DEO, M-S-UEO) than PS-1 (DEO, MBS). However, in view of the completed picture for the notions, we see that PS-2 also achieves the previously unknown notion Mal-S-M. Thus, it makes sense that the two transforms symmetrically provide coverage of all notions: PS-1 covers all notions for which the message is bound (as H(`msg`) is appended to the signature) and PS-2 achieves all notions for which the public key is bound (as H(`pk`) is appended to the signature). In particular, the transforms cross over in the three notions where both message *and* public key are bound. Lastly, we showed that the BUFF-lite and the BUFF transform not only yield all of the BUFF notions but also all notions from our new framework. In particular, this implies that no new transforms need to be developed in order to obtain the complete framework. An illustration of the new results regarding the PS-1, PS-2, BUFF-lite, and BUFF transform is given in Fig. 13.

Lastly, when comparing the PS-3 and the BUFF(-lite) transform, we observe that both can be considered for achieving all notions: While the BUFF(-lite) transform can be applied to *any* signature scheme, this comes at the cost of signature expansion; for the PS-3 transform this drawback is not present, however, a scheme-specific analysis of weak keys—according to our new definition—is necessary.

5 Unforgeability in our new Framework

In Section 3, we excluded several notions, arguing that they are unachievable. On of these notions was Hon-P-M. We showed that one can just query two different messages to the signing oracle and thus break the notion. In this section, we consider an adapted variant and discuss its connection to unforgeability. This variant, denoted by Hon-P-M*, agrees with Hon-P-M except for the following: While one of the message signature pairs outputted by the adversary has to stem from a Sign query, the second message is *not* allowed to have been

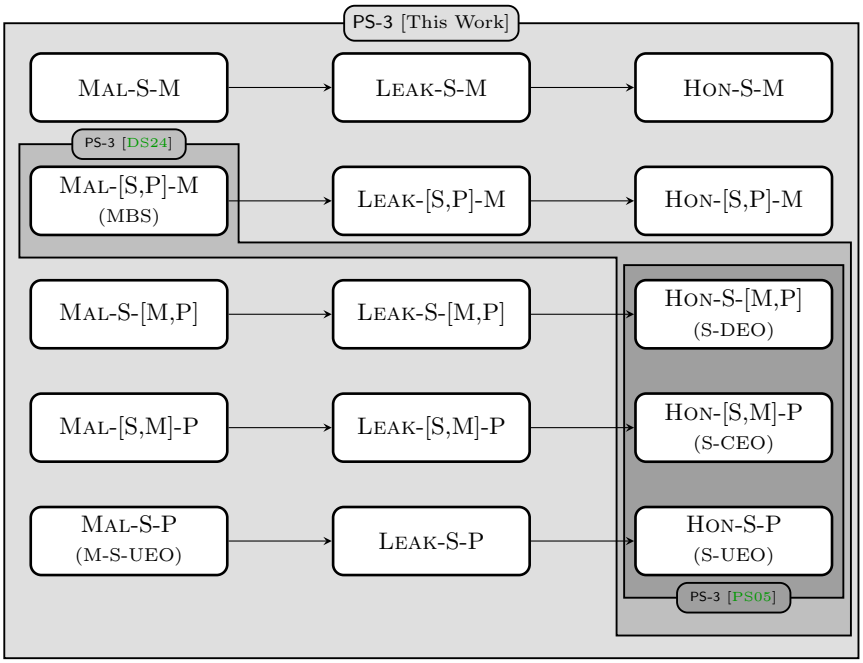

Fig. 12: Existing and new results regarding the PS-3 transform. Note that the result from [DS24] assumes that the *underlying* signature scheme achieves Mal-[S,P]-M security.

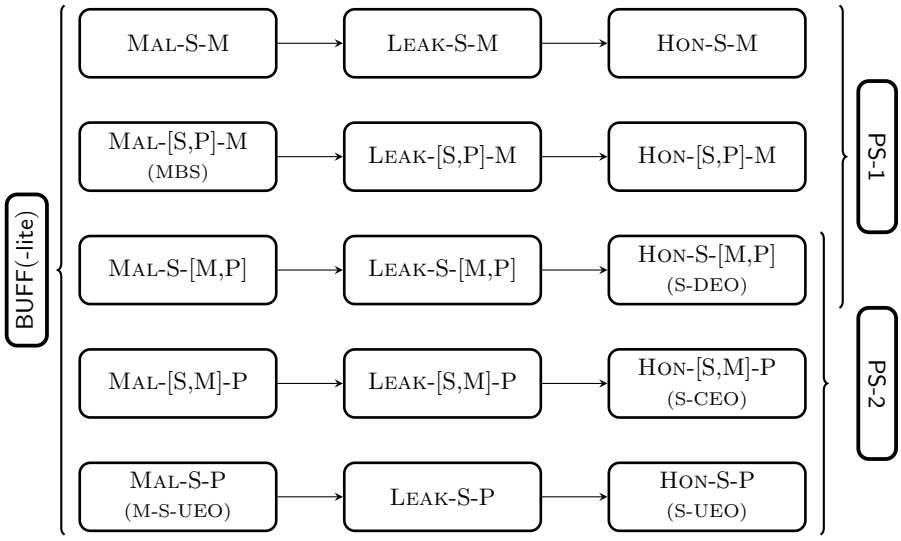

Fig. 13: Our results regarding the transforms PS-1, PS-2, BUFF-lite, and BUFF. The brackets denote that PS-1 and PS-2 achieve the upper 9 and lower 9 notions, respectively, while BUFF(-lite) achieve all 15 notions.

queried. Due to this change, the notion HON-P-M* (shown in Fig. 15) corresponds to EUF-CMA. More generally, in the following an asterisk appended to message/signature denotes that the second message/signature outputted by the adversary is *not* allowed to have been query/response to the Sign oracle.

Analogously to Theorem 4 and Theorem 5, we can also break down HON-P-M* in the two notions HON-[S,P]-M* and HON-P-[S*,M*]. The latter notion is also part of a second relation: HON-P-[S*,M*] and HON-[M,P]-S* are equivalent to HON-P-S*. All of the new notions are described in detail in Fig. 15 and Fig. 16.

Note that—additionally to HON-P-M* agreeing with EUF-CMA—strong unforgeability arises as the combination of HON-P-M* and HON-[M,P]-S*: the former is EUF-CMA as just discussed, while the latter asks to find two signatures for the same message, i.e., exactly the difference between EUF-CMA and SUF-CMA. Further, observe that all of the new notions in Fig. 15 and Fig. 16 did not exist in our framework previously (due to not being achievable)—with the exception of HON-[S,P]-M. Due to the fact that HON-[S,P]-M* is more restricted, HON-[S,P]-M implies HON-[S,P]-M*. However, the converse is not true, as an adversary against HON-[S,P]-M can output a message that was queried to the Sign oracle before, while an adversary against HON-[S,P]-M* cannot. For example, consider a HON-[S,P]-M adversary that makes the following Sign queries:

$$\mathsf{Sign}(\mathtt{sk}, \mathtt{msg}_1) = \mathtt{sig}_1$$
$$\mathsf{Sign}(\mathtt{sk}, \mathtt{msg}_2) = \mathtt{sig}_2$$

If the adversary then notices that $\mathtt{msg}_1$ also verifies for $\mathtt{sig}_2$, i.e.,

$$\mathtt{Verify}(\mathtt{pk}, \mathtt{msg}_1, \mathtt{sig}_2)) = \mathtt{Verify}(\mathtt{pk}, \mathtt{msg}_2, \mathtt{sig}_2)) = 1\,,$$

the adversary can break HON-[S,P]-M by outputting $(\mathtt{sig}_2, \mathtt{msg}_1, \mathtt{msg}_2)$. However, the adversary cannot break HON-[S,P]-M* as both messages have been queried to Sign. An overview of all relations is given in Fig. 14.

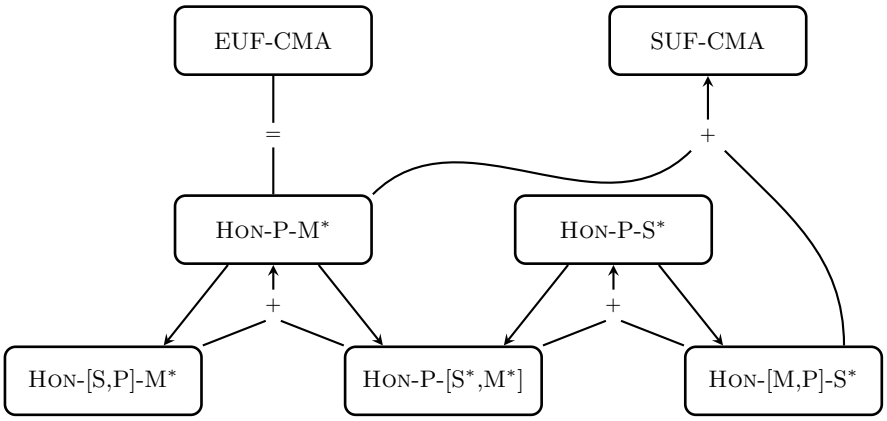

Fig. 14: Implications between the modified notions and their relation to unforgeability.

Game HON-P-M*	Game HON-P-S*
$\mathcal{Q} \leftarrow \emptyset$	$\mathcal{Q} \leftarrow \emptyset$
$(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathtt{S.KeyGen}()$	$(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathtt{S.KeyGen}()$
$(\mathtt{sig}, \overline{\mathtt{sig}}, \mathtt{msg}, \overline{\mathtt{msg}}) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathtt{sk},\cdot)}(\mathtt{pk})$	$(\mathtt{sig}, \overline{\mathtt{sig}}, \mathtt{msg}, \overline{\mathtt{msg}}) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathtt{sk},\cdot)}(\mathtt{pk})$
if $(\mathtt{msg}, \mathtt{sig}) \notin \mathcal{Q}$	**if** $(\mathtt{msg}, \mathtt{sig}) \notin \mathcal{Q}$
return 0	**return** 0
if $(\overline{\mathtt{msg}}, \cdot) \in \mathcal{Q}$	**if** $(\cdot, \overline{\mathtt{sig}}) \in \mathcal{Q}$
return 0	**return** 0
if $\mathtt{msg} = \overline{\mathtt{msg}}$	**if** $\mathtt{sig} = \overline{\mathtt{sig}}$
return 0	**return** 0
$\mathtt{v}_0 = \mathtt{S.Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig})$	$\mathtt{v}_0 = \mathtt{S.Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig})$
$\mathtt{v}_1 = \mathtt{S.Verify}(\mathtt{pk}, \overline{\mathtt{msg}}, \overline{\mathtt{sig}})$	$\mathtt{v}_1 = \mathtt{S.Verify}(\mathtt{pk}, \overline{\mathtt{msg}}, \overline{\mathtt{sig}})$
return $[\![\mathtt{v}_0 = 1 \wedge \mathtt{v}_1 = 1]\!]$	**return** $[\![\mathtt{v}_0 = 1 \wedge \mathtt{v}_1 = 1]\!]$

Fig. 15: Modified (generalized) notions related to unforgeability.

Game HON-[S,P]-M*

$\mathcal{Q} \leftarrow \emptyset$
$(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathtt{S.KeyGen}()$
$(\mathtt{sig}, \mathtt{msg}, \overline{\mathtt{msg}}) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathtt{sk},\cdot)}(\mathtt{pk})$
if $(\mathtt{msg}, \mathtt{sig}) \notin \mathcal{Q}$
  return 0
if $(\overline{\mathtt{msg}}, \cdot) \in \mathcal{Q}$
  return 0
if $\mathtt{msg} = \overline{\mathtt{msg}}$
  return 0
$\mathtt{v}_0 = \mathtt{S.Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig})$
$\mathtt{v}_1 = \mathtt{S.Verify}(\mathtt{pk}, \overline{\mathtt{msg}}, \mathtt{sig})$
return $[\![\mathtt{v}_0 = 1 \wedge \mathtt{v}_1 = 1]\!]$

Game HON-[M,P]-S*

$\mathcal{Q} \leftarrow \emptyset$
$(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathtt{S.KeyGen}()$
$(\mathtt{sig}, \overline{\mathtt{sig}}, \mathtt{msg}) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathtt{sk},\cdot)}(\mathtt{pk})$
if $(\mathtt{msg}, \mathtt{sig}) \notin \mathcal{Q}$
  return 0
if $(\cdot, \overline{\mathtt{sig}}) \in \mathcal{Q}$
  return 0
if $\mathtt{sig} = \overline{\mathtt{sig}}$
  return 0
$\mathtt{v}_0 = \mathtt{S.Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig})$
$\mathtt{v}_1 = \mathtt{S.Verify}(\mathtt{pk}, \overline{\mathtt{msg}}, \overline{\mathtt{sig}})$
return $[\![\mathtt{v}_0 = 1 \wedge \mathtt{v}_1 = 1]\!]$

Game HON-P-[S*,M*]

$\mathcal{Q} \leftarrow \emptyset$
$(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathtt{S.KeyGen}()$
$(\mathtt{sig}, \overline{\mathtt{sig}}, \mathtt{msg}, \overline{\mathtt{msg}}) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathtt{sk},\cdot)}(\mathtt{pk})$
if $(\mathtt{msg}, \mathtt{sig}) \notin \mathcal{Q}$
  return 0
if $(\overline{\mathtt{msg}}, \cdot) \in \mathcal{Q}$
  return 0
if $(\cdot, \overline{\mathtt{sig}}) \in \mathcal{Q}$
  return 0
if $\mathtt{msg} = \overline{\mathtt{msg}}$
  return 0
if $\mathtt{sig} = \overline{\mathtt{sig}}$
  return 0
$\mathtt{v}_0 = \mathtt{S.Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig})$
$\mathtt{v}_1 = \mathtt{S.Verify}(\mathtt{pk}, \overline{\mathtt{msg}}, \overline{\mathtt{sig}})$
return $[\![\mathtt{v}_0 = 1 \wedge \mathtt{v}_1 = 1]\!]$

Oracle $\mathsf{Sign}(\mathtt{sk}, \mathtt{msg})$

$\mathtt{sig} \leftarrow \mathtt{S.Sign}(\mathtt{sk}, \mathtt{msg})$
$\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(\mathtt{msg}, \mathtt{sig})\}$
return $\mathtt{sig}$

Fig. 16: Modified (restricted) notions related to unforgeability.

References

ADG+22. Ange Albertini, Thai Duong, Shay Gueron, Stefan Kölbl, Atul Luykx, and Sophie Schmieg. How to abuse and fix authenticated encryption without key commitment. In Kevin R. B. Butler and Kurt Thomas, editors, *USENIX Security 2022*, pages 3291–3308. USENIX Association, August 2022. (Cited on page 1.)

ADM+24. Thomas Aulbach, Samed Düzlü, Michael Meyer, Patrick Struck, and Maximiliane Weishäupl. Hash your keys before signing - BUFF security of the additional NIST PQC signatures. In Markku-Juhani Saarinen and Daniel Smith-Tone, editors, *Post-Quantum Cryptography - 15th International Workshop, PQCrypto 2024, Part II*, pages 301–335. Springer, Cham, June 2024. (Cited on pages 5, 13, and 33.)

Aye15. Andrew Ayer. Duplicate signature key selection attack in Let's Encrypt. https://www.agwa.name/blog/post/duplicate_signature_key_selection_attack_in_lets_encrypt, 2015. (Cited on page 1.)

BCD+23. Ward Beullens, Ming-Shing Chen, Jintai Ding, Boru Gong, Matthias J. Kannwischer, Jacques Patarin, Bo-Yuan Peng, Dieter Schmidt, Cheng-Jhih Shih, Chengdong Tao, and Bo-Yin Yang. UOV — Unbalanced Oil and Vinegar. Technical report, National Institute of Standards and Technology, 2023. available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures. (Cited on page 17.)

BCD+24. Ward Beullens, Ming-Shing Chen, Jintai Ding, Boru Gong, Matthias J. Kannwischer, Jacques Patarin, Bo-Yuan Peng, Dieter Schmidt, Cheng-Jhih Shih, Chengdong Tao, and Bo-Yin Yang. UOV — Unbalanced Oil and Vinegar. Technical report, National Institute of Standards and Technology, 2024. available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-2-additional-signatures. (Cited on pages 19 and 33.)

BJKS24. Karthikeyan Bhargavan, Charlie Jacomme, Franziskus Kiefer, and Rolfe Schmidt. Formal verification of the PQXDH post-quantum key agreement protocol for end-to-end secure messaging. In Davide Balzarotti and Wenyuan Xu, editors, *USENIX Security 2024*. USENIX Association, August 2024. (Cited on page 1.)

BWM99. Simon Blake-Wilson and Alfred Menezes. Unknown key-share attacks on the station-to-station (STS) protocol. In Hideki Imai and Yuliang Zheng, editors, *PKC'99*, volume 1560 of *LNCS*, pages 154–170. Springer, Berlin, Heidelberg, March 1999. (Cited on page 5.)

CDF+21. Cas Cremers, Samed Düzlü, Rune Fiedler, Marc Fischlin, and Christian Janson. BUFFing signature schemes beyond unforgeability and the case of post-quantum signatures. In *2021 IEEE Symposium on Security and Privacy*, pages 1696–1714. IEEE Computer Society Press, May 2021. (Cited on pages 4, 5, 10, 13, 14, 15, 16, 17, and 23.)

CDM24. Cas Cremers, Alexander Dax, and Niklas Medinger. Keeping up with the KEMs: Stronger security notions for KEMs and automated analysis of KEM-based protocols. In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, *ACM CCS 2024*, pages 1046–1060. ACM Press, October 2024. (Cited on pages 2, 3, 6, and 7.)

DFF24. Samed Düzlü, Rune Fiedler, and Marc Fischlin. BUFFing FALCON without increasing the signature size. In *SAC 2024*, 2024. (Cited on page 5.)

DFH+24. Jelle Don, Serge Fehr, Yu-Hsuan Huang, Jyun-Jie Liao, and Patrick Struck. Hide-and-seek and the non-resignability of the BUFF transform. In Elette Boyle and Mohammad Mahmoody, editors, *TCC 2024, Part III*, volume 15366 of *LNCS*, pages 347–370. Springer, Cham, December 2024. (Cited on page 5.)

DFHS24. Jelle Don, Serge Fehr, Yu-Hsuan Huang, and Patrick Struck. On the (in)security of the BUFF transform. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part I*, volume 14920 of *LNCS*, pages 246–275. Springer, Cham, August 2024. (Cited on pages 5 and 17.)

DGRW18. Yevgeniy Dodis, Paul Grubbs, Thomas Ristenpart, and Joanne Woodage. Fast message franking: From invisible salamanders to encryptment. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 155–186. Springer, Cham, August 2018. (Cited on page 1.)

DS24. Samed Düzlü and Patrick Struck. The role of message-bound signatures for the beyond UnForgeability features and weak keys. In Nicky Mouha and Nick Nikiforakis, editors, *ISC 2024, Part II*, volume 15258 of *LNCS*, pages 61–80. Springer, Cham, October 2024. (Cited on pages 4, 5, 17, 20, and 25.)

Emu24. Keita Emura. On the BUFF security of ECDSA with key recovery. Cryptology ePrint Archive, Report 2024/2018, 2024. (Cited on page 5.)

FMT25. Marc Fischlin, Aikaterini Mitrokotsa, and Jenit Tomy. BUFFing threshold signature schemes. In *PKC 2025*, 2025. (Cited on page 5.)

JCCS19. Dennis Jackson, Cas Cremers, Katriel Cohn-Gordon, and Ralf Sasse. Seems legit: Automated analysis of subtle attacks on protocols that use signatures. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2165–2180. ACM Press, November 2019. (Cited on pages 1 and 5.)

KSW24a. Juliane Krämer, Patrick Struck, and Maximiliane Weishäupl. Binding security of implicitly-rejecting KEMs and application to BIKE and HQC. Cryptology ePrint Archive, Report 2024/1233, 2024. (Cited on page 3.)

KSW24b. Juliane Krämer, Patrick Struck, and Maximiliane Weishäupl. Committing AE from sponges: Security analysis of the NIST LWC finalists. *IACR Trans. Symm. Cryptol.*, 2024(4):191–248, 2024. (Cited on page 3.)

KX24. Mukul Kulkarni and Keita Xagawa. Strong existential unforgeability and more of MPC-in-the-head signatures. Cryptology ePrint Archive, Report 2024/1069, 2024. (Cited on page 5.)

LGR21. Julia Len, Paul Grubbs, and Thomas Ristenpart. Partitioning oracle attacks. In Michael Bailey and Rachel Greenstadt, editors, *USENIX Security 2021*, pages 195–212. USENIX Association, August 2021. (Cited on page 1.)

Lyu12. Vadim Lyubashevsky. Lattice signatures without trapdoors. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 738–755. Springer, Berlin, Heidelberg, April 2012. (Cited on pages 19 and 22.)

MLGR23. Sanketh Menda, Julia Len, Paul Grubbs, and Thomas Ristenpart. Context discovery and commitment attacks - how to break CCM, EAX, SIV, and more. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part IV*, volume 14007 of *LNCS*, pages 379–407. Springer, Cham, April 2023. (Cited on pages 2 and 3.)

MS04. Alfred Menezes and Nigel P. Smart. Security of signature schemes in a multi-user setting. *DCC*, 33(3):261–274, 2004. (Cited on page 5.)

NIST15. National Institute of Standards and Technology. Lightweight cryptography standardization process. https://csrc.nist.gov/projects/lightweight-cryptography, 2015. (Cited on page 3.)
NIST17. National Institute of Standards and Technology. Post-quantum cryptography standardization process. https://csrc.nist.gov/projects/post-quantum-cryptography, 2017. (Cited on page 5.)
NIST22. National Institute of Standards and Technology. Call for additional digital signature schemes for the post-quantum cryptography standardization process. https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/call-for-proposals-dig-sig-sept-2022.pdf, 2022. (Cited on page 5.)
NSS23. Yusuke Naito, Yu Sasaki, and Takeshi Sugawara. Committing security of Ascon: Cryptanalysis on primitive and proof on mode. *IACR Trans. Symm. Cryptol.*, 2023(4):420–451, 2023. (Cited on page 3.)
PS05. Thomas Pornin and Julien P. Stern. Digital signatures do not guarantee exclusive ownership. In John Ioannidis, Angelos Keromytis, and Moti Yung, editors, *ACNS 05International Conference on Applied Cryptography and Network Security*, volume 3531 of *LNCS*, pages 138–150. Springer, Berlin, Heidelberg, June 2005. (Cited on pages 4, 5, 7, 14, 15, 16, 18, 21, and 25.)
Sch91. Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, January 1991. (Cited on page 19.)
SPMS02. Jacques Stern, David Pointcheval, John Malone-Lee, and Nigel P. Smart. Flaws in applying proof methodologies to signature schemes. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 93–110. Springer, Berlin, Heidelberg, August 2002. (Cited on page 5.)

A Proofs

A.1 Proof of Theorem 9

Proof. Let $\mathtt{S}$ be a signature scheme that is both Am-S-[M,P] and Am-[S,M]-P. Let further $\mathtt{S}^*$ be the signature scheme displayed in Fig. 17.

To break Am-[S,M]-P, consider an adversary that receives an honestly generated public key $\mathtt{pk}^* = (\mathtt{pk}, 1)$, queries $\mathtt{msg}^*$ to its signing oracle, and receives a signature $\mathtt{sig}^* = (\mathtt{sig}, 0)$. By outputting $\overline{\mathtt{pk}}^* = (\mathtt{pk}, 0)$, $\mathtt{Verify}^*(\mathtt{pk}^*, \mathtt{msg}^*, \mathtt{sig}^*)$ will output 1 via the normal verification check while $\mathtt{Verify}^*(\overline{\mathtt{pk}}^*, \mathtt{msg}^*, \mathtt{sig}^*)$ will output 1 via the special verification check; therefore this adversary breaks Am-[S,M]-P for $\mathrm{Am} \in \{\mathrm{Mal}, \mathrm{Leak}, \mathrm{Hon}\}$.

Regarding Am-S-[M,P], note that any public key with a trailing 0 can only verify exactly one message, that is, $\mathtt{msg}^*$. Any Am-[S,M]-P adversary therefore needs to output a public key having a trailing 1 in which case the signature scheme is equal to the underlying signature scheme, thus inheriting its Am-[S,M]-P security for $\mathrm{Am} \in \{\mathrm{Mal}, \mathrm{Leak}, \mathrm{Hon}\}$. □

A.2 Proof of Theorem 10

Proof. Let $\mathtt{S}$ be a signature scheme that is both Am-[S,M]-P and Am-S-[M,P]. Let further $\mathtt{S}^*$ be the signature scheme displayed in Fig. 18.

$\underline{\mathtt{KeyGen}^*()}$

$(\mathtt{pk}, \mathtt{sk}) \leftarrow\!\!\$\ \mathtt{KeyGen}()$
$\mathtt{pk}^* \leftarrow (\mathtt{pk}, 1)$
return $(\mathtt{pk}^*, \mathtt{sk})$

$\underline{\mathtt{Sign}^*(\mathtt{sk}, \mathtt{msg})}$

$\mathtt{sig} \leftarrow\!\!\$\ \mathtt{Sign}(\mathtt{sk}, \mathtt{msg})$
if $\mathtt{msg} = \mathtt{msg}_*$
  $\mathtt{sig}^* \leftarrow (\mathtt{sig}, 0)$
else
  $\mathtt{sig}^* \leftarrow (\mathtt{sig}, 1)$
return $\mathtt{sig}^*$

$\underline{\mathtt{Verify}^*(\mathtt{pk}^*, \mathtt{msg}, \mathtt{sig}^*)}$

$(\mathtt{pk}, b) \leftarrow \mathtt{pk}^*$
$(\mathtt{sig}, d) \leftarrow \mathtt{sig}^*$
if $b = 0$ // Dishonest key
  return $[\![d = 0 \wedge \mathtt{msg} = \mathtt{msg}_*]\!]$
if $b = 1$
  return $\mathtt{Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig})$

Fig. 17: Separation example for Am-S-[M,P] $\not\Rightarrow$ Am-[S,M]-P (Theorem 9).

$\underline{\mathtt{KeyGen}^*()}$

$(\mathtt{pk}, \mathtt{sk}) \leftarrow\!\!\$\ \mathtt{KeyGen}()$
$\mathtt{pk}^* \leftarrow (\mathtt{pk}, 1)$
return $(\mathtt{pk}^*, \mathtt{sk})$

$\underline{\mathtt{Sign}^*(\mathtt{sk}, \mathtt{msg})}$

$\mathtt{sig} \leftarrow\!\!\$\ \mathtt{Sign}(\mathtt{sk}, \mathtt{msg})$
if $\mathtt{msg} = \mathtt{msg}_*$
  $\mathtt{sig}^* \leftarrow (\mathtt{sig}, 0)$
else
  $\mathtt{sig}^* \leftarrow (\mathtt{sig}, 1)$
return $\mathtt{sig}^*$

$\underline{\mathtt{Verify}^*(\mathtt{pk}^*, \mathtt{msg}, \mathtt{sig}^*)}$

$(\mathtt{pk}, b) \leftarrow \mathtt{pk}^*$
$(\mathtt{sig}, d) \leftarrow \mathtt{sig}^*$
if $b = 0 \wedge d = 0$
  return $[\![\mathtt{msg} = \mathtt{msg}_{**}]\!]$
if $b = 0 \wedge d = 1$
  return 0
if $b = 1 \wedge d = 0$
  return $[\![\mathtt{msg} = \mathtt{msg}_* \wedge \mathtt{Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig})]\!]$
if $b = 1 \wedge d = 1$
  return $\mathtt{Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig})$

Fig. 18: Separation example for Am-[S,M]-P $\not\Rightarrow$ Am-S-[M,P] (Theorem 10).

To break Am-S-[M,P], consider an adversary that receives an honestly generated public key $\mathtt{pk}^* = (\mathtt{pk}, 1)$, queries $\mathtt{msg}^*$ to its signing oracle, and receives a signature $\mathtt{sig}^* = (\mathtt{sig}, 0)$. By outputting $(\mathtt{sig}^* = (\mathtt{sig}, 0), \mathtt{msg}^*, \mathtt{msg}^{**}, \overline{\mathtt{pk}}^* = (\mathtt{pk}, 0))$, $\mathtt{Verify}^*(\mathtt{pk}^*, \mathtt{msg}^*, \mathtt{sig}^*)$ will output 1 via the special verification check (third if condition) while $\mathtt{Verify}^*(\overline{\mathtt{pk}}^*, \mathtt{msg}^{**}, \mathtt{sig}^*)$ will output 1 via the other special verification check (first if condition); thus this adversary breaks Am-S-[M,P] for $\mathrm{Am} \in \{\mathrm{Mal}, \mathrm{Leak}, \mathrm{Hon}\}$.

Regarding Am-[S,M]-P, observe that signatures with a trailing 0 can verify exactly two messages ($\mathtt{msg}^*$ and $\mathtt{msg}^{**}$) but only under distinct public keys (verification of $\mathtt{msg}^*$ and $\mathtt{msg}^{**}$ requires a public key with a trailing 1 and 0, respectively). Thus, any successful Am-[S,M]-P needs to output a signature with a trailing 1. However, in this case, the transformed signature scheme is equivalent to the underlying signature scheme and thus inherits its Am-[S,M]-P security.

□

$\mathtt{KeyGen}^*()$	$\mathtt{Sign}^*(\mathtt{sk},\mathtt{msg})$	$\mathtt{Verify}^*(\mathtt{pk},\mathtt{msg},\mathtt{sig}^*)$
$(\mathtt{pk},\mathtt{sk}) \leftarrow\!\!\$\ \mathtt{KeyGen}()$	$\mathtt{sig} \leftarrow\!\!\$\ \mathtt{Sign}(\mathtt{sk},\mathtt{msg})$	$(\mathtt{sig}, z) \leftarrow \mathtt{sig}^*$
return $(\mathtt{pk},\mathtt{sk})$	$\mathtt{sig}^* \leftarrow (\mathtt{sig},\mathtt{pk})$	**if** $z \neq \bot$
	return $\mathtt{sig}^*$	**return** $[\![z = \mathtt{pk}]\!]$
		if $z = \bot$
		return $\mathtt{Verify}(\mathtt{pk},\mathtt{msg},\mathtt{sig})$

Fig. 19: Separation example for AM-S-[M,P] $\not\Rightarrow$ AM-[S,P]-M (Theorem 11).

A.3 Proof of Theorem 11

Note that the separation example for this proof is a signature scheme that is not unforgeable. This is to be expected as our separations apply to all attack models and the notion HON-[S,P]-M, i.e. AM-[S,P]-M in the HON setting, is related to unforgeability. We elaborate further on this connection in Section 5.

Proof. Let $\mathtt{S}$ be a signature scheme that is both AM-[S,P]-M and AM-S-[M,P]. Let further $\mathtt{S}^*$ be the signature scheme displayed in Fig. 19.

Firstly, we show that the signature scheme $\mathtt{S}^*$ is not AM-[S,P]-M. An adversary $\mathcal{A}$, receiving a public key $\mathtt{pk}$, can simply ask for any signature and receive $\mathtt{sig}^* = (\mathtt{sig},\mathtt{pk})$. By construction, $\mathtt{sig}^*$ verifies under $\mathtt{pk}$ for *any* message, i.e., $\mathcal{A}$ can simply output $(\mathtt{sig},\mathtt{msg},\overline{\mathtt{msg}},\mathtt{pk})$.

To show that $\mathtt{S}^*$ is AM-S-[M,P], observe that signatures $(\mathtt{sig}, z)$ for $z \neq \bot$ verify *only* under public key z—in which case they verify any message though. This makes it impossible to break AM-S-[M,P] via this special verification step, as $\mathcal{A}$ needs to provide a signature and two *distinct* public keys that verify this signature. For signatures $(\mathtt{sig}, z)$ with $z = \bot$, $\mathtt{S}^*$ is equal to the underlying signature scheme and thus maintains its AM-S-[M,P] security. □

A.4 Proof of Theorem 12

Proof. The result follows from [ADM+24], which shows that UOV [BCD+24] achieves MBS, i.e., it is AM-[S,P]-M for $\mathrm{AM} \in \{\mathrm{MAL}, \mathrm{LEAK}, \mathrm{HON}\}$, but not S-DEO, i.e., it is not AM-S-[M,P] for $\mathrm{AM} \in \{\mathrm{MAL}, \mathrm{LEAK}, \mathrm{HON}\}$. □

B Explicit Security Notions

In this section, we give the individual security games for all 15 notions: AM-S-M in Fig. 20, AM-[S,P]-M in Fig. 21, AM-S-[M,P] in Fig. 22, AM-[S,M]-P in Fig. 23, and lastly AM-S-P in Fig. 24.

Game MAL-S-M

$(\mathtt{sig}, \mathtt{msg}, \overline{\mathtt{msg}}, \mathtt{pk}, \overline{\mathtt{pk}}) \leftarrow \mathcal{A}()$
if $\mathtt{msg} = \overline{\mathtt{msg}}$
  return 0
$\mathtt{v}_0 = \mathtt{S.Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig})$
$\mathtt{v}_1 = \mathtt{S.Verify}(\overline{\mathtt{pk}}, \overline{\mathtt{msg}}, \mathtt{sig})$
return $[\![\mathtt{v}_0 = 1 \wedge \mathtt{v}_1 = 1]\!]$

Game LEAK-S-M

$(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathtt{S.KeyGen}()$
$(\mathtt{sig}, \mathtt{msg}, \overline{\mathtt{msg}}, \overline{\mathtt{pk}}) \leftarrow \mathcal{A}(\mathtt{pk}, \mathtt{sk})$
if $\mathtt{msg} = \overline{\mathtt{msg}}$
  return 0
$\mathtt{v}_0 = \mathtt{S.Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig})$
$\mathtt{v}_1 = \mathtt{S.Verify}(\overline{\mathtt{pk}}, \overline{\mathtt{msg}}, \mathtt{sig})$
return $[\![\mathtt{v}_0 = 1 \wedge \mathtt{v}_1 = 1]\!]$

Game HON-S-M

$\mathcal{Q} \leftarrow \emptyset$
$(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathtt{S.KeyGen}()$
$(\mathtt{sig}, \mathtt{msg}, \overline{\mathtt{msg}}, \overline{\mathtt{pk}}) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathtt{sk}, \cdot)}(\mathtt{pk})$
if $(\mathtt{msg}, \mathtt{sig}) \notin \mathcal{Q}$
  return 0
if $\mathtt{msg} = \overline{\mathtt{msg}}$
  return 0
$\mathtt{v}_0 = \mathtt{S.Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig})$
$\mathtt{v}_1 = \mathtt{S.Verify}(\overline{\mathtt{pk}}, \overline{\mathtt{msg}}, \mathtt{sig})$
return $[\![\mathtt{v}_0 = 1 \wedge \mathtt{v}_1 = 1]\!]$

Oracle $\mathsf{Sign}(\mathtt{sk}, \mathtt{msg})$

$\mathtt{sig} \leftarrow \mathtt{S.Sign}(\mathtt{sk}, \mathtt{msg})$
$\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(\mathtt{msg}, \mathtt{sig})\}$
return sig

Fig. 20: The notions MAL-S-M, LEAK-S-M, and HON-S-M.

Game MAL-[S,P]-M

$(\mathtt{sig}, \mathtt{msg}, \overline{\mathtt{msg}}, \mathtt{pk}) \leftarrow \mathcal{A}()$
if $\mathtt{msg} = \overline{\mathtt{msg}}$
  return 0
$\mathtt{v}_0 = \mathtt{S.Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig})$
$\mathtt{v}_1 = \mathtt{S.Verify}(\mathtt{pk}, \overline{\mathtt{msg}}, \mathtt{sig})$
return $[\![\mathtt{v}_0 = 1 \wedge \mathtt{v}_1 = 1]\!]$

Game LEAK-[S,P]-M

$(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathtt{S.KeyGen}()$
$(\mathtt{sig}, \mathtt{msg}, \overline{\mathtt{msg}}) \leftarrow \mathcal{A}(\mathtt{pk}, \mathtt{sk})$
if $\mathtt{msg} = \overline{\mathtt{msg}}$
  return 0
$\mathtt{v}_0 = \mathtt{S.Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig})$
$\mathtt{v}_1 = \mathtt{S.Verify}(\mathtt{pk}, \overline{\mathtt{msg}}, \mathtt{sig})$
return $[\![\mathtt{v}_0 = 1 \wedge \mathtt{v}_1 = 1]\!]$

Game HON-[S,P]-M

$\mathcal{Q} \leftarrow \emptyset$
$(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathtt{S.KeyGen}()$
$(\mathtt{sig}, \mathtt{msg}, \overline{\mathtt{msg}}) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathtt{sk}, \cdot)}(\mathtt{pk})$
if $(\mathtt{msg}, \mathtt{sig}) \notin \mathcal{Q}$
  return 0
if $\mathtt{msg} = \overline{\mathtt{msg}}$
  return 0
$\mathtt{v}_0 = \mathtt{S.Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig})$
$\mathtt{v}_1 = \mathtt{S.Verify}(\mathtt{pk}, \overline{\mathtt{msg}}, \mathtt{sig})$
return $[\![\mathtt{v}_0 = 1 \wedge \mathtt{v}_1 = 1]\!]$

Oracle $\mathsf{Sign}(\mathtt{sk}, \mathtt{msg})$

$\mathtt{sig} \leftarrow \mathtt{S.Sign}(\mathtt{sk}, \mathtt{msg})$
$\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(\mathtt{msg}, \mathtt{sig})\}$
return sig

Fig. 21: The notions MAL-[S,P]-M, LEAK-[S,P]-M, and HON-[S,P]-M.

Game MAL-S-[M,P]

$(\mathtt{sig}, \mathtt{msg}, \overline{\mathtt{msg}}, \mathtt{pk}, \overline{\mathtt{pk}}) \leftarrow \mathcal{A}()$
if $\mathtt{msg} = \overline{\mathtt{msg}} \vee \mathtt{pk} = \overline{\mathtt{pk}}$
  return 0
$\mathtt{v}_0 = \mathtt{S.Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig})$
$\mathtt{v}_1 = \mathtt{S.Verify}(\overline{\mathtt{pk}}, \overline{\mathtt{msg}}, \mathtt{sig})$
return $[\![\mathtt{v}_0 = 1 \wedge \mathtt{v}_1 = 1]\!]$

Game LEAK-S-[M,P]

$(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathtt{S.KeyGen}()$
$(\mathtt{sig}, \mathtt{msg}, \overline{\mathtt{msg}}, \overline{\mathtt{pk}}) \leftarrow \mathcal{A}(\mathtt{pk}, \mathtt{sk})$
if $\mathtt{msg} = \overline{\mathtt{msg}} \vee \mathtt{pk} = \overline{\mathtt{pk}}$
  return 0
$\mathtt{v}_0 = \mathtt{S.Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig})$
$\mathtt{v}_1 = \mathtt{S.Verify}(\overline{\mathtt{pk}}, \overline{\mathtt{msg}}, \mathtt{sig})$
return $[\![\mathtt{v}_0 = 1 \wedge \mathtt{v}_1 = 1]\!]$

Game HON-S-[M,P]

$\mathcal{Q} \leftarrow \emptyset$
$(\mathtt{pk}, \mathtt{sk}) \leftarrow\!\!\$\ \mathtt{S.KeyGen}()$
$(\mathtt{sig}, \mathtt{msg}, \overline{\mathtt{msg}}, \overline{\mathtt{pk}}) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathtt{sk}, \cdot)}(\mathtt{pk})$
if $(\mathtt{msg}, \mathtt{sig}) \notin \mathcal{Q}$
  return 0
if $\overline{\mathtt{msg}} = \mathtt{msg} \vee \overline{\mathtt{pk}} = \mathtt{pk}$
  return 0
$\mathtt{v}_0 \leftarrow \mathtt{S.Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig})$
$\mathtt{v}_1 \leftarrow \mathtt{S.Verify}(\overline{\mathtt{pk}}, \overline{\mathtt{msg}}, \mathtt{sig})$
return $[\![\mathtt{v}_0 = 1 \wedge \mathtt{v}_1 = 1]\!]$

Oracle $\mathsf{Sign}(\mathtt{sk}, \mathtt{msg})$

$\mathtt{sig} \leftarrow \mathtt{S.Sign}(\mathtt{sk}, \mathtt{msg})$
$\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(\mathtt{msg}, \mathtt{sig})\}$
return sig

Fig. 22: The notions MAL-S-[M,P], LEAK-S-[M,P], and HON-S-[M,P].

Game MAL-[S,M]-P

$(\mathtt{sig}, \mathtt{msg}, \mathtt{pk}, \overline{\mathtt{pk}}) \leftarrow \mathcal{A}()$
if $\mathtt{pk} = \overline{\mathtt{pk}}$
  return 0
$\mathtt{v}_0 = \mathtt{S.Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig})$
$\mathtt{v}_1 = \mathtt{S.Verify}(\overline{\mathtt{pk}}, \mathtt{msg}, \mathtt{sig})$
return $[\![\mathtt{v}_0 = 1 \wedge \mathtt{v}_1 = 1]\!]$

Game LEAK-[S,M]-P

$(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathtt{S.KeyGen}()$
$(\mathtt{sig}, \mathtt{msg}, \overline{\mathtt{pk}}) \leftarrow \mathcal{A}(\mathtt{pk}, \mathtt{sk})$
if $\mathtt{pk} = \overline{\mathtt{pk}}$
  return 0
$\mathtt{v}_0 = \mathtt{S.Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig})$
$\mathtt{v}_1 = \mathtt{S.Verify}(\overline{\mathtt{pk}}, \mathtt{msg}, \mathtt{sig})$
return $[\![\mathtt{v}_0 = 1 \wedge \mathtt{v}_1 = 1]\!]$

Game HON-[S,M]-P

$\mathcal{Q} \leftarrow \emptyset$
$(\mathtt{pk}, \mathtt{sk}) \leftarrow\!\!\$\ \mathtt{S.KeyGen}()$
$(\mathtt{sig}, \mathtt{msg}, \overline{\mathtt{pk}}) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathtt{sk}, \cdot)}(\mathtt{pk})$
if $(\mathtt{msg}, \mathtt{sig}) \notin \mathcal{Q}$
  return 0
if $\overline{\mathtt{pk}} = \mathtt{pk}$
  return 0
$\mathtt{v}_1 \leftarrow \mathtt{S.Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig})$
$\mathtt{v}_2 \leftarrow \mathtt{S.Verify}(\overline{\mathtt{pk}}, \mathtt{msg}, \mathtt{sig})$
return $[\![\mathtt{v}_1 = 1 \wedge \mathtt{v}_2 = 1]\!]$

Oracle $\mathsf{Sign}(\mathtt{sk}, \mathtt{msg})$

$\mathtt{sig} \leftarrow \mathtt{S.Sign}(\mathtt{sk}, \mathtt{msg})$
$\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(\mathtt{msg}, \mathtt{sig})\}$
return sig

Fig. 23: The notions MAL-[S,M]-P, LEAK-[S,M]-P, and HON-[S,M]-P.

Game MAL-S-P

$(\mathtt{sig}, \mathtt{msg}, \overline{\mathtt{msg}}, \mathtt{pk}, \overline{\mathtt{pk}}) \leftarrow \mathcal{A}()$
if $\mathtt{pk} \neq \overline{\mathtt{pk}}$
  return 0
$\mathtt{v}_0 = \mathtt{S.Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig})$
$\mathtt{v}_1 = \mathtt{S.Verify}(\overline{\mathtt{pk}}, \overline{\mathtt{msg}}, \mathtt{sig})$
return $[\![\mathtt{v}_0 = 1 \wedge \mathtt{v}_1 = 1]\!]$

Game LEAK-S-P

$(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathtt{S.KeyGen}()$
$(\mathtt{sig}, \mathtt{msg}, \overline{\mathtt{msg}}, \overline{\mathtt{pk}}) \leftarrow \mathcal{A}(\mathtt{pk}, \mathtt{sk})$
if $\mathtt{pk} \neq \overline{\mathtt{pk}}$
  return 0
$\mathtt{v}_0 = \mathtt{S.Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig})$
$\mathtt{v}_1 = \mathtt{S.Verify}(\overline{\mathtt{pk}}, \overline{\mathtt{msg}}, \mathtt{sig})$
return $[\![\mathtt{v}_0 = 1 \wedge \mathtt{v}_1 = 1]\!]$

Game HON-S-P

$\mathcal{Q} \leftarrow \emptyset$
$(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathtt{S.KeyGen}()$
$(\mathtt{sig}, \mathtt{msg}, \overline{\mathtt{msg}}, \overline{\mathtt{pk}}) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathtt{sk}, \cdot)}(\mathtt{pk})$
if $(\mathtt{msg}, \mathtt{sig}) \notin \mathcal{Q}$
  return 0
if $\mathtt{pk} \neq \overline{\mathtt{pk}}$
  return 0
$\mathtt{v}_0 = \mathtt{S.Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig})$
$\mathtt{v}_1 = \mathtt{S.Verify}(\overline{\mathtt{pk}}, \overline{\mathtt{msg}}, \mathtt{sig})$
return $[\![\mathtt{v}_0 = 1 \wedge \mathtt{v}_1 = 1]\!]$

Oracle $\mathsf{Sign}(\mathtt{sk}, \mathtt{msg})$

$\mathtt{sig} \leftarrow \mathtt{S.Sign}(\mathtt{sk}, \mathtt{msg})$
$\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(\mathtt{msg}, \mathtt{sig})\}$
return sig

Fig. 24: The notions MAL-S-P, LEAK-S-P, and HON-S-P.