Post-quantum security of cryptographic transformations in the random oracle model

Huang, Y.

Chapter 4

BUFF Transformations

4.1 Introduction

Beyond the standard unforgeability security for digital signature schemes, additional security properties are sometimes desirable, like *exclusive ownership* [PS05], *message-bound signatures*, and *non-resignability* [JCCS19]. The NIST explicitly mentioned them as "*additional desirable security properties*" in their call for additional post-quantum signatures [NIST22]. As discussed in [CDF+21], there are real-life attacks in certain applications that exploit the lack of these additional security properties.

Non-resignability for example requires, informally, that it is hard for an attacker to maul a signature σ for a message m into a signature σ' under his own public-key, when he only gets to see the signature σ of m (under someone else's public key) and some auxiliary information on m, but not the message m itself. The relevance of non-resignability has been shown in [JCCS19], where the authors identified an attack against the "Dynamically Recreatable Key" protocol [KBJ+14] that indeed applies in case the deployed signature scheme does not satisfy non-resignability, uncovering a flaw in the protocol's original security analysis [ZBPB17].

On top of discussing these additional security properties and their relevance in applications, Cremers, Düzlü, Fiedler, Fischlin, and Janson [CDF+21] offer a generic transformation, the *BUFF transform* (which stands for **B**eyond **U**n**F**orgeability **F**eatures), that turns any signature scheme into a new signature scheme that is argued to then satisfy these additional security properties either in the random oracle model (ROM) or in the plain model under some non-standard assumptions on the hash function $\mathcal{F}$.[1]

[1]The original publication [CDF+21] has been revised in reaction to [DFHS24], one of the works on which this chapter is based; our discussion here is with respect to the original version of [CDF+21]; we discuss the revision [CDF+23] explicitly in Section 4.1.2.

The BUFF transform is very simple and causes little computational overhead. Indeed, instead of signing the original message, the BUFF-transformed signature scheme simply signs the hash $y = \mathcal{F}(m\|\mathsf{pk})$ of the message and the public-key, to get a signature σ, and then outputs the pair (σ, y) as signature; verification works in the obvious way. Motivated by the reference in the NIST call and the little overhead caused by the BUFF transform, several of the submissions to the NIST call for additional post-quantum signatures have the BUFF transform built in, or mention the possibility of applying the BUFF transform to the proposed scheme.[2]

There have also been some claims about (some of) these additional security properties being achieved by the three signature schemes that were selected by the NIST in 2022 to be standardized. Indeed, Cremers *et al.* [CDF+21] argue that Dilithium [LDK+20] uses the BUFF transform *implicitly*, allowing them to apply their main result regarding the BUFF transform. Falcon [PFH+20] does not achieve the beyond unforgeability features; however, the Falcon team announced that they will deploy the BUFF transform to achieve them [FHK+22]. Finally, Cremers *et al.* expect that SPHINCS [HBD+20] also achieves non-resignability, though only using some informal arguments.

4.1.1 Our Contribution

In this chapter, we show that the security of BUFF transformation, particularly the non-resignability (NR), is more subtle than originally believed. We then launch a quest to re-establishing the "right" definition of NR, via studying strength and achievability of various candidate definitions, respectively.

Impossibility of NR. First, we show in Section 4.3 that the NR security, as defined in [CDF+21], is essentially unachievable. When considering the plain model, we observe that for any signature scheme with the property that there is sufficient (computational) entropy in the message when given its signature (and the public key), there is a simple attack that entirely breaks NR of the signature scheme.[3]

Given that, by design, the BUFF transform satisfies this entropy requirement, it follows directly that the BUFF transform does *not* satisfy NR in the plain model, regardless of the hash function used (and regardless of the hash function being fixed or chosen from a family of possible hash functions). We stress that not only is there no proof for the NR of the BUFF transform in the plain model, but our aforementioned attack easily breaks it.

Moving to the random oracle model (ROM), somewhat surprising in the light of the positive results claimed in [CDF+21] on the BUFF transform in

[2]The following submissions explicitly refer to the BUFF transform: Squirrels [ENST23], Racoon [dEK+23], HAWK [BBD+24], PROV [GCF+23], Vox [PCF+23], and eMLE [LZ23].

[3]On the other hand, if the message can be efficiently computed from its signature, NR is also not satisfied, as already pointed out in [CDF+21].

the ROM, we show that, as a matter of fact, also in the ROM the BUFF transform does not satisfy NR. The matter is slightly more subtle here since prior works did not rigorously define NR in the ROM. What we show is that for the *natural extension* of NR to the ROM, our negative results from the plain model carry over, and thus, in particular, that the BUFF transform does not achieve (this natural notion of) NR in the ROM.

Given the positive claims in prior work, we discuss what is wrong with the reasoning in [CDF+21], where the BUFF transform is claimed to satisfy NR. Namely, the issue lies in the Φ-non-malleability property of the random oracle, incorrectly claimed in [BFS11] and used in [CDF+21]. More precisely, we show that Φ-non-malleability as stated in these works is unachievable.

We note that our generic attack on NR is embarrassingly simple in retrospect. It exploits that there is no restriction on the attacker's auxiliary information on the signed message m, subject to that it does not reveal m; this pretty much allows to embed the mauled signature σ' into the auxiliary information, making the attacker's job of finding σ' trivial. This attack has no (direct) real-world impact, since the auxiliary information is typically not adversarially chosen, but determined by the application. Instead, the point of our attack is to show that the formal definition put forward in [CDF+21] is too strong, and that prior positive results on achieving NR are incorrect. Thus, we need to go back to the drawing board: both the formal definition as well as achievability results need to be revised. This is what we do, to a certain degree, in the main part of this chapter, as discussed below.

A weaker variant: $\mathsf{NR}^{H,\perp}$. Facing the above strong negative result, we introduce in Section 4.4 a weaker variant of the original definition of the non-resignability property, which is still meaningful from an application perspective yet avoids the above generic attack, by requesting the auxiliary information to be computed without access to the random oracle.[4] This definition is thus still meaningful whenever in the considered application the computation of the auxiliary information does not depend on the random oracle that is used in the signing process for the considered signature scheme (which can typically be enforced via domain separation). We call this weaker variant $\mathsf{NR}^{H,\perp}$ (pronounced "NR-bot").

A natural question then is whether the BUFF transform satisfies $\mathsf{NR}^{H,\perp}$. Interestingly, this remains a non-trivial problem; as a matter of fact, depending on the precise formulation of the entropy requirement, which captures that the signed message m should remain hidden to the attacker, we show yet another negative result (see below).

On the positive side, we show that $\mathsf{NR}^{H,\perp}$ is satisfied in the ROM by a *salted version* of the BUFF transform, *if* the entropy requirement on the message m

[4] A more radical solution is to disallow any auxiliary information altogether, which in essence is done in [CDF+23]; see later for a more elaborate discussion of [CDF+23].

is *statistical* (rather than computational). The salted version of the BUFF transform includes a random salt in the hash and appends the salt to the signature. This is proven in the classical as well as in the quantum ROM (with different respective reduction losses), covering thus both classical and quantum attacks. Yet again on the negative side, by means of a counterexample in the form of a contrived signature scheme, we show that the above result on the salted version of the BUFF transform does not carry over in case the entropy requirement on the message m is *computational* (by means of the HILL entropy), as originally considered in [CDF+21]. This in particular also applies to the original (unsalted) BUFF transform.

Secret-key non-resignability. In Section 4.5, we introduce yet another variant of non-resignability, which we call *secret-key non-resignability* (sNR), and we show that the (original) BUFF transform satisfies sNR in the statistical setting, where the entropy condition holds statistically and adversaries may be computationally unbounded. Looking ahead at Section 4.8, this positive result also carries over to the computational setting, where the entropy condition holds computationally and adversaries have bounded computing power only.

In the statistical setting, sNR is strictly stronger than $\mathsf{NR}^{H,\perp}$; in the computational setting, the two notions are (probably) incomparable, yet sNR is strictly stronger than the notion considered in [CDF+23]. Given that, as shown in Section 4.4, the BUFF transform does not satisfy $\mathsf{NR}^{H,\perp}$ in the computational setting, our results appear to be the best we can hope for towards proving positive results on the non-resignability of the BUFF transform.

The approach in Section 4.5 recycle and adjust some of the arguments from the analyses of salted BUFF in Section 4.4. The crucial part of course is when Section 4.4 exploits the salt that originates from the salted BUFF transform, which we cannot do in Section 4.5, given that the original, unsalted variant is considered. Instead, we capture the crucial, missing piece in terms of a simple security game of the underlying hash function $\mathcal{F}$ called *Hide-and-Seek*, in the random oracle model, where, slightly more general than the usual case, we consider $\mathcal{F}$ that is given query access to a random oracle H. In essence, the game asks to find x when given $\mathcal{F}^H(x)$ and query-bounded access to H, where x may be chosen arbitrarily *dependent* on H subject to the condition that it is hard to guess when given access to H only, i.e., without being given $\mathcal{F}^H(x)$. We then reduce the sNR property of the BUFF transform to the hardness of winning Hide-and-Seek, when $\mathcal{F}$ is modelled as a random oracle H.

Despite its simplicity and harmless appearance, this game turns out to be surprisingly tricky to analyze. Thus, the technical core of Section 4.5 is in analyzing Hide-and-Seek of the random oracle, and showing that it is hard to win, both in the classical and in the quantum setting.

Fine-grained attack on BUFF. A natural question that is still open from Section 4.5 though, is whether the BUFF transform satisfies sNR in a more fine-grained use of the random oracle model, when considering a real-life iterative-hash-function design (such as Merkle-Damgård [Mer79, Mer90, Dam90] or Sponge [BDPA07]), where merely the round function is modelled via an idealized function. One might expect that the results where $\mathcal{F}$ is modelled as a random oracle would carry over, and that it is mainly a question of extending the proof — however, as we will see in the following, this is not the case.

We describe in Section 4.6 a simple attack on sNR of the BUFF transform when instantiated with *any* iterative hash function, where the round function may have access to an idealized function. This covers both Merkle-Damgård (in the random oracle model) and Sponge-based hash functions (in the random permutation model), and thus the design principles behind SHA-2, SHA-3, and SHAKE.

Again, the devil lies in the auxiliary information, which here can be abused to communicate an intermediate hash value to the adversary, making its task of resigning the unknown message a very easy one. Indeed, the adversary can then finalize the computation of the hash $\mathcal{F}(m\|\mathsf{pk}')$ even when it has uncertainty in the first blocks of the message m, and then simply sign the hash using its secret key sk', resulting in a resigning of the (partially) unknown message m. We note that this attack also applies to other notions of non-resignability where the adversary receives auxiliary information about the message.

We note that our negative result does not contradict the fact that Merkle-Damgård and Sponge are known to be *indifferentiable* from a random oracle, and thus can securely replace a random oracle (in certain cases), since the latter only holds for single-stage games, while non-resignability is a two-stage game. In that light, it is also not surprising that our attack shows some resemblance to the counter example provided in [RSS11].

Our attack is theoretical in nature, as similar to the attack in Section 4.3, it also exploits an artificial choice of the message and auxiliary information, which would be very unlikely to actually occur in a real-world protocol. However, more realistic attacks exploiting the iterative structure of the hash function might exist, highlighting the importance of provable security of the non-resignability property in this model.

Regaining sNR via the Sandwich BUFF. Towards preventing the above attack, and with the hope to re-establish sNR for the BUFF transform in the considered setting, we propose in Section 4.7 a simple modification to the transform, where instead of hashing $m\|\mathsf{pk}$ for signing, the hash of $m\|\mathsf{pk}\|m$ is computed and signed. Due to this sandwich structure of the hash input, we call this variant of the BUFF transform Sandwich BUFF, or sBUFF for short.

The main technical challenge in Section 4.7 is to show that this modification not only mitigates the attack, but also allows us to prove sNR for the Merkle-

Damgård hash function, when the round function is modelled using a random oracle.[5] To this end, we follow the similar strategy as in Section 4.5, show how an adversary against the sNR can be used to break the Hide-and-Seek game, which we then show is hard to win for the Merkle-Damgård hash function.

Achieving sNR in the computational setting Finally, we show in Section 4.8 that all positive results regarding sNR carries over to the computationally setting, where attackers are computationally bounded, and where the underlying entropy requirement is computational. This includes the sNR property of BUFF when $\mathcal{F}$ is modelled as a random oracle, and the sNR property of Sandwich BUFF in the fine-grained idealization setting.

4.1.2 Related Work

We already mentioned [CDF+21] and [JCCS19], upon which our work builds up. In reaction to our negative results in [DFHS24], on which Section 4.3 is based, the authors of [CDF+21] have updated their work; we briefly discuss this update [CDF+23] here.

In order to avoid our negative results (cf. Theorem 4.9), which exploit that the auxiliary information can be misused to embed a mauled signature, the authors modified the definition of non-resignability to require the auxiliary information to be *computationally independent* of the message (see [CDF+23, Fig. 4] for the non-resignability game and [CDF+23, Definition 4.3] for the actual definition). This is equivalent to not allowing any auxiliary information at all, and thus weaker than the variant of non-resignability we consider. Indeed, in the security reduction it is argued that, due to the computational independence, one can drop the auxiliary information altogether, and so reduce the non-resignability of the original BUFF transform to a variant of Φ-non-malleability with no auxiliary information (see [CDF+23, Fig. 1, right]).

Interestingly though, in the updated version [CDF+23], the authors have not adjusted their reasoning for their claim on the random oracle satisfying (the now weaker notion of) Φ-non-malleability, which is the place where the original flaw was hiding. They still argue via the very same informal reasoning as in the original version (see the quote in our Section 4.3.3). Although it is tempting to believe that this argumentation is sufficient, it is actually not.

Indeed, by means of a simple counter example we show in Appendix A.1 that *no* hash function (or hash function family), including the random oracle, satisfies the considered (weaker) notion of Φ-non-malleability. Thus, their claim on the BUFF transform satisfying the version of non-resignability considered in [CDF+23], under the assumption that the hash function satisfies the considered Φ-non-malleability notion, is vacuous.

[5] In [FHK25], one of the works on which this chapter is based, we also analyzed the security of Sponge, but for simplicity it is omitted here.

4.2 Preliminaries

4.2.1 (HILL) Entropy

The HILL entropy is a computational variant of min-entropy. First, we recall that for two random variables x and y, the computational distance

$$\delta_s(x, y) := \max_C \mid \Pr[C(x) = 1] - \Pr[C(y) = 1] \mid$$

is the maximum distinguishing advantage over all circuits C of size s. Then, the HILL entropy is defined as below.

Definition 4.1. *For a pair of random variables (x, z), the* conditional HILL entropy *(with parameters δ and s) is defined as*

$$^{\delta,s}\mathsf{HILL}_\infty(x \mid z) := \max_y \mathsf{H}_\infty(y \mid z)\,,$$

where the max *is over all random variables y such that $\delta_s\big((x, z), (y, z)\big) \leq \delta$.*

Remark 4.2. *When switching to asymptotic notation, for a family of pairs of random variables $\{(x_\lambda, z_\lambda)\}_{\lambda\in\mathbb{N}}$, a bound $\mathsf{HILL}_\infty(x_\lambda \mid z_\lambda) \geq k(\lambda)$ then naturally means that for every λ there exists Y_λ such that $\mathsf{H}_\infty(y_\lambda \mid z_\lambda) \geq k(\lambda)$, and for every polynomial $s(\lambda)$ there is a negligible $\delta(\lambda)$ such that $\delta_{s(\lambda)}\big((x_\lambda, z_\lambda), (y_\lambda, z_\lambda)\big) \leq \delta(\lambda)$. In this case, we tend to omit the security parameter λ and simply write (x, y) and $\mathsf{HILL}_\infty(x \mid z) \geq k$.*

In the plain model, the notion of min-entropy and the above HILL entropy captures unpredictability of random variables. Throughout this chapter, though, it is also relevant to consider the similar notion of unpredictability in the ROM. However, the random oracle itself becomes a source of randomness. Thus, by choosing the message to be the hash of a fixed string, it has high min-entropy but is still easy to guess, by just making one query to the random oracle. In the statistical setting, this can be dealt with by considering the min-entropy and additionally conditioning on the (function table of the) random oracle. On the other hand, conditioning on the exponentially large function table of the random oracle is problematic when considering the HILL entropy, which is defined computationally.

Below, we consider a natural way to overcome this issue by introducing a notion of HILL entropy in the ROM. In spirit, a random variable x, which may be correlated with the random oracle, has high HILL entropy, if it is indistinguishable from a random variable that has high statistical entropy, and where indistinguishability must hold for efficient oracle algorithms that may query the random oracle. Formally, for two random variables x, y that may depend on the random oracle H, we define the computational distance relative

to H as

$$\delta^H_{s,q}(x,y) := \max_C \left| \Pr\left[1 \leftarrow C^H(x)\right] - \Pr\left[1 \leftarrow C^H(y)\right] \right| ,$$

where the maximum is taken over all circuits C of size s and additionally given at most q queries to H. The definition of the (conditional) HILL entropy HILL^H_∞ relative to H is then in line with Definition 4.1.

Definition 4.3. *For a pair of random variables (x, z), possibly dependent on the random oracle H, the conditional HILL entropy (with parameters δ, s and q) relative to the random oracle is defined as*

$$^{\delta,s,q}\mathsf{HILL}^H_\infty(x \mid z) := \max_y \mathsf{H}_\infty(y \mid z, H) ,$$

where the max *is over all random variables y with $\delta^H_{s,q}((x,z),(y,z)) \leq \delta$.*

Note that we can also consider a variant where the indistinguishability is captured via *quantum* circuits, but for the sake of simplicity, here we consider only the classical variant of HILL entropy. Furthermore, similarly to Remark 4.2, we may also use an asymptotic notation and omit the parameters.

4.2.2 Non-Resignability and Φ-Non-Malleability

For a signature scheme $\mathcal{S} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrfy})$, *non-resignability* is defined via game NR, given in Fig. 4.1, which involves an *adversary* $\mathcal{A}_{\mathsf{NR}} = (\mathcal{D}, \mathcal{A})$ and a (possibly randomized) auxiliary function aux. We say that $\mathcal{A}_{\mathsf{NR}}$ is PPT if $\mathcal{D}$ and $\mathcal{A}$ are. In spirit, the goal of the adversary is to turn a signature σ for an unknown message m into a signature for the same message, but under a different, adversarially chosen, public key. We write

$$\mathbf{Adv}^{\mathsf{NR}}_{\mathcal{S}}(\lambda, \mathcal{A}_{\mathsf{NR}}, \mathsf{aux}) = \Pr[1 \leftarrow \mathsf{NR}_{\mathcal{S}}]$$

for the probability that the $\mathsf{NR}_{\mathcal{S}}$ game outputs 1 when instantiated with signature scheme $\mathcal{S}$ and with adversary $\mathcal{A}$ and auxiliary function aux. Similarly, for variants of NR and for other games that we will consider. For simplicity, we will often leave the security parameter λ implicit. We stress that beyond the input given to $\mathcal{A}$, no additional information is communicated from $\mathcal{D}$ to $\mathcal{A}$.

$\mathsf{NR}_{\mathcal{S}}$:
1: $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}$
2: $m \leftarrow \mathcal{D}(\mathsf{pk})$
3: $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$
4: $(\sigma', \mathsf{pk}') \leftarrow \mathcal{A}(\mathsf{pk}, \sigma, \mathsf{aux}(m, \mathsf{pk}))$
5: $v := \mathsf{Vrfy}(\mathsf{pk}', m, \sigma')$
6: **return** $(v = 1 \wedge \mathsf{pk}' \neq \mathsf{pk})$

Figure 4.1: Security game $\mathsf{NR}_{\mathcal{S}}$ (in the plain model) with an explicit hint function aux and a signature scheme $\mathcal{S} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrfy})$. The original definition in [CDF+21] had both m and $\mathsf{aux}(m, \mathsf{pk})$ produced by $\mathcal{D}(\mathsf{pk})$. This change in the definition only makes our negative result stronger (since it is a restriction on how h is produced), and will be convenient later on when trying to restore positive results.

As pointed out in [CDF+21], an adversary $\mathcal{A}_{\mathsf{NR}} = (\mathcal{D}, \mathcal{A})$ can easily win NR if $\mathcal{A}$ can compute m; indeed, it can then just sign m under a public-key pk' for which it knows the secret key. Thus, for this to be a potentially hard game, we need to enforce an entropy requirement on m. In this work, we consider two variants, by requiring the *statistical* entropy $\mathsf{H}_{\infty}(m \mid \mathsf{pk}, a)$ or the *computational* entropy $\mathsf{HILL}_{\infty}(m \mid \mathsf{pk}, a)$ to be lower bounded, where m, pk and a are chosen as in NR.

Following [CDF+21] (subject to this minor change in the game NR mentioned in Fig. 4.1), non-resignability is defined as follows.

Definition 4.4. *In the plain model, a signature scheme $\mathcal{S} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrfy})$ is called non-resignable if for every PPT adversary $\mathcal{A}_{\mathsf{NR}} = (\mathcal{D}, \mathcal{A})$ and every PPT function aux that satisfy the computational entropy condition*

$$\mathsf{HILL}_{\infty}\limits_{\substack{(\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{KGen} \\ m \leftarrow \mathcal{D}(\mathsf{pk})}} \big(m \,\big|\, \mathsf{pk}, \mathsf{aux}(m, \mathsf{pk})\big) \geq \omega(\log \lambda) \tag{4.1}$$

it holds that $\mathbf{Adv}^{\mathsf{NR}}_{\mathcal{S}}(\mathcal{A}_{\mathsf{NR}}, \mathsf{aux}) \leq \mathsf{negl}(\lambda)$.

A related notion, which is used in [CDF+21] towards proving non-resignability of the BUFF transform (see Section 4.2.3), is Φ-non-malleability, first introduced in [BCFW09], for a hash function $\mathcal{F}$. The definition is via game $\Phi\text{-}\mathsf{NM}_{\mathcal{F}}$, given in Fig. 4.2, and, as for non-resignability, we need to require a certain amount of statistical entropy $\mathsf{H}_{\infty}(x \mid \mathsf{aux}(x))$ or computational entropy $\mathsf{HILL}_{\infty}(x \mid \mathsf{hk}, \mathsf{aux}(x))$, for the game to be non-trivial.

Remark 4.5. *Strictly speaking, [BCFW09, CDF+21] consider key-ed hash functions, while we restrict our attention to key-less hash functions for simplicity.*

$\Phi\text{-}\mathsf{NM}_{\mathcal{F}}$:
1: $x \leftarrow \mathcal{D}$
2: $y := \mathcal{F}(x)$
3: $(y', \phi) \leftarrow \mathcal{A}(y, \mathsf{aux}(x))$
4: **return** $[\, \phi \in \Phi \wedge \mathcal{F}(\phi(x)) = y' \wedge \phi(x) \neq x \,]$

Figure 4.2: Security game $\Phi\text{-}\mathsf{NM}_{\mathcal{F}}$ with explicit hint function aux and a hash function $\mathcal{F}$.

Following [BFS11, CDF+21], for a family Φ of functions, Φ-non-malleability is defined as follows. Looking ahead, of particular interest is the case where x consists of two parts, conveniently written as $x = (\mathsf{pk}, m)$, and Φ consists of shifts of pk but leaves m untouched.

Definition 4.6. *A hash function $\mathcal{F}$ is called Φ-non-malleable if for any PPT adversary $\mathcal{A}_{\mathsf{NR}} = (\mathcal{D}, \mathcal{A})$ that satisfies the computational entropy condition*

$$\underset{x \leftarrow \mathcal{D}}{\mathsf{HILL}^{\infty}_{\infty}}\big(x \,\big|\, \mathsf{aux}(x)\big) \geq \omega(\log \lambda) \tag{4.2}$$

it holds that $\mathbf{Adv}^{\Phi\text{-}\mathsf{NM}}_{\mathcal{F}}(\mathcal{A}) \leq \mathsf{negl}(\lambda)$.

4.2.3 BUFF Transform

The BUFF transform [CDF+21] is a generic transform for signature schemes to achieve additional security properties beyond standard unforgeability. The transformation comes in two variants — BUFF and BUFF-lite. Throughout this work, we only consider the former, stronger variant[6] which is displayed in Fig. 4.3.

KGen:	$\mathsf{Sign}(\mathsf{sk}, m)$:	$\mathsf{Vrfy}(\mathsf{pk}, m, \sigma)$:
1: $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}_{\mathcal{S}}$	1: $y := \mathcal{F}(m, \mathsf{pk})$	1: $(\sigma_{\mathcal{S}}, y) := \sigma$
2: **return** $(\mathsf{sk}, \mathsf{pk})$	2: $\sigma_{\mathcal{S}} \leftarrow \mathsf{Sign}_{\mathcal{S}}(\mathsf{sk}, y)$	2: **return** $\mathsf{Vrfy}(\mathsf{pk}, y, \sigma_{\mathcal{S}}) \wedge$
	3: **return** $\sigma := (\sigma_{\mathcal{S}}, y)$	3: $y = \mathcal{F}(m, \mathsf{pk})$

Figure 4.3: The signature scheme $\mathsf{BUFF}[\mathcal{S}, \mathcal{F}] = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrfy})$, obtained from applying the BUFF transform to $\mathcal{S} = (\mathsf{KGen}_{\mathcal{S}}, \mathsf{Sign}_{\mathcal{S}}, \mathsf{Vrfy}_{\mathcal{S}})$ with a hash function $\mathcal{F}$.

[6] The weaker one, BUFF-lite, does not achieve non-resignability, which is the focus of our work.

Remark 4.7. *A typical hash function takes as input a bit string, in which case we take it as understood that the hash* $\mathcal{F}(m, \mathsf{pk})$ *in BUFF is defined in terms of a unique representation of* (m, pk)*. Throughout the chapter, whenever this becomes relevant (like, in Section 4.6), we assume for simplicity that the public-key length* $|\,\mathsf{pk}\,|$ *is fixed and the hash is evaluated after concatenating its input, i.e.* $\mathcal{F}(m, \mathsf{pk}) := \mathcal{F}(m \| \mathsf{pk})$*.*

The idea of the BUFF transform is to sign the hash of the message and the public key, and to append this hash to the signature. This "binds" the public key to the signature, which ensures that no other keys can be generated that Ver such a signature, thus ensuring what is known as *exclusive ownership*. The idea behind *non-resignability* is as follows. In order to turn a signature into a new signature for the same message but under a different public key pk', the adversary needs to produce the hash value $y' := \mathcal{F}(m, \mathsf{pk}')$ for the modified public key pk' and the unknown message m, which should be hard by the Φ-non-malleability of the hash function. Indeed, [CDF+21] states the following result (where we omit the claims regarding further security properties that are not relevant to our work).

Claim 4.8 ([CDF+21, Theorem 5.5])**.** *Let* $\mathcal{S}$ *be an EUFCMA-secure signature scheme. Then the application of the BUFF transformation produces an EUFCMA-secure signature scheme* $\mathsf{BUFF}[\mathcal{S}, \mathcal{F}]$ *that additionally provides [...]* NR *if* $\mathcal{F}$ *is* Φ*-non-malleable where* $\Phi = \{\phi_{\mathsf{pk}'} \mid \mathsf{pk} \in \mathcal{K}\}$ *and* $\phi_{\mathsf{pk}'}(\mathsf{pk}, m) = (\mathsf{pk}', m)$*.*

In combination with the claim on the random oracle being Φ-non-malleable for this choice Φ from [BFS11], the authors of [CDF+21] then conclude non-resignability of the BUFF transform in the ROM.

4.3 On the Impossibility of Non-Resignability

In this section, we show strong negative results on the non-resignability property in general, and on the BUFF transform in particular.

First, we consider the plain model, where we show, by means of a simple attack, that non-resignability is not satisfied when applied to a signature scheme with the property that the message has high (computational) entropy when given its signature (and the public key).[7]

Since the BUFF transform, when applied to any signature scheme, satisfies the considered entropy requirement (assuming the hash function to be compressing), it follows directly that the BUFF transform does *not* satisfy non-resignability in the plain model, regardless of the hash function used. We

[7] In the other extreme, if the message can be efficiently computed from its signature, non-resignability is also not satisfied, as already pointed out in [CDF+21].

stress that not only is there no proof for the non-resignability of the BUFF transform in the plain model, but there is also an attack that breaks it.

These negative results from the plain model carry over to the random oracle model (ROM) when considering the *natural extension* of the non-resignability property to the ROM (prior works did not rigorously specify the property in the ROM). Thus, also in the ROM the BUFF transform does not (necessarily) satisfy non-resignability, invalidating the positive results claimed in [CDF+21] in that respect. In essence, the claim on the random oracle being Φ-non-malleable, made in [CDF+21, BFS11], is false.

4.3.1 Non-Resignability and BUFF Transform in the Plain Model

It is clear that the $\mathsf{NR}_{\mathcal{S}}$ game (Fig. 4.1) is easy to win if the message m can be efficiently computed from its signature σ. In the following theorem, we show that if, on the other hand, the signature scheme is such that the message m has high computational entropy given its signature (and the public key), then another attack applies.[8]

Theorem 4.9. *Let $\mathcal{S}$ be a signature scheme such that for a random message $m' \leftarrow \mathcal{M}$ and keys $(\mathsf{sk}', \mathsf{pk}') \leftarrow \mathsf{KGen}(1^\lambda)$ we have $\mathsf{HILL}_\infty(m' \mid \mathsf{pk}', \mathsf{Sign}(\mathsf{sk}', m)) \geq \omega(\log \lambda)$. Then there exists a PPT adversary $\mathcal{A}_{\mathsf{NR}} = (\mathcal{D}, \mathcal{A})$ and a PPT function aux such that the computational entropy condition (4.1) is satisfied, yet*

$$\mathbf{Adv}^{\mathsf{NR}}_{\mathcal{S}}(\mathcal{A}_{\mathsf{NR}}, \mathsf{aux}) \geq 1 - \mathsf{negl}(\lambda)\,.$$

The attack is surprisingly simple. Instead of burdening $\mathcal{A}$ with finding σ, which, intuitively, is hard since $\mathcal{A}$ does not know m, we simply let aux compute σ and hand it over to $\mathcal{A}$ as auxiliary information h. The entropy condition on the signature scheme then ensures that this is an eligible attack. The proof below spells out the details.

Proof. We construct the adversary $\mathcal{A}_{\mathsf{NR}} = (\mathcal{D}, \mathcal{A})$ and function aux as shown in Fig. 4.4. In the first stage, $\mathcal{D}$ samples a message uniformly at random and outputs it. The function aux, which receives m as an input, generates a new key pair $(\mathsf{sk}', \mathsf{pk}') \leftarrow \mathsf{KGen}(1^\lambda)$, computes $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}', m)$, and outputs the hint $h := (\sigma, \mathsf{pk}')$. In the second stage, $\mathcal{A}$ receives as input the public key pk, the signature $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$, and the hint $h = (\sigma, \mathsf{pk}')$, and it outputs (σ, pk').

By the completeness property, we have $\Pr[\mathsf{Vrfy}(\mathsf{pk}', m, \sigma) = 1] \geq 1 - \mathsf{negl}(\lambda)$. We further have $\Pr[\mathsf{pk}' \neq \mathsf{pk}] \geq 1 - \mathsf{negl}(\lambda)$ due to the high min-entropy of key generation.

[8] This leaves open only a very small, artificial gap for signature schemes that may potentially satisfy non-resignability: the message must be hard to compute from its signature while having low conditional HILL entropy.

It remains to argue that $\mathcal{A}_{\mathsf{NR}}$ satisfies the entropy condition (4.1). It holds that

$$\begin{aligned}\mathsf{HILL}_{\infty}\big(m \,\big|\, \mathsf{pk}, \mathsf{aux}(m, \mathsf{pk})\big) &= \mathsf{HILL}_{\infty}\big(m \,\big|\, \mathsf{aux}(m, \mathsf{pk})\big) \\ &= \mathsf{HILL}_{\infty}\big(m \,\big|\, \mathsf{pk}', \mathsf{Sign}(\mathsf{sk}', m)\big) \geq \omega(\log \lambda)\,,\end{aligned}$$

where the first equality holds by the independence of pk and $(m, \mathsf{aux}(m, \mathsf{pk}))$, the second equality holds by the construction of aux, and the last inequality holds from the entropy requirement. Taking all of this together, we get that $\mathcal{A}_{\mathsf{NR}}$ is a valid adversary playing $\mathsf{NR}_{\mathcal{S}}$ such that

$$\mathbf{Adv}^{\mathsf{NR}}_{\mathcal{S}}(\mathcal{A}_{\mathsf{NR}}, \mathsf{aux}) \geq 1 - \mathsf{negl}(\lambda)\,.$$

This concludes the proof. □

$\mathcal{D}(\mathsf{pk})$:	$\mathcal{A}(\mathsf{pk}, \sigma, h)$:	$\mathsf{aux}(m, \mathsf{pk})$:
1: $m \leftarrow \mathcal{M}$	1: $(\sigma, \mathsf{pk}') := h$	1: $(\mathsf{sk}', \mathsf{pk}') \leftarrow \mathsf{KGen}$
2: **return** m	2: **return** (σ, pk')	2: $\sigma \leftarrow \mathcal{S}.\mathsf{Sign}(\mathsf{sk}', m)$
		3: **return** (σ, pk')

Figure 4.4: Adversary $\mathcal{A}_{\mathsf{NR}} = (\mathcal{D}, \mathcal{A})$ and function aux used in the proof of Theorem 4.9.

Having established Theorem 4.9, it then follows as an immediate corollary that the BUFF-transform does not achieve non-resignability, no matter what hash function is used, as long as it is compressing, so that there is entropy in the message m when given $\mathcal{F}_{\mathsf{hk}}(\mathsf{pk}, m)$ (here, the entropy is even statistical).

Corollary 4.10. *Let $\mathcal{S}$ be a signature scheme, and let $\mathsf{BUFF}[\mathcal{S}, \mathcal{F}]$ be the signature scheme obtained via the BUFF transform obtained by using a (keyed) hash function $\mathcal{F}$ that compresses by at least the size of the public key plus $\omega(\log \lambda)$ bits. Then there exists a PPT adversary $\mathcal{A}_{\mathsf{NR}} = (\mathcal{D}, \mathcal{A})$ and a PPT function aux such that the entropy condition (4.1) is satisfied, yet*

$$\mathbf{Adv}^{\mathsf{NR}}_{\mathsf{BUFF}[\mathcal{S},\mathcal{F}]}(\mathcal{A}_{\mathsf{NR}}, \mathsf{aux}) \geq 1 - \mathsf{negl}(\lambda)\,.$$

Clearly, a non-compressing hash function avoids this particular attack; however, it is unclear if security would be restored (in particular in the light of Footnote 8). Also, from a practical point of view, hash functions used in the BUFFtransform will be compressing.

4.3.2 Non-Resignability and the BUFF Transform in the ROM

When considering the random oracle model, things become somewhat subtle. First, we note that no definition of non-resignability *in the ROM* has been explicitly provided in the previous literature; the definitions given in [CDF+21] are in the plain model. When switching to the ROM, one needs to specify who is given access to the random oracle. Clearly, considering a signature scheme "in the ROM", we give KGen, Sign, and Vrfy oracle access to the random oracle H.[9] Also, by default, the attacker is given oracle access to the random oracle. Thus, looking at Fig. 4.1, this means we certainly want to give $\mathcal{D}$ and $\mathcal{A}$ oracle access to the random oracle. But, say, what about the function aux? Given that in the original definition in [CDF+21], the auxiliary information h is actually computed by $\mathcal{D}$ (and our re-writing in terms of a function aux is for later convenience), it is natural to then also allow the function aux to have oracle access the random oracle. We make this explicit in Fig. 4.5, where we give the resulting security game for non-resignability *in the ROM*.

However, there is another subtle matter in the definition of non-resignability when switching to the ROM. Namely, the entropy condition (4.1), as well as its unconditional counterpart $\mathsf{H}_\infty(m \mid \mathsf{pk}, \mathsf{aux}^H(m, \mathsf{pk})) \geq \omega(\log \lambda)$, are not sufficient anymore for the definition to be meaningful. Indeed, $\mathcal{D}^H$ could simply choose m to be the hash of 0. In the ROM, this will be of high entropy and independent of pk; yet, $\mathcal{A}^H$ can easily recover it (as the hash of 0), and then honestly Sign it using his secret key. For this reason, in the definition of non-resignability in the ROM, we change the entropy requirement on the message to hold when additionally conditioning on the random oracle, i.e., we require

$$\mathsf{H}_\infty\limits_{\substack{(\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{KGen}^H(1^\lambda) \\ m \leftarrow \mathcal{D}^H(\mathsf{pk})}} \left(m \,\middle|\, \mathsf{pk}, \mathsf{aux}^H(m, \mathsf{pk}), H\right) \geq \omega(\log \lambda)\,. \tag{4.3}$$

We stress that we consider the *statistical* variant of the entropy condition here; with H an exponentially large function table, switching to the computational HILL variant will bring up further issues, which we want to avoid — for now (though we will look into this issue in Section 4.4.4). Furthermore, having this more stringent requirement on the attacker only makes our negative result stronger.

The following theorem shows that, considering the above natural definition of non-resignability in the ROM, the impossibility result from the plain model (Theorem 4.9) carries over pretty much in the obvious way.

Theorem 4.11. *Let H be a random oracle and $\mathcal{S}^H = (\mathsf{KGen}^H, \mathsf{Sign}^H, \mathsf{Vrfy}^H)$ be a signature scheme given query access to H such that for message $m' \leftarrow \mathcal{M}$ and key-pair $(\mathsf{sk}', \mathsf{pk}') \leftarrow \mathsf{KGen}^H$ we have $\mathsf{H}_\infty(m' \mid \mathsf{pk}', \mathsf{Sign}^H(\mathsf{sk}', m), H) \geq$*

[9] We make this explicit by writing KGen^H etc.

$\mathsf{NR}_{\mathcal{S}}^{H}$:
1: $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}^{H}(1^{\lambda})$
2: $m \leftarrow \mathcal{D}^{H}(\mathsf{pk})$
3: $\sigma \leftarrow \mathsf{Sign}^{H}(\mathsf{sk}, m)$
4: $(\sigma, \mathsf{pk}') \leftarrow \mathcal{A}^{H}\left(\mathsf{pk}, \sigma, \mathsf{aux}^{H}(m, \mathsf{pk})\right)$
5: $v := \mathsf{Vrfy}^{H}(\mathsf{pk}', m, \sigma)$
6: **return** $(v = 1 \wedge \mathsf{pk}' \neq \mathsf{pk})$

Figure 4.5: Security game $\mathsf{NR}_{\mathcal{S}}^{H}$ for the signature $\mathcal{S}^{H} = (\mathsf{KGen}^{H}, \mathsf{Sign}^{H}, \mathsf{Vrfy}^{H})$ in the random oracle model. In this variant, both the adversary and the function aux are granted access to the random oracle H.

$\omega(\log \lambda)$. *Then there exists a PPT adversary* $\mathcal{A}_{\mathsf{NR}}^{H} = (\mathcal{D}^{H}, \mathcal{A}^{H})$ *and a PPT function* aux^{H} *with access to* H *such that the entropy condition* (4.3) *is satisfied, yet*

$$\mathbf{Adv}_{\mathcal{S}}^{\mathsf{NR}^{H}}(\mathcal{A}_{\mathsf{NR}}, \mathsf{aux}) \geq 1 - \mathsf{negl}(\lambda)\,.$$

The proof follows essentially from the proof of Theorem 4.9 with some obvious adjustments regarding the random oracle.

Proof. For the signature scheme $\mathcal{S}^{H} = (\mathsf{KGen}^{H}, \mathsf{Sign}^{H}, \mathsf{Vrfy}^{H})$, we give adversaries $\mathcal{D}^{H}$ and $\mathcal{A}^{H}$ and the auxiliary function aux^{H} that wins the game $\mathsf{NR}_{\mathcal{S}}^{H}$ with overwhelming probability, in Fig. 4.6. In the first stage, $\mathcal{D}^{H}$ chooses a random message m that it will output. The auxiliary function, which receives the message m as input, first generates a new key pair $(\mathsf{sk}', \mathsf{pk}') \leftarrow \mathsf{KGen}^{H}(1^{\lambda})$, computes $\sigma' \leftarrow \mathsf{Sign}^{H}(\mathsf{sk}', y')$, and outputs $h := (\mathsf{pk}', \sigma')$. In the second state, $\mathcal{A}^{H}$ gets the public key pk, the signature σ (of message m using secret key sk), and the hint h (consisting of pk' and σ') as input and simply outputs $h = (\mathsf{pk}', \sigma')$. It is easy to see that $\mathsf{NR}_{\mathcal{S}}^{H}$ outputs 1 with overwhelming probability. This is because, by the correctness of $\mathcal{S}$, we have that σ' is a valid signature of m under pk' with overwhelming probability, and by the high min-entropy of a public key conditioned on H, we have $\mathsf{pk}' \neq \mathsf{pk}$ with overwhelming probability.

The remaining is to argue that $\mathcal{A}_{\mathsf{NR}}$ satisfies the entropy condition (4.3). It holds that

$$\begin{aligned}\mathsf{H}_{\infty}(m \mid \mathsf{pk}, \mathsf{aux}^{H}(m, \mathsf{pk}), H) &= \mathsf{H}_{\infty}(m \mid \mathsf{aux}^{H}(m, \mathsf{pk}), H) \\ &= \mathsf{H}_{\infty}(m \mid \mathsf{pk}', \mathsf{Sign}^{H}(\mathsf{sk}', m), H) \geq \omega(\log \lambda)\,.\end{aligned}$$

The first equality holds by noticing that $m \to (\mathsf{aux}^{H}(m, \mathsf{pk}), H) \to \mathsf{pk}$ forms a Markov chain, the second equality holds by the construction of aux^{H}, and the

last inequality holds from the entropy requirement. Collecting the above yields

$$\mathbf{Adv}^{\mathsf{NR}^H}_{\mathcal{S}}(\mathcal{A}_{\mathsf{NR}}, \mathsf{aux}) \geq 1 - \mathsf{negl}(\lambda)\,,$$

which concludes the proof. □

$\mathcal{D}^H(\mathsf{pk})$:	$\mathcal{A}^H(\mathsf{pk}, \sigma, h)$:	$\mathsf{aux}^H(m, \mathsf{pk})$:
1: $m \leftarrow \mathcal{M}$	1: $(\sigma, \mathsf{pk}') := h$	1: $(\mathsf{sk}', \mathsf{pk}') \leftarrow \mathsf{KGen}^H(1^\lambda)$
2: **return** m	2: **return** (σ, pk')	2: $\sigma \leftarrow \mathcal{S}.\mathsf{Sign}^H(\mathsf{sk}', m)$
		3: **return** (σ, pk')

Figure 4.6: Adversary $\mathcal{A}^H_{\mathsf{NR}} = (\mathcal{D}^H, \mathcal{A}^H)$ and function aux used in the proof of Theorem 4.11.

We can leverage Theorem 4.11 to show that a BUFF-transformed signature scheme does not achieve non-resignability in the ROM. This is formalized in the following corollary.

Corollary 4.12. *Let H be a random oracle, $\mathcal{F}^H$ be a PPT hash function given query access to H, compressing by at least the size of the public-key plus $\omega(\log \lambda)$ bits, $\mathcal{S}^H$ be a signature scheme given query access to H, and $\mathsf{BUFF}[\mathcal{S}, \mathcal{F}]$ be the signature scheme obtained via the BUFF transform with $\mathcal{F}^H$. Then there exists a PPT adversary $\mathcal{A}^H_{\mathsf{NR}} = (\mathcal{D}^H, \mathcal{A}^H)$ and a PPT function aux such that the entropy condition (4.3) is satisfied, yet*

$$\mathbf{Adv}^{\mathsf{NR}^H}_{\mathsf{BUFF}[\mathcal{S},\mathcal{F}]}(\mathcal{A}_{\mathsf{NR}}, \mathsf{aux}) \geq 1 - \mathsf{negl}(\lambda)\,.$$

The above negative claim on the BUFF transform contradicts [CDF+21], which claims that the BUFF transform does satisfy non-resignability in the ROM (though without being explicit about the definition in the ROM). We discuss below the source of the false positive claim.

4.3.3 Φ-Non-Malleability in the ROM

[CDF+21] argues the non-resignability of the BUFF transform (in the ROM) in two steps. First, they prove the security of the BUFF transform *in the plain model* under the assumption that the hash function (family) satisfies the notion of Φ-non malleability (for a certain class Φ of functions). The formal statement is given in [CDF+21, Theorem 5.5] (cf. Theorem 4.8). Then, the following is remarked in [CDF+21, page 9], from which it is then concluded that the BUFF transform satisfies non-resignability in the ROM.

> *We note that if we model H as a random oracle then the hash function satisfies the definition of Φ-non-malleability for any class Φ*

> *where the functions ϕ preserve sufficient entropy in x, as will be the case for our results. The reason is that the adversary can only output a related random oracle value y' if it has queried the random oracle about $\phi(x)$ before. But this is infeasible if $\phi(x)$ still contains enough entropy.*

Note that this claim originates from [BFS11], where a similar argument is made.

We show that this claim on the Φ-non-malleability of the random oracle is incorrect (under some mild assumption on Φ). As a matter of fact, the same kind of attack as for the non-resignability of signature schemes applies here as well: we can simply let aux compute the mauled hash value. This bypasses the argument that the adversary has to make this particular query to the random oracle.

First, we explicitly spell out in Fig. 4.7 the security game of Φ-non-malleability *in the ROM*, for any PPT hash function $\mathcal{F}^H$ with query access to the random oracle H. Then, we state the negative result in Theorem 4.13 below. Note that the latter in particular implies that the random oracle itself, i.e., when setting $\mathcal{F}^H = H$, is not Φ-non-malleable.

$\Phi\text{-}\mathsf{NM}^H_{\mathcal{F}}$:
1: $x \leftarrow \mathcal{D}^H$
2: $h := \mathsf{aux}^H(x)$
3: $y := \mathcal{F}^H(x)$
4: $(y', \phi) \leftarrow \mathcal{A}^H(y, h)$
5: **return** $(\mathcal{F}^H(\phi(x)) = y' \wedge \phi(x) \neq x)$

Figure 4.7: Security game $\Phi\text{-}\mathsf{NM}^H_{\mathcal{F}}$ for the hash function $\mathcal{F}^H$ in the ROM and an arbitrary function family Φ.

Theorem 4.13. *Let H be a random oracle, $\mathcal{F}^H : \mathcal{X} \to \mathcal{Y}$ be a PPT hash function given query access to H, compressing by least $\omega(\log \lambda)$ bits, and $\Phi \subseteq \mathcal{X}^{\mathcal{X}}$ be such that there is a PPT algorithm $\mathcal{D}_\Phi$ producing $\phi \in \Phi$ that does not fix most points with overwhelming probability, i.e.*

$$\Pr_{\substack{\phi \leftarrow \mathcal{D}_\Phi \\ x \leftarrow \mathcal{X}}} [\phi(x) = x] \leq \mathsf{negl}(\lambda) \; . \tag{4.4}$$

Then, there exist a PPT adversary $\mathcal{A}^H_{\Phi\text{-}\mathsf{NM}} = (\mathcal{D}^H, \mathcal{A}^H)$ and a polynomial-time computable function aux^H, both given query access to H, such that the entropy condition

$$\mathsf{H}_\infty\big(x \,\big|\, \mathsf{aux}^H(x), H\big) \geq \omega(\log \lambda)$$

is satisfied, yet

$$\mathbf{Adv}_{\mathcal{F}}^{\Phi\text{-}\mathsf{NM}^H}(\mathcal{A}_{\Phi\text{-}\mathsf{NM}}, \mathsf{aux}) \geq 1 - \mathsf{negl}(\lambda)\,.$$

Proof. We give a PPT adversary $\mathcal{A}^H_{\Phi\text{-}\mathsf{NM}} = (\mathcal{D}^H, \mathcal{A}^H)$ that wins the game $\Phi\text{-}\mathsf{NM}^H$ with overwhelming probability. Both $\mathcal{A}^H_{\Phi\text{-}\mathsf{NM}} = (\mathcal{D}^H, \mathcal{A}^H)$ and aux^H are given in Fig. 4.8. In the first stage, adversary $\mathcal{D}$ chooses x uniformly at random. The auxiliary function on input $x \in \mathcal{X}$, chooses a function $\phi \leftarrow \mathcal{D}_\Phi$, computes $y' := \mathcal{F}^H(\phi(x))$, and returns the pair (y', ϕ) as hint. In the second stage, adversary $\mathcal{A}$ gets $y = \mathcal{F}^H(x)$ along with the hint $h = (y', \phi)$ as input and outputs the (y', ϕ) from the hint. It is easy to see that $1 \leftarrow \Phi\text{-}\mathsf{NM}^H$, unless $\phi(x) = x$, which only happens with negligible probability due to (4.4). Furthermore, the entropy requirement follows from the fact that x is chosen uniformly from $\mathcal{X}$ independent of (ϕ, H), and that $\mathcal{F}^H$ is compressing by $\omega(\log \lambda)$ bits. □

$\mathcal{D}^H(\mathsf{pk})$	$\mathcal{A}^H(y, h)$	$\mathsf{aux}^H(x)$
1: $x \leftarrow \mathcal{X}$	1: $(y', \phi) := h$	1: $\phi \leftarrow \mathcal{D}_\Phi$
2: **return** x	2: **return** (y', ϕ)	2: $y' := \mathcal{F}^H(\phi(x))$
		3: **return** (y', ϕ)

Figure 4.8: The adversary $\mathcal{A}^H_{\Phi\text{-}\mathsf{NM}} = (\mathcal{D}^H, \mathcal{A}^H)$ and the function aux^H used in the proof of Theorem 4.13.

Remark 4.14. *The above theorem is stated for the random oracle model. It is easy to see, though, that the result carries over to the plain model for concrete hash functions.*

4.4 Salted BUFF and NR-bot

In this section, we partly recover from the negative results from the previous section by considering a salted version of the BUFF transform, and showing that it satisfies a weaker variant of non-resignability in the ROM. At the end of this section, we show another negative result, when the entropy requirement posed on the adversaries is only computational.

4.4.1 Positive Results

The formal specification of the salted BUFF transform, denoted $\$\text{-}\mathsf{BUFF}$, is given in Fig. 4.9. It matches with the original BUFF transform, except that some random salt s is added to the signature and used for the hash.

Our goal is to show that the salted BUFF transform satisfies the weaker variant of non-resignability in the ROM obtained by replacing the non-resignability

KGen:	$\mathsf{Sign}(\mathsf{sk}, m)$:	$\mathsf{Vrfy}(\mathsf{pk}, m, (\sigma, y, s))$:
1: $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}_{\mathcal{S}}$	1: $s \leftarrow \{0,1\}^{\ell}$	1: $y' := \mathcal{F}(m, \mathsf{pk}, s)$
2: **return** $(\mathsf{sk}, \mathsf{pk})$	2: $y := \mathcal{F}(m, \mathsf{pk}, s)$	2: $d := \mathsf{Vrfy}_{\mathcal{S}}(\mathsf{pk}, y', \sigma)$
	3: $\sigma \leftarrow \mathsf{Sign}_{\mathcal{S}}(\mathsf{sk}, y)$	3: **return** $(d = 1 \wedge y = y')$
	4: **return** (σ, y, s)	

Figure 4.9: The salted BUFF transform $\$\text{-}\mathsf{BUFF}[\mathcal{S}, \mathcal{F}]$ for a signature scheme $\mathcal{S} = (\mathsf{KGen}_{\mathcal{S}}, \mathsf{Sign}_{\mathcal{S}}, \mathsf{Vrfy}_{\mathcal{S}})$ and a hash function $\mathcal{F}$.

game $\mathsf{NR}_{\mathcal{S}}$ to $\mathsf{NR}_{\mathcal{S}}^{H,\perp}$, as given in Fig. 4.10. The only difference is that the function aux, which computes a piece of auxiliary information, is not given access to the random oracle anymore.

$\mathsf{NR}_{\mathcal{S}}^{H,\perp}$

1: $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}^{H}$
2: $m \leftarrow \mathcal{D}^{H}(\mathsf{pk})$
3: $(\sigma', \mathsf{pk}') \leftarrow \mathcal{A}^{H}\left(\mathsf{pk}, \mathsf{Sign}^{H}(\mathsf{sk}, m), \mathsf{aux}(m, \mathsf{pk})\right)$
4: **return** $\mathsf{Vrfy}^{H}(\mathsf{pk}', m, \sigma') \ \wedge \ \mathsf{pk}' \neq \mathsf{pk}$

Figure 4.10: The $\mathsf{NR}_{\mathcal{S}}^{H,\perp}$ game.

Indeed, below we will prove the following. To start with, we consider the entropy condition on the message to be statistical; as explained in Section 4.3.2, here in the ROM we additionally need to condition on H (in order to avoid letting $m = H(0)$). In Section 4.4.4, we then discuss the case of computational entropy.

We consider both the case of *classical* and of *quantum* queries by the adversary $\mathcal{A}$ when querying the random oracle, as well as a "semi-quantum" case where $\mathcal{D}$ is classical yet $\mathcal{A}$ may be quantum. The latter is motivated by the fact that $\mathcal{D}$ is typically not adversarially chosen, but determined by the considered application.[10]

Theorem 4.15. *Let H be a random oracle with co-domain denoted by $\mathcal{Y}$, let $\mathcal{S}^H = (\mathsf{KGen}, \mathsf{Sign}^H, \mathsf{Vrfy}^H)$ be a signature scheme with a key generation that has no access to H, and let $\$\text{-}\mathsf{BUFF}[\mathcal{S}, H]$ be the signature scheme obtained by applying the salted BUFF transform (cf. Fig. 4.9) to $\mathcal{S}$. Furthermore, let $\mathcal{A}_{\mathsf{NR}}^H = (\mathcal{D}^H, \mathcal{A}^H)$ be a computationally unbounded $\mathsf{NR}^{H,\perp}$ adversary, aux be any (possibly randomized) function, and let $\mathsf{Sign}^H, \mathcal{D}^H, \mathcal{A}^H$ make at most $q_S, q_{\mathcal{D}}, q_{\mathcal{A}}$*

[10] In the fully quantum case, we even allow the signing procedure to make quantum queries to H; this is not really relevant but obtained for free.

queries to H respectively. Assuming

$$\mathsf{H}_{\infty}^{\substack{(\mathsf{sk},\mathsf{pk})\leftarrow\mathsf{KGen}\\ m\leftarrow\mathcal{D}^{H}(\mathsf{pk})}}\big(m \,\big|\, H,\mathsf{pk},\mathsf{aux}(m,\mathsf{pk})\big) \geq \log(1/\epsilon)\,,$$

then for $\mathsf{Sign}^H, \mathcal{D}^H, \mathcal{A}^H$ making quantum queries in general, it holds that

$$\mathbf{Adv}^{\mathsf{NR}^{H,\perp}}_{\$\text{-}\mathsf{BUFF}[\mathcal{S},H]}(\mathcal{A}_{\mathsf{NR}},\mathsf{aux}) \leq \sqrt{q_{\mathcal{D}}\cdot 2^{-\ell}} + \frac{q_{\mathcal{D}}\cdot 2^{-\ell}}{2} + 4(q_{\mathcal{A}}+q_S)\sqrt{\epsilon} + \frac{(2q_{\mathcal{D}}+1)^2}{|\mathcal{Y}|}\,, \tag{4.5}$$

and if $\mathcal{D}^H$ is restricted to classical queries,

$$\mathbf{Adv}^{\mathsf{NR}^{H,\perp}}_{\$\text{-}\mathsf{BUFF}[\mathcal{S},H]}(\mathcal{A}_{\mathsf{NR}},\mathsf{aux}) \leq q_{\mathcal{D}}\cdot 2^{-\ell} + 4(q_{\mathcal{A}}+q_S)\sqrt{\epsilon} + \frac{(q_{\mathcal{D}}+1)}{|\mathcal{Y}|}\,. \tag{4.6}$$

In case where $\mathsf{Sign}^H, \mathcal{D}^H, \mathcal{A}^H$ are all restricted to classical queries, then we have

$$\mathbf{Adv}^{\mathsf{NR}^{H,\perp}}_{\$\text{-}\mathsf{BUFF}[\mathcal{S},H]}(\mathcal{A}_{\mathsf{NR}},\mathsf{aux}) \leq q_{\mathcal{D}}\cdot 2^{-\ell} + 2(q_{\mathcal{A}}+q_S)\cdot\epsilon + \frac{(q_{\mathcal{D}}+1)}{|\mathcal{Y}|}\,. \tag{4.7}$$

Remark 4.16. *In the setting where the key generation KGen^H is given access to the random oracle H, it is not too hard to extend the non-resignability of* \$-BUFF *into such a setting, by noticing that any sufficiently long portion of the random salt s in* \$-BUFF *is hard to guess by KGen^H, and hence separating the domain queried by KGen^H from ones queried by Sign^H and Vrfy^H up to some negligible advantage.*

4.4.2 Handling Classical Adversaries

Proof of (4.7). The proof proceeds via the games $\mathcal{G}_0,\ldots,\mathcal{G}_6^i$ displayed in Fig. 4.11. Steps from $\mathcal{G}_0$ to $\mathcal{G}_3$ are symmetric, in that they argue closeness between games, while each of the rest upperbounds the winning probability of one game via another.

The closeness between games $\mathcal{G}_0 \approx \mathcal{G}_1 \approx \mathcal{G}_2 \approx \mathcal{G}_3$ is via arguing that an adversary cannot detect some reprogramming in the random oracle except with small probability. For $\mathcal{G}_0 \approx \mathcal{G}_1$, one exploits that the reprogrammed point involves a freshly chosen salt s in uniform distribution. For $\mathcal{G}_1 \approx \mathcal{G}_2 \approx \mathcal{G}_3$, one exploits that m has high entropy, conditioned on the view of the attacker throughout the execution of the intermediate game $\mathcal{G}_2$.

$\mathcal{G}_0$ to $\mathcal{G}_1$ hop. The only difference between $\mathcal{G}_0$ and $\mathcal{G}_1$ is that the former computes $y \leftarrow H(m,\mathsf{pk},s)$, while the latter does a reprogramming via $H(m,\mathsf{pk},s) :=$

$\mathcal{G}_0$

1: $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}$
2: $m \leftarrow \mathcal{D}^H(\mathsf{pk})$
3: $h := \mathsf{aux}(m, \mathsf{pk})$
4: $s \leftarrow \{0,1\}^\ell$
5: $y := H(m, \mathsf{pk}, s)$
6: $(y', \mathsf{pk}', s') \leftarrow \mathcal{B}^H_{\mathsf{sk}}(y, h, s)$
7: **return** $H(m, \mathsf{pk}', s) = y' \wedge \mathsf{pk} \neq \mathsf{pk}'$

$\mathcal{B}^H_{\mathsf{sk}}(y, h, s)$

1: $\sigma \leftarrow \mathsf{Sign}^H(\mathsf{sk}, y)$
2: **return** $\mathcal{A}^H(\mathsf{pk}, \sigma, h)$

$\mathcal{C}^\bullet_{\mathsf{sk}}(m)$

1: $(s, y) \leftarrow \{0,1\}^\ell \times \mathcal{Y}$
2: **return** $\mathcal{B}^\bullet_{\mathsf{sk}}(y, \mathsf{aux}(m, \mathsf{pk}), s)$

$\mathcal{G}_1$

1: $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}$
2: $m \leftarrow \mathcal{D}^H(\mathsf{pk})$
3: $h := \mathsf{aux}(m, \mathsf{pk})$
4: $s \leftarrow \{0,1\}^\ell$
5: $H(m, \mathsf{pk}, s) := y \leftarrow \mathcal{Y}$
6: $(y', \mathsf{pk}', s') \leftarrow \mathcal{B}^H_{\mathsf{sk}}(y, h, s)$
7: **return** $H(m, \mathsf{pk}', s) = y' \wedge \mathsf{pk} \neq \mathsf{pk}'$

$\mathcal{G}_2$

1: $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}$
2: $m \leftarrow \mathcal{D}^H(\mathsf{pk})$
3: $h := \mathsf{aux}(m, \mathsf{pk})$
4: $s \leftarrow \{0,1\}^\ell$
5: $y \leftarrow \mathcal{Y}$
6: $(y', \mathsf{pk}', s') \leftarrow \mathcal{B}^H_{\mathsf{sk}}(y, h, s)$
7: **return** $H(m, \mathsf{pk}', s) = y' \wedge \mathsf{pk} \neq \mathsf{pk}'$

$\mathcal{G}_3$

1: $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}$
2: $m \leftarrow \mathcal{D}^H(\mathsf{pk})$
3: $h := \mathsf{aux}(m, \mathsf{pk})$
4: $s \leftarrow \{0,1\}^\ell$
5: $y \leftarrow \mathcal{Y}$
6: $(y', \mathsf{pk}', s') \leftarrow \mathcal{B}^{H[(m,\cdot,\cdot) \mapsto \perp]}_{\mathsf{sk}}(y, h, s)$
7: **return** $H(m, \mathsf{pk}', s) = y' \wedge \mathsf{pk} \neq \mathsf{pk}'$

$\mathcal{G}_4$

1: $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}$
2: $m \leftarrow \mathcal{D}^H(\mathsf{pk})$
3: $(y', \mathsf{pk}', s') \leftarrow \mathcal{C}^{H[(m,\cdot,\cdot) \mapsto \perp]}_{\mathsf{sk}}(m)$
4: **return** $H(m, \mathsf{pk}', s) = y'$

$\mathcal{G}^i_5$

1: $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}$
2: $m \leftarrow \mathcal{D}^H(\mathsf{pk})$
3: $(y', \mathsf{pk}', s') \leftarrow \mathcal{C}^{H[(m,\cdot,\cdot) \mapsto \perp]}_{\mathsf{sk}}(m)$
4: **return** $H(m, \mathsf{pk}', s) = y'$
5: $\qquad \wedge (m, \mathsf{pk}', s') = (m^i, \mathsf{pk}^i, s^i)$
6: {where $(m^i, \mathsf{pk}^i, s^i)$ is $\mathcal{D}$'s ith query}

$\mathcal{G}^i_6$

1: $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}$
2: $m \leftarrow \mathcal{D}^H(\mathsf{pk})$
3: $(y', \mathsf{pk}', s') \leftarrow \mathcal{C}^{H[(m^i,\cdot,\cdot) \mapsto \perp]}_{\mathsf{sk}}(m^i)$
4: **return** $H(m^i, \mathsf{pk}', s) = y'$
5: $\qquad \wedge (m, \mathsf{pk}', s') = (m^i, \mathsf{pk}^i, s^i)$
6: {where $(m^i, \mathsf{pk}^i, s^i)$ is $\mathcal{D}$'s ith query}

Figure 4.11: The sequence of games considered in the proof of (4.7). In all games, H is understood to be uniformly random. In the game $\mathcal{G}_4$ etc., $H[(m, \cdot, \cdot) \mapsto \perp]$ denotes the oracle that blocks queries of the form $(m, \cdot, \cdot)$, i.e., replies with some special value $\perp$ in that case, and replies with H applied to the query otherwise.

$y \leftarrow \mathcal{Y}$. Thus, both games behave identically unless $\mathcal{D}$ has queried the corresponding input (m, pk, s), which happens with probability at most $q_{\mathcal{D}} \cdot 2^{-\ell}$ in either game, due to the random choice of s.

$\mathcal{G}_1$ to $\mathcal{G}_2$ hop. Without loss of generality, we may assume $\mathsf{pk}' \neq \mathsf{pk}$, in which case the reprogramming of H, done in $\mathcal{G}_1$ but not in $\mathcal{G}_2$, does not affect the final hash $H(m, \mathsf{pk}', s')$. Thus, there is a difference in the two games only if $\mathcal{B}$ makes a query to (m, pk, s); however, in $\mathcal{G}_2$, using that y and s are independent of (m, H, h), $\mathsf{sk} \to (\mathsf{pk}, H, h) \to m$ is a Markov's chain, and by the entropy condition, we have that

$$\mathsf{guess}(m \mid \mathsf{sk}, H, y, h, s) = \mathsf{guess}(m \mid \mathsf{sk}, H, h) = \mathsf{guess}(m \mid \mathsf{pk}, H, h) \leq \epsilon \ ,$$

and so this happens with probability at most $(q_{\mathcal{A}} + q_{\mathcal{S}})\epsilon$. Thus,

$$\Pr\left[1 \leftarrow \mathcal{G}_1\right] \leq \Pr\left[1 \leftarrow \mathcal{G}_2\right] + (q_{\mathcal{A}} + q_{\mathcal{S}})\epsilon \ .$$

$\mathcal{G}_2$ to $\mathcal{G}_3$ hop. Similar as above, the difference between $\mathcal{G}_2$ and $\mathcal{G}_3$ can only be noticed when $\mathcal{B}$ makes a query of the form $(m, \cdot, \cdot)$, which again happens with probability at most $\mathsf{guess}(m \mid \mathsf{sk}, H, y, h, s) \leq \epsilon$. Therefore,

$$\Pr\left[1 \leftarrow \mathcal{G}_2\right] \leq \Pr\left[1 \leftarrow \mathcal{G}_3\right] + (q_{\mathcal{A}} + q_{\mathcal{S}})\epsilon \ .$$

$\mathcal{G}_3$ to $\mathcal{G}_4$ hop. In $\mathcal{G}_4$, we relax the winning condition by dropping the requirement $\mathsf{pk}' \neq \mathsf{pk}$; this only increases the winning probability. Furthermore, we replace $\mathcal{B}$ in $\mathcal{G}_3$ by $\mathcal{C}$ in $\mathcal{G}_4$, which computes $h := h(m)$ and samples $s \leftarrow \{0,1\}^{\ell}$ and $y \leftarrow \mathcal{Y}$ as a first step, and then runs $\mathcal{B}$ on input (y, h, s); this change is only syntactically and does not affect the winning probability. Thus,

$$\Pr\left[1 \leftarrow \mathcal{G}_3\right] \leq \Pr\left[1 \leftarrow \mathcal{G}_4\right] \ .$$

$\mathcal{G}_4$ to $\mathcal{G}_5^i$ hop. Since $\mathcal{D}$ is classical, assume without loss of generality that it never repeats a query. If (m, pk', s') has never been queried by $\mathcal{D}$, i.e., $(m, \mathsf{pk}', s') \neq (m^j, \mathsf{pk}^j, s^j)$ for all $j \in \{1, \ldots, q_{\mathcal{D}}\}$, then (using that $\mathcal{B}$ is blocked from queries of the form $(m, \cdot, \cdot)$) the hash $H(m, \mathsf{pk}', s')$ is random and independent of y, in which case they are equal with probability $1/|\mathcal{Y}|$. Therefore we obtain,

$$\begin{aligned}\Pr\left[1 \leftarrow \mathcal{G}_4\right] &= 1/|\mathcal{Y}| + \sum_{i \in [q_{\mathcal{D}}]} \Pr\begin{bmatrix} 1 \leftarrow \mathcal{G}_4 \\ (m, \mathsf{pk}', s') = (m^i, \mathsf{pk}^i, s^i) \end{bmatrix} \\ &= 1/|\mathcal{Y}| + \sum_{i \in [q_{\mathcal{D}}]} \Pr\left[1 \leftarrow \mathcal{G}_5^i\right] \ .\end{aligned}$$

$\mathcal{G}_5^i$ to $\mathcal{G}_6^i$ hop. In $\mathcal{G}_5^i$, due to the extra condition $m = m^i$ for winning the game, replacing m by m^i as in $\mathcal{G}_6^i$ has no effect on the winning probability, and dropping the requirement again then only increases the probability. Thus

$$\Pr\left[1 \leftarrow \mathcal{G}_5^i\right] \leq \Pr\left[1 \leftarrow \mathcal{G}_6^i\right] .$$

It remains to show that the latter probability is small. First, we may assume that $\mathcal{D}$'s queries $(m^j, \mathsf{pk}^j, s^j)$ are all distinct. Furthermore, we may assume that once $\mathcal{D}$ has decided on the ith query $(m^i, \mathsf{pk}^i, s^i)$, it stops without making this query; the game then simply proceeds as described with running $\mathcal{C}$ on input m^i. This shows that $\mathcal{C}$'s input is independent of $H(m^i, \mathsf{pk}^i, s^i)$, and so is his output y' then, given that he is blocked from queries of the form $(m, \cdot, \cdot)$. Hence

$$\Pr\left[1 \leftarrow \mathcal{G}_6^i\right] \leq 1/|\mathcal{Y}| .$$

Combining all the (in)equalities then concludes (4.7). □

4.4.3 Handling Quantum Adversaries

Proof of (4.5). The proof of (4.5) is identical to that of (4.7) up to some small changes in the argumentation for the first four game hops, and a more significant change of strategy in the last two. Indeed, we reuse the first four games and define modified versions of $\mathcal{G}_5^i$ and $\mathcal{G}_6^i$, as specified in Fig. 4.12. Namely, in case of superposition queries by $\mathcal{G}_0$, we cannot define $(m^i, \mathsf{pk}^i, s^i)$ as the ith query; instead, rather naturally, we introduce $(m^i, \mathsf{pk}^i, s^i)$ by *measuring* the ith query. Furthermore, for technical reasons, we then reprogram H on $(m^i, \mathsf{pk}^i, s^i)$ by a random value Θ, from this or the next query onward.

$\mathcal{G}_5^i$	$\mathcal{G}_6^i$
1: $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}$	1: $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}$
2: $m \leftarrow \mathcal{D}^{H_i^\Theta}$	2: $m \leftarrow \mathcal{D}^{H_i^\Theta}$
3: {measure ith query $(m^i, \mathsf{pk}^i, s^i)$}	3: {measure ith query $(m^i, \mathsf{pk}^i, s^i)$}
4: $(y', \mathsf{pk}', s') \leftarrow \mathcal{C}^{H[(m,\cdot,\cdot)\mapsto\perp]}(m)$	4: $(y', \mathsf{pk}', s') \leftarrow \mathcal{C}^{H[(m^i,\cdot,\cdot)\mapsto\perp]}(m^i)$
5: **return** $H^{\Theta_i}(m, \mathsf{pk}', s') = y'$	5: **return** $H^{\Theta_i}(m^i, \mathsf{pk}^i, s^i) = y'$
6: $\wedge (m, \mathsf{pk}', s') = (m^i, \mathsf{pk}^i, s^i)$	

Figure 4.12: The modified games $\mathcal{G}_5^i$ and $\mathcal{G}_6^i$ for the proof of (4.5). In both games, H is understood to be uniformly random. H_i^Θ is the oracle that implements H until just before the ith query, then measures that query in the computational basis to obtain $(m^i, \mathsf{pk}^i, s^i)$, and subsequently answers queries from $\mathcal{D}$ with $H[(m^i, \mathsf{pk}^i, s^i) \mapsto \Theta]$, i.e., with H but reprogrammed to a random value Θ at $(m^i, \mathsf{pk}^i, s^i)$, either from the ith or the $(i+1)$th query onward, with this choice being made uniformly at random as well.

$\mathcal{G}_0$ to $\mathcal{G}_1$ hop. The only difference between $\mathcal{G}_0$ and $\mathcal{G}_1$ is that the former computes $y := H(m, \mathsf{pk}, s)$, while the latter reprograms $H(m, \mathsf{pk}, s) := y \leftarrow \mathcal{Y}$. With s being uniformly random chosen, this is a direct application of the *adaptive reprogramming lemma* (Theorem 1 in [GHHM21]) to bound the distinguishing probability:

$$\Pr[1 \leftarrow \mathcal{G}_0] \leq \Pr[1 \leftarrow \mathcal{G}_1] + \sqrt{q_{\mathcal{D}} \cdot 2^{-\ell}} + \frac{q_{\mathcal{D}} \cdot 2^{-\ell}}{2} .$$

$\mathcal{G}_1$ to $\mathcal{G}_2$ hop. Without loss of generality, we may assume $\mathsf{pk}' \neq \mathsf{pk}$, in which case the reprogramming of H, done in $\mathcal{G}_1$ but not in $\mathcal{G}_2$, does not affect the final hash $H(m, \mathsf{pk}', s')$. Thus, the only difference between the two games is that $\mathcal{B}$ interacts with the original H in $\mathcal{G}_2$, and with H that is reprogrammed to $\perp$ at the point (pk, m, s) in $\mathcal{G}_1$.

This is a direct application for O2H ([AHU19, Theorem 3]). We note that in game $\mathcal{G}_2$, $\mathcal{B}$ has access to H, y, h and s. Using that y and s are independent of (sk, m, H, h) and that $m \to (\mathsf{pk}, H, h) \to \mathsf{sk}$ forms a Markov's chain, we obtain

$$\mathsf{guess}(m \mid \mathsf{sk}, H, y, h, s) = \mathsf{guess}(m \mid \mathsf{sk}, H, h) = \mathsf{guess}(m \mid \mathsf{pk}, H, h) \leq \epsilon ,$$

where the last inequality is given by the entropy condition. Thus, measuring a random query of $\mathcal{B}$ in $\mathcal{G}_2$ yields (pk, m, s) with probability at most ϵ. Therefore, by O2H,

$$\Pr[1 \leftarrow \mathcal{G}_1] \leq \Pr[1 \leftarrow \mathcal{G}_2] + 2(q_{\mathcal{A}} + q_{\mathcal{S}})\sqrt{\epsilon} .$$

$\mathcal{G}_2$ to $\mathcal{G}_3$ hop. Again by O2H - arguing as above that the measurement outcome when measuring a random query of $\mathcal{B}$ in $\mathcal{G}_2$ is of the form $(m, \cdot, \cdot)$ with probability at most ϵ - we obtain

$$\Pr[1 \leftarrow \mathcal{G}_2] \leq \Pr[1 \leftarrow \mathcal{G}_3] + 2(q_{\mathcal{A}} + q_{\mathcal{S}})\sqrt{\epsilon} .$$

$\mathcal{G}_3$ to $\mathcal{G}_4$ hop. Here we argue precisely as in the classical case: we relax the winning condition, and we do a syntactical change by introducing $\mathcal{C}$, which does the computation of h and the sampling of s and y locally, before it runs $\mathcal{B}$. Thus, also here

$$\Pr[1 \leftarrow \mathcal{G}_3] \leq \Pr[1 \leftarrow \mathcal{G}_4] .$$

$\mathcal{G}_4$ to $\mathcal{G}_5^i$ hop. In $\mathcal{G}_5^i$, the oracle for $\mathcal{D}$ is replaced by H_i^{Θ}, which implements H until just before the ith query, then measures that query in the computational basis to obtain $(m^i, \mathsf{pk}^i, s^i)$, and subsequently switches to $H[(m^i, \mathsf{pk}^i, s^i) \mapsto \Theta]$ either from the ith or the $(i+1)$th query onward, with this choice being made uniformly at random.

The goal here is to use the measure-and-reprogram technique from [DFM20] to control the effect of this change. For this purpose, we consider the oracle algorithm $\mathcal{E}^H(\mathcal{D},\mathcal{C})$, which simply runs $m \leftarrow \mathcal{D}^H$ followed by $(y',\mathsf{pk}',s') \leftarrow \mathcal{C}^{H[(m,\cdot,\cdot)\mapsto\perp]}(m)$.

We allow $\mathcal{E}$ conditional superposition query access to the random oracle H, which it uses to forward all queries from $\mathcal{D}$ unconditionally and queries x from $\mathcal{C}$ conditional on $x \neq (m,\cdot,\cdot)$, returning $\perp$ for $x = (m,\cdot,\cdot)$. $\mathcal{E}^H$ is thus an oracle algorithm with query complexity $q_0 + q_1$ and such that in its second phase the only query inputs with non-zero amplitude are of the form $x \neq (m,\cdot,\cdot)$. At the end of its run, $\mathcal{E}^H(\mathcal{D},\mathcal{C})$ outputs (x,z) with $x := (m,\mathsf{pk}',s')$ and $z := y'$.

Furthermore, we define the verification predicate $V(x,y,z)$ that is 1 if and only if $y = z$. Then, $V(x,H(x),z) = 1$ if and only if $H(m,\mathsf{pk}',s') = y'$, which is the verification condition in $\mathcal{G}_4$. Thus,

$$\Pr\left[V(x,H(x),z) = 1 : (x,z) \leftarrow \mathcal{E}^H(\mathcal{D},\mathcal{C})\right] = \Pr\left[1 \leftarrow \mathcal{G}_4(\mathcal{D},\mathcal{C})\right] .$$

We are now in a situation where we can apply a modified version of the measure-and-reprogram technique from Theorem A.4 in the Appendix, which ensures the existence of a "simulator" $\mathcal{S}^{\mathcal{E}}$ such that, for a random Θ,

$$\begin{aligned}&\frac{\Pr[V(x,H(x),z) = 1 : (x,z) \leftarrow \mathcal{E}^H]}{(2q+1)^2}\\ &\leq \Pr\left[V(x,\Theta,z) = 1 \wedge x = x' : (x',x,z,i) \leftarrow \langle \mathcal{S}^{\mathcal{E}}(Q),\Theta\rangle\right] ,\end{aligned}$$

where Q is a set of queries where $\mathcal{S}$ has non-zero probability of success, and $q = |Q|$. We need to describe $\mathcal{S}$ in a bit more detail before we can determine Q.

In the following, let $Q_0 := \{1,\ldots,q_{\mathcal{D}}\}$ and $Q_1 := \{q_{\mathcal{D}}+1,\ldots,q_{\mathcal{D}}+q_{\mathcal{A}}+q_{\mathcal{S}}\}$. The algorithm $\langle \mathcal{S}^{\mathcal{E}},\Theta\rangle$ works as follows: it measures the ith query of $\mathcal{E}^H$ for a random $i \in Q_0 \cup Q_1 \cup \{q_{\mathcal{D}} + q_{\mathcal{A}} + q_{\mathcal{S}} + 1\}$, with the measurement outcome being x', and then reprograms future queries to input x' by Θ (starting from this or the next query, chosen uniformly at random).[11] Finally, $\mathcal{S}$ outputs x' along with i and the final output (x,z) of $\mathcal{E}$.

In the case of our algorithm $\mathcal{E}$ it is easy to determine Q; $\mathcal{E}$ knows m by the end of its first phase, and by construction 1. never queries any input of the form $(m,\cdot,\cdot)$ from that point on and 2. at the end of its run outputs $x = (m,\mathsf{pk}',s')$. Hence, for $i \in Q_1$ we have $x \neq x'$ with certainty and thus

$$\Pr_{(x',x,z,i)\leftarrow\langle \mathcal{S}^{\mathcal{E}},\Theta\rangle}[V(x,\Theta,z) = 1 \wedge x = x' | i \in Q_1] = 0 .$$

It follows that $Q = Q_0$ and therefore $q = q_0$.

[11] The choice $i = q_{\mathcal{D}} + q_{\mathcal{A}} + q_{\mathcal{S}} + 1$ indicates that the final output (m,pk',s') of $\mathcal{E}$ is measured, instead of one of its queries.

Thus, *conditioned* on $i \in Q_0$, $\langle \mathcal{S}^{\mathcal{E}}, \Theta \rangle$ works as $\mathcal{D}^{H_i^{\Theta}}$ followed by $\mathcal{C}^{H[(m,\cdot,\cdot)\mapsto\perp]}(m)$ in $\mathcal{G}_5^i$ for a random $i \in Q_0$, and the event $V(x,\Theta,z) = 1 \wedge x = x'$ matches with the winning condition of $\mathcal{G}_5^i$.

Hence, omitting the specification $(x', x, z, i) \leftarrow \langle \mathcal{S}^{\mathcal{E}}, \Theta \rangle$ of the probability space and writing V as a shorthand for $V(x,\Theta,z)$ in the expressions to simplify notation, we obtain

$$\begin{aligned}
\Pr[V = 1 \wedge x = x'] &= \Pr[i \in Q_0] \Pr\left[V = 1 \wedge x = x' \mid i \in Q_0\right] \\
&\quad + \Pr[i \notin Q_0] \Pr\left[V = 1 \wedge x = x' \mid i \notin Q_0\right] \\
&= \frac{q_{\mathcal{D}}}{q_{\mathcal{D}}+1} \Pr\left[1 \leftarrow \mathcal{G}_5^i \mid i \in Q_0\right] \\
&\quad + \frac{1}{q_{\mathcal{D}}+1} \Pr\left[V = 1 \wedge x = x' \mid i \notin Q_0\right] . \qquad (4.8)
\end{aligned}$$

Finally, we argue that for $(x', x, z, i) \leftarrow \langle \mathcal{S}^{\mathcal{E}}, \Theta \rangle$

$$\Pr[V(x,\Theta,z) = 1 \wedge x = x' \mid i \notin Q_0] = \frac{1}{|\mathcal{Y}|} .$$

Consider $i \notin Q_0$, i.e. $\mathcal{S}$ measures the final output of $\mathcal{E}$. Then $\mathcal{C}$ learns no information on Θ before producing its output, and so $V(x,\Theta,z) = 1$ with probability $\frac{1}{|\mathcal{Y}|}$.

Putting all together, we obtain that

$$\frac{\Pr\left[1 \leftarrow \mathcal{G}_4\right]}{(2q_{\mathcal{D}}+1)^2} \leq \frac{q_{\mathcal{D}}}{q_{\mathcal{D}}+1} \Pr\left[1 \leftarrow \mathcal{G}_5^i \mid i \in Q_0\right] + \frac{1}{(q_{\mathcal{D}}+1) \cdot |\mathcal{Y}|} .$$

$\mathcal{G}_5^i$ to $\mathcal{G}_6^i$ hop. Let $i \in Q_0$ now be fixed. As in the classical case, due to the extra condition in $\mathcal{G}_5^i$, we may replace the occurrence of m with m^i without affecting the output of the game. Thus

$$\Pr\left[1 \leftarrow \mathcal{G}_5^i\right] = \Pr\left[1 \leftarrow \mathcal{G}_6^i\right] .$$

Similarly (but not identically) to the classical case, we can argue the latter probability to be small. Indeed, we may assume that $\mathcal{D}$ stops after having produced the ith query, which is then measured. This then means, given that $H[(m^i,\cdot,\cdot) \mapsto \perp])$ blocks the query that would reveal Θ, the output (y', pk', s') produced by $\mathcal{C}$ is independent of Θ. Thus, the probability that $y' = \Theta$ is at most $1/|\mathcal{Y}|$, showing that

$$\Pr\left[1 \leftarrow \mathcal{G}_6^i\right] = 1/|\mathcal{Y}| \qquad \text{for all fixed } i \in Q_0.$$

Substituting terms in Equation 4.8, we obtain that

$$\Pr\left[1 \leftarrow \mathcal{G}_4\right] \leq \frac{(2q_{\mathcal{D}}+1)^2}{|\mathcal{Y}|} \ .$$

Combining the above bounds concludes the proof. □

Given that, in typical applications, $\mathcal{D}$ is not adversarially chosen but determined by the environment, and typical applications take place in a classical environment, it makes sense to also consider the "semi-quantum case" in (4.6) where we restrict $\mathcal{D}$ to classical queries, but we still allow $\mathcal{A}$ to be quantum. By an appropriate mix-and-match of the classical proof (of (4.7)) and the fully quantum proof (of (4.5)), we immediately conclude (4.6).

4.4.4 Negative Results with *Computational* Entropy

Below, we show that our positive result on the salted BUFF transform in the ROM does not carry over to the computational setting when considering the entropy requirement to be computational, i.e., captured by the above notion of HILL entropy in the ROM. Concretely, we show that under the computational Diffie-Hellman (CDH) assumption, there exists a (contrived) signature scheme that is secure in the standard sense (and thus a meaningful signature scheme), but for which the salted BUFF transformation does not provide the computational variant of $\mathsf{NR}^{H,\perp}$. Formally, this is summarized in Theorem 4.17.

Theorem 4.17. *Let H be a random oracle with co-domain $\mathcal{Y}$. Assuming CDH is hard, there exists a signature scheme $\mathcal{S}^H = (\mathsf{KGen}, \mathsf{Sign}^H, \mathsf{Vrfy}^H)$ in the ROM, with a key generation that has no access to H, for which $\$\text{-}\mathsf{BUFF}[\mathcal{S}, H]$, obtained by applying the salted BUFF transform (see Fig. 4.9), satisfies the following:*

There exists a PPT adversary $\mathcal{A}^H_{\mathsf{NR}} = (\mathcal{D}^H, \mathcal{A}^H)$ given query access to H, and a PPT function aux without any query to H such that

$$\mathsf{HILL}^{H}_{\infty}_{\substack{(\mathsf{sk},\mathsf{pk})\leftarrow\$\text{-}\mathsf{BUFF}[\mathcal{S},H].\mathsf{KGen}(1^\lambda)\\ m\leftarrow\mathcal{D}^H(\mathsf{pk})}} \left(m \,\middle|\, \mathsf{pk}, \mathsf{aux}(m,\mathsf{pk})\right) \geq \log(|\mathcal{Y}|) \ ,$$

and yet they win the game $\mathsf{NR}^{H,\perp}$ against $\$\text{-}\mathsf{BUFF}[\mathcal{S}, H]$ with overwhelming probability, i.e.,

$$\mathbf{Adv}^{\mathsf{NR}^{H,\perp}}_{\$\text{-}\mathsf{BUFF}[\mathcal{S},H]}(\mathcal{A}_{\mathsf{NR}}, \mathsf{aux}) \geq 1 - \mathsf{negl}(\lambda) \ .$$

Moreover, $\mathcal{S}^H$ is strongly unforgeable under chosen message attacks.

Let $S^H_\circ$ be an arbitrary CDH-based (strongly unforgeable) signature scheme, with a key generation that does not query H. Define S to be as $S_\circ$, but modified

as follows. The key generation additionally produces a pair (a, g^a), and attaches a to the secret key and g^a to the public key. Furthermore, signing attaches a to the actual signature (but will be ignored by the verification). Then, we consider an attacker that produces the message m as $m := (H(g^{ab}), g^b)$, and the auxiliary function $\mathsf{aux}(m, \mathsf{pk}) := g^b$. Then we have

$$\mathsf{HILL}^{H}_{\infty}\big(m \,\big|\, \mathsf{pk}, \mathsf{aux}(m, \mathsf{pk})\big) \geq \mathsf{HILL}^{H}_{\infty}\big(H(g^{ab}) \,\big|\, g^a, g^b\big)\,,$$

which is at least as large as $\log(|\mathcal{Y}|)$ by the CDH assumption; yet when given the signature of m, which includes a (be it BUFF transformed or not), the attacker can compute all of m and so produce a new signature by freshly signing m, which breaks the $\mathsf{NR}^{H,\perp}$ security of \$-$\mathsf{BUFF}[\mathcal{S}, H]$.

4.5 BUFF and sNR

4.5.1 Secret-key Non-resignability (sNR)

In this section, we consider a new variant of *non-resignability*, called secret-key non-resignability and denoted by $\mathsf{sNR}^{H,\perp}$. It is similar in spirit as $\mathsf{NR}^{H,\perp}$ introduced in Section 4.4; in particular, a crucial aspect is that aux is not given access to H, but we additionally provide the adversary with the secret key sk, and we adjust the entropy condition correspondingly (see below for a more detailed comparison). The security game is shown in Fig. 4.13. It is played by randomized oracle algorithms[12],

$$\mathcal{D}^H : \mathcal{SK} \to \mathcal{M} \qquad \text{and} \qquad \mathcal{A}^H : \mathcal{SK} \times \mathcal{SGN} \times \mathcal{AUX} \to \mathcal{PK} \times \mathcal{SGN}$$

given query access to H, referred to as *adversaries*, and a randomized algorithm $\mathsf{aux} : \mathcal{SK} \times \mathcal{M} \to \mathcal{AUX}$ with no access to H, referred to as *hint function.*[13]

$\mathsf{sNR}^{H,\perp}_{\mathcal{S}}(\mathcal{D}, \mathcal{A}, \mathsf{aux})$:
1: $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}^H$
2: $m \leftarrow \mathcal{D}^H(\mathsf{sk})$
3: $\sigma \leftarrow \mathsf{Sign}^H(\mathsf{sk}, m)$
4: $(\mathsf{pk}', \sigma') \leftarrow \mathcal{A}^H(\mathsf{sk}, \sigma, \mathsf{aux}(m, \mathsf{sk}))$
5: **return** $\mathsf{pk} \neq \mathsf{pk}' \wedge \mathsf{Vrfy}^H(\mathsf{pk}', m, \sigma') = 1$

Figure 4.13: Our new variant of the non-resignability game $\mathsf{sNR}^{H,\perp}$.

[12] Here and in the remainder, we borrow from set notation to indicate the input and output space of (oracle) algorithms. In case of an algorithm that takes no input, we write the singleton set $\{\perp\}$ as domain.

[13] The hint function may be randomized, but we refer to it as a function for convenience.

While playing $\mathsf{sNR}^{H,\perp}$, we consider restricted ($\mathcal{S}$-dependent) classes of adversaries with a give bound h on the entropy

$$\mathsf{H}_{\infty}^{\substack{(\mathsf{sk},\mathsf{pk})\leftarrow\mathsf{KGen}^{H} \\ m\leftarrow\mathcal{D}^{H}(\mathsf{sk})}}(m \mid H, \mathsf{sk}, \mathsf{aux}(\mathsf{sk}, m)) \geq h\,. \tag{4.9}$$

For now we only consider the statistical variant, where we take an arbitrary but fixed security parameter for $\mathcal{S}$, where $\mathcal{D}$, $\mathcal{A}$ and aux may be computationally unbounded and we only limit their query complexity, and where the entropy requirement holds statistically, i.e., as in (4.9). The computational setting is handled later in Section 4.8; there, $\mathcal{D}, \mathcal{A}$ and aux are restricted to be (uniform or non-uniform) PPT algorithms, and the entropy requirement is expressed via HILL entropy.

Informally, we say that a signature scheme $\mathcal{S} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrfy})$ is *non-resignable* if for all $\mathcal{D}$, $\mathcal{A}$ and any hint function aux that satisfy the statistical entropy condition (4.9) for sufficiently large h, the probability of winning the $\mathsf{sNR}^{H,\perp}$ game, i.e.,

$$\mathbf{Adv}^{\mathsf{sNR}^{H,\perp}}(\mathcal{D}, \mathcal{A}, \mathsf{aux}) := \Pr\left[1 = \mathsf{sNR}_{\mathcal{S}}^{H,\perp}(\mathcal{D}, \mathcal{A}, \mathsf{aux})\right]\,,$$

is small.

The recent developments have shown that formalizing non-resignability is a non-trivial task, and different weaker variants of the original (unachievable) version have been proposed. We quickly discuss here how $\mathsf{sNR}^{H,\perp}$ relates to those variants; namely, we show that is stronger than the versions proposed in Section 4.4 and [CDF+23].

Comparison with Non-Resignability from Section 4.4. The difference to $\mathsf{NR}^{H,\perp}$ as defined in Section 4.4 is that $\mathsf{sNR}^{H,\perp}$ provides the $\mathcal{D}$, $\mathcal{A}$ and the hint function aux with the secret key sk, whereas $\mathsf{NR}^{H,\perp}$ only provides the public key pk (recall that we assume that pk can be computed from sk). This of course gives more power to the adversary. The other difference lies in the entropy requirement: for $\mathsf{NR}^{H,\perp}$, the message is required to have high entropy conditioned on pk (and aux) only, i.e.,

$$\mathsf{H}_{\infty}(m \mid H, \mathsf{pk}, \mathsf{aux}(\mathsf{pk}, m)) \geq h$$

whereas $\mathsf{sNR}^{H,\perp}$ requires (4.9) to hold, which conditions on sk instead; this seems to be a stronger restriction, but we observe that for $m \leftarrow \mathcal{D}(\mathsf{pk})$, produced by a $\mathcal{D}$ that only gets the public key as input (as in $\mathsf{NR}^{H,\perp}$),

$$\mathsf{H}_{\infty}(m \mid H, \mathsf{pk}, \mathsf{aux}(\mathsf{pk}, m)) = \mathsf{H}_{\infty}(m \mid H, \mathsf{sk}, \mathsf{aux}(\mathsf{pk}, m))$$

since $\mathsf{sk} \to (H, \mathsf{pk}, \mathsf{aux}(\mathsf{pk}, m)) \to m$ forms a Markov chain then. This implies that any attack against $\mathsf{NR}^{H,\perp}$ can be cast as an attack against $\mathsf{sNR}^{H,\perp}$ with the same entropy bound, making the latter a stronger security notion.

Comparison with Non-Resignability from [CDF+23]. We first note that [CDF+23] defines non-resignability only in the computational setting, so we compare it with the computational version of $\mathsf{sNR}^{H,\perp}$. While we have postponed the exact definition to Section 4.8, the high level reasoning can still be understood. First of all, in [CDF+23] the side information on m (given by aux in our case) is required to be *computationally independent* of m, which is equivalent to allowing *no side information at all* when considering computationally bounded adversaries. Furthermore, in line with $\mathsf{sNR}^{H,\perp}$, the entropy condition (though phrased in terms of HILL entropy) is required to hold when conditioning on the secret key sk. But on the other hand and in the spirit of $\mathsf{NR}^{H,\perp}$, the adversaries are only given pk as input, and not sk. Altogether, this makes their notion weaker than our computational version of $\mathsf{sNR}^{H,\perp}$, which provided sk as input to the adversaries.

4.5.2 The Hide-and-Seek Game

To prove the non-resignability (in the sense of $\mathsf{sNR}^{H,\perp}$) of the BUFF transform, which signs a message m by signing $\mathcal{F}(\mathsf{pk}, m)$, with the hash value then appended to the signature, it must *necessarily* be hard to recover m from $H(m, \mathsf{pk})$. This hardness may look trivial at first glance, since $\mathcal{F}$ is (typically) compressing, and modelled as a random oracle; however, it turns out to be not trivial at all. The reason is that in the $\mathsf{sNR}^{H,\perp}$ game, m is produced arbitrarily and dependent on $\mathcal{F}$, with the only promise being that m is hard to guess from scratch (i.e., when $\mathcal{F}(\mathsf{pk}, m)$ is not given).

We formally capture (a particular formulation of) this hardness via a game, which we call *Hide-and-Seek*. Looking ahead, in Section 4.5.4 we will show that hardness of winning Hide-and-Seek is *sufficient* for proving the non-resignability of the BUFF transform (when the hash function is modelled as a random oracle). The main technical challenge in Section 4.5 then lies in proving that Hide-and-Seek is hard to win, which we do in Section 4.5.5.

Throughout the remainder of this section, let $\mathcal{X}, \mathcal{Y}$, and $\mathcal{Z}$ be finite non-empty sets, and let $\mathcal{F}^H \colon \mathcal{X} \to \mathcal{Y}$ be any hash function given query access to a random oracle H.

The Hide-and-Seek game is played by two adversaries $\mathcal{D}$ and $\mathcal{A}$: the (possibly query-unbounded) *hider* $\mathcal{D}^H \colon \{\perp\} \to \mathcal{X} \times \mathcal{Z}$, and the query-bounded *seeker* $\mathcal{A}^H \colon \mathcal{Y} \times \mathcal{Z} \to \mathcal{X}$ that is allowed to make at most q queries to H. First, $\mathcal{D}^H$ chooses a challenge $x \in \mathcal{X}$ together with a hint $z \in \mathcal{Z}$ and "hides" x as $\mathcal{F}^H(x)$, and then $\mathcal{A}^H$ is supposed to find x from $H(x)$ and z. The game is formally specified as follows:

$\mathsf{HnS}^H_{\mathcal{F}}(\mathcal{D}, \mathcal{A})$:
1: $(x, z) \leftarrow \mathcal{D}^H$
2: **return** $x = \mathcal{A}^H(\mathcal{F}^H(x), z)$

In line with the entropy condition in $\mathsf{sNR}^{H,\perp}$, we require x to be statistically hidden given H and z. I.e., we require that

$$\mathsf{guess}(x \mid H, z) \leq \epsilon \tag{4.10}$$

for some small $\epsilon > 0$. Informally, we say that the random oracle H satisfies the Hide-and-Seek property, or HnS^H for short, if for every such pair of $\mathcal{D}, \mathcal{A}$ as above, the winning probability, given as

$$\mathbf{Adv}^{\mathsf{HnS}^H}_{\mathcal{F}}(\mathcal{D}, \mathcal{A}) \coloneqq \Pr\left[1 = \mathsf{HnS}^H_{\mathcal{F}}(\mathcal{D}, \mathcal{A})\right] = \Pr_{(x,z)\leftarrow\mathcal{D}^H}\left[x = \mathcal{A}^H(\mathcal{F}^H(x), z)\right],$$

is small.

As mentioned above already, what is tricky about this game is that x (and z) may depend arbitrarily on H, subject to the bound (4.10) on the guessing probability. Because of this, known results on inverting the random oracle do not apply, and it may not be fully clear whether we can actually expect it to be hard to win, i.e., that there is no sneaky way to win the game. We discuss this in more detail in Section 4.5.5, where we then analyze Hide-and-Seek and prove that it *is* hard to win after all.

4.5.3 BUFF via Random Oracles is sNR

Our main goal in this section is to prove that BUFF satisfy the above notion of non-resignability, when modelling the underlying hash function $\mathcal{F}^H := H$ as a random oracle, which we formally state below.

Theorem 4.18. *Let $\mathcal{D}^H : \mathcal{SK} \to \mathcal{M}$ and $\mathcal{A}^H : \mathcal{SK} \times \mathcal{SGN} \times \mathcal{AUX} \to \mathcal{PK} \times \mathcal{SGN}$ be $\mathsf{sNR}^{H,\perp}$-adversaries against $\mathsf{BUFF}[\mathcal{S}, H]$ for some $\mathsf{aux} : \mathcal{SK} \times \mathcal{M} \to \mathcal{AUX}$, making at most q_D and q_A queries to H, respectively, where (4.9) is satisfied for h such that $0 < \epsilon := 2^{-h} \leq \frac{1}{2}$. Then*

$$\mathbf{Adv}^{\mathsf{sNR}^{H,\perp}}_{\mathsf{BUFF}[\mathcal{S},H]}(\mathcal{D}, \mathcal{A}, \mathsf{aux}) \leq 6(q_A + q_S + 1)^2 \log\left(\frac{|\mathcal{SK}| \cdot |\mathcal{AUX}|}{\epsilon}\right)\epsilon + q_K\epsilon + \frac{q_D + 1}{|\mathcal{Y}|}$$

and in the case where $\mathcal{A}$ makes quantum queries, we have $\mathbf{Adv}^{\mathsf{sNR}^{H,\perp}}_{\mathsf{BUFF}[\mathcal{S},H]}(\mathcal{D}, \mathcal{A}, \mathsf{aux})$

$$\leq 18\sqrt{\left(\log\frac{|\mathcal{SK}| \cdot |\mathcal{AUX}|}{\epsilon} + q_{\mathcal{A}} + q_S\right)(q_{\mathcal{A}} + q_S + 1)^3\,\epsilon} + q_K\epsilon + \frac{q_D + 1}{|\mathcal{Y}|}\,.$$

Remark 4.19. *In the case where $\mathcal{D}$ makes quantum queries to H, we expect a similar result (with adjusted bounds) holds, via the method described in the proof of* (4.5) *.*

4.5.4 Reducing sNR of BUFF to Hide-and-Seek

In the following statement, we reduce the $\mathsf{sNR}^{H,\perp}$ security of the BUFF transform $\mathsf{BUFF}[\mathcal{S}, H]$ of a signature scheme $\mathcal{S} = (\mathsf{KGen}^H, \mathsf{Sign}^H, \mathsf{Vrfy}^H)$ to the hardness of winning the Hide-and-Seek game HnS^H. In the lemma statement, the parameters q_K and q_S refer to (an upper bound on) the number of queries to H that KGen^H and Sign^H perform.

Lemma 4.20. *Let $\mathcal{D}^H : \mathcal{SK} \to \mathcal{M}$ and $\mathcal{A}^H : \mathcal{SK} \times \mathcal{SGN} \times \mathcal{AUX} \to \mathcal{PK} \times \mathcal{SGN}$ be $\mathsf{sNR}^{H,\perp}$-adversaries against $\mathsf{BUFF}[\mathcal{S}, H]$ for some $\mathsf{aux} : \mathcal{SK} \times \mathcal{M} \to \mathcal{AUX}$, making at most q_D and q_A queries to H, respectively. Then there exists a hider $\bar{\mathcal{D}} : \{\perp\} \to \mathcal{X} \times \mathcal{Z}$ and a seeker $\bar{\mathcal{A}} : \mathcal{Y} \times \mathcal{Z} \to \mathcal{X}$ with $\mathcal{Z} = \mathcal{SK} \times \mathcal{AUX}$, where $\bar{\mathcal{A}}$ makes at most $q_A + q_S$ queries to H, and such that*

$$\mathsf{H}_\infty\underset{(x,z)\leftarrow\bar{\mathcal{D}}^H}{}(x \mid H, z) = \mathsf{H}_\infty\underset{\substack{(\mathsf{sk},\mathsf{pk})\leftarrow\mathsf{KGen}^H \\ m\leftarrow\mathcal{D}^H(\mathsf{sk})}}{}(m \mid H, \mathsf{sk}, \mathsf{aux}(\mathsf{sk}, m)) \tag{4.11}$$

and $\mathbf{Adv}^{\mathsf{sNR}^{H,\perp}}_{\mathsf{BUFF}[\mathcal{S},H]}(\mathcal{D}, \mathcal{A}, \mathsf{aux})$

$$\leq (q_A + q_S) \cdot \mathbf{Adv}^{\mathsf{HnS}^H}_H(\bar{\mathcal{D}}, \bar{\mathcal{A}}) + q_K\epsilon + \frac{q_D + 1}{|\mathcal{Y}|}\,, \tag{4.12}$$

where $\epsilon := 2^{-\mathsf{H}_\infty(x|H,z)}$. In the case $\mathcal{A}$ makes quantum queries to H, then

$$\mathbf{Adv}^{\mathsf{sNR}^{H,\perp}}_{\mathsf{BUFF}[\mathcal{S},H]}(\mathcal{D}, \mathcal{A}, \mathsf{aux}) \leq 2(q_A + q_S) \cdot \sqrt{\mathbf{Adv}^{\mathsf{HnS}^H}(\bar{\mathcal{D}}, \bar{\mathcal{A}})} + q_K\epsilon + \frac{q_D + 1}{|\mathcal{Y}|}\,, \tag{4.13}$$

holds in place of (4.12)*, and $\bar{\mathcal{A}}$ then makes quantum queries as well.*

Furthermore, in the computational setting when considering a non-fixed security parameter and PPT algorithms $\mathcal{D}$ and $\mathcal{A}$, then $\bar{\mathcal{D}}$ and $\bar{\mathcal{A}}$ are PPT as well.

The intuition behind the proof is as follows. Consider the $\mathsf{sNR}^{H,\perp}$ game. Due to the assumed hardness of Hide-and-Seek, $\mathcal{A}$ cannot recover m from its input and thus makes no query to H that has m as suffix. But then it cannot gather any information on $H(\mathsf{pk}', m)$ for any pk', and thus it will not be able to output $y' = H(\mathsf{pk}', m)$ for any pk'. Formally, we have to make sure that $\mathcal{A}$ gets no information on y' via its input, which is controlled by KGen and $\mathcal{D}$, which may query H on $H(\mathsf{pk}', m)$ for any pk'. This is taken care of in our formal proof below.

Proof. In Fig. 4.14 we define a hybrid sequence reducing the $\mathsf{sNR}^{H,\perp}$ property of $\mathsf{BUFF}[\mathcal{S}, H]$ to the HnS^{H} property of H.

G_0:

1: $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}^{H}$; $m \leftarrow \mathcal{D}^{H}(\mathsf{sk})$
2: $(\mathsf{pk}', y') \leftarrow \mathcal{B}^{H}(\mathsf{sk}, y, \mathsf{aux}(\mathsf{sk}, m))$
3: **return** $H(\mathsf{pk}', m) = y' \ \wedge \ \mathsf{pk}' \neq \mathsf{pk}$

$\mathcal{B}^{H}(\mathsf{sk}, y, h)$:

1: $\sigma \leftarrow \mathsf{Sign}^{H}(\mathsf{sk}, y)$
2: $(\mathsf{pk}', y') \leftarrow \mathcal{A}^{H}(\mathsf{sk}, (\sigma, y), h)$
3: **return** (pk', y')

G_1:

1: $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}^{H}$; $m \leftarrow \mathcal{D}^{H}(\mathsf{sk})$
2: $(\mathsf{pk}', y') \leftarrow \mathcal{B}^{H[(\cdot, m) \mapsto \perp]}(\mathsf{sk}, H(m, \mathsf{pk}), \mathsf{aux}(\mathsf{sk}, m))$
3: **return** $H(\mathsf{pk}', m) = y' \ \wedge \ \mathsf{pk}' \neq \mathsf{pk}$

G_2:

1: $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}^{H}$; $m \leftarrow \mathcal{D}^{H}(\mathsf{sk})$
2: **abort** if KGen queried $H(m, \cdot)$
3: $(\mathsf{pk}', y') \leftarrow \mathcal{B}^{H[(\cdot, m) \mapsto \perp]}(\mathsf{sk}, H(m, \mathsf{pk}), \mathsf{aux}(\mathsf{sk}, m))$
4: **return** $H(\mathsf{pk}', m) = y' \ \wedge \ \mathsf{pk}' \neq \mathsf{pk}$

G_3^i:

1: $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}^{H}$; $m \leftarrow \mathcal{D}^{H}(\mathsf{sk})$
2: **abort** if KGen queried $H(m, \cdot)$
3: $(\mathsf{pk}', y') \leftarrow \mathcal{B}^{H[(\cdot, m) \mapsto \perp]}(\mathsf{sk}, H(m, \mathsf{pk}), \mathsf{aux}(\mathsf{sk}, m))$
4: **return** $H(\mathsf{pk}', m) = y' \ \wedge \ \mathsf{pk}' \neq \mathsf{pk} \ \wedge (\mathsf{pk}', m) = (\mathsf{pk}_i, m_i)$
5: {where (pk_i, m_i) is $\mathcal{D}$'s ith query.}

G_4^i:

1: $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}^{H}$; $m \leftarrow \mathcal{D}^{H}(\mathsf{sk})$
2: **abort** if KGen queried $H(\cdot, m_i)$
3: $(\mathsf{pk}', y') \leftarrow \mathcal{B}^{H[(\cdot, m_i) \mapsto \perp]}(\mathsf{sk}, H(\mathsf{pk}, m_i), \mathsf{aux}(\mathsf{sk}, m_i))$
4: **return** $H(\mathsf{pk}_i, m_i) = y' \ \wedge \ \mathsf{pk}_i \neq \mathsf{pk}$
5: {where (pk_i, m_i) is $\mathcal{D}$'s ith query.}

Figure 4.14: Hybrid steps reducing $\mathsf{sNR}^{H,\perp}$ of $\mathsf{BUFF}[\mathcal{S}, H]$ to HnS^{H} of H when $\mathcal{D}$ is classical, i.e., (4.13), (4.12). In the derivations below we drop the parameter k for notational convenience.

To start with, we note that the adversary $(\mathcal{D}, \mathcal{B})$ playing G_0 is identical to $(\mathcal{D}, \mathcal{A})$ playing $\mathsf{sNR}^{H,\perp}_{\mathsf{BUFF}[\mathcal{S},H]}$.

The G_0 *to* G_1 *hop.* The only difference between G_0 and G_1 is whether $\mathcal{B}$ is

given oracle access to the original random oracle H, or the reprogrammed oracle $H[(m,\cdot) \mapsto \perp]$ and replies with $\perp$ to any query that has suffix m.

Consider the hider $\bar{\mathcal{D}}$ and seeker $\bar{\mathcal{A}}$, where $\bar{\mathcal{D}}$ samples $(\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{KGen}^H$ and $m \leftarrow \mathcal{D}^H(\mathsf{sk})$ and returns

$$x := (\mathsf{pk}, m) \qquad \text{and} \qquad z := (\mathsf{sk}, \mathsf{aux}(\mathsf{sk}, m))\,,$$

and on input $H(x) = H(m,\mathsf{pk})$ and z, the seeker $\bar{\mathcal{A}}$ samples a random index $i \leftarrow [q_A + q_S]$, runs

$$\mathcal{B}^H(\mathsf{sk}, H(m,\mathsf{pk}), \mathsf{aux}(\mathsf{sk},m)) = \mathcal{A}^H\big(\mathsf{sk}, \big(H(m,\mathsf{pk}), \mathsf{Sign}^H(\mathsf{sk},y)\big), \mathsf{aux}(\mathsf{sk},m)\big)$$

internally, but then looks at / does a full measurement of the ith query to obtain (pk_i^*, m_i^*), and returns (pk, m_i^*). It is clear by construction that $z \in \mathcal{SK} \times \mathcal{AUX}$, and (4.11) immediately follows from the fact that pk can be derived from sk, and so

$$\mathsf{H}_\infty(x \mid H, z) = \mathsf{H}_\infty(\mathsf{pk}, m \mid H, \mathsf{sk}, \mathsf{aux}(\mathsf{sk},m)) = \mathsf{H}_\infty(m \mid H, \mathsf{sk}, \mathsf{aux}(\mathsf{sk},m)) \tag{4.14}$$

as claimed. It also follows from construction that $\bar{\mathcal{D}}$ and $\bar{\mathcal{A}}$ preserve the efficiency of $\mathcal{D}$ and $\mathcal{A}$. In terms of query complexity, $\bar{\mathcal{A}}$ makes at most $q_A + q_S$ queries to H.

In the case where $\mathcal{A}$ makes classical queries, there is no difference in the two games when $\mathcal{B}$ makes no query to a point where the two oracles differ, and thus

$$\begin{aligned}\Pr[1 \leftarrow \mathsf{G}_0] &\le \Pr[1 \leftarrow \mathsf{G}_1] + \Pr\left[\exists\, i \in [q_A+q_S] \text{ s.t. } m_i^* = m\right] \\ &\le \Pr[1 \leftarrow \mathsf{G}_1] + (q_A+q_S)\cdot \mathbf{Adv}^{\mathsf{HnS}^H}(\bar{\mathcal{D}}, \bar{\mathcal{A}})\,.\end{aligned}$$

In the quantum case, the same kind of guarantee follows from the O2H lemma [AHU19, Theorem 3], which gives us that

$$\begin{aligned}\Pr[1 \leftarrow \mathsf{G}_0] &\le \Pr[1 \leftarrow \mathsf{G}_1] + 2(q_A+q_S)\cdot \sqrt{\Pr[m_i^* = m]} \\ &\le \Pr[1 \leftarrow \mathsf{G}_1] + 2(q_A+q_S)\cdot \sqrt{\mathbf{Adv}^{\mathsf{HnS}^H}(\bar{\mathcal{D}}, \bar{\mathcal{A}})}\,.\end{aligned}$$

The G_1 to G_2 hop. The difference between G_1 and G_2 is that the latter aborts if KGen^H ever makes a query of the form $(\cdot, m)$. Given that m is produced given (H,sk) but independent of KGen's q_K queries conditioned on (H,sk), and m satisfies (4.9) for $h := \log(1/\epsilon)$, we have

$$\Pr[1 \leftarrow \mathsf{G}_1] \le \Pr[1 \leftarrow \mathsf{G}_2] + \Pr[\mathsf{G}_2 \text{ abort}] \le \Pr[1 \leftarrow \mathsf{G}_2] + q_K\epsilon\,.$$

The G_2 *to* G_3^i *hop.* Assume without loss of generality that $\mathcal{D}$ never repeats its queries $(k_1, m_1), \ldots, (k_{q_D}, m_{q_D})$. Note that the queries of $\mathcal{A}$ in G_1 are blocked at $(\cdot, m)$, and the game aborts if KGen ever queries $(\cdot, m)$. Since, conditioned on KGen not querying with $(\cdot, m)$, $k' \neq k$ and $(k', m) \neq (k_i, m_i)$ for all i, the output $H(k', m)$ is uniformly random and independent of $\mathcal{A}$'s input $(\mathsf{sk}, H(m, \mathsf{pk}), \mathsf{aux}(\mathsf{sk}, m))$ together with the oracle $H[(m, \cdot) \mapsto \perp]$ it has access to, we have

$$\begin{aligned}\Pr\left[1 \leftarrow \mathsf{G}_2\right] &\leq \Pr\left[\begin{array}{c}\exists i \in [q_D] \text{ s.t. } (k', m) = (k_i, m_i) \\ 1 \leftarrow \mathsf{G}_2\end{array}\right] \\ &\quad + \Pr\left[1 \leftarrow \mathsf{G}_2 \;\middle|\; \begin{array}{c}\mathsf{KGen} \text{ not querying } (\cdot, m) \\ (k', m) \neq (k_i, m_i)\ \forall i \in [q_D] \\ k' \neq k\end{array}\right] \\ &\leq \sum_{i \in [q_D]} \Pr\left[1 \leftarrow \mathsf{G}_3^i\right] + 1/|\mathcal{Y}| \,.\end{aligned}$$

The G_3^i *to* G_4^i *hop.* Because of the extra condition $(\mathsf{pk}', m') = (\mathsf{pk}_i, m_i)$ in G_3^i, replacing pk' with pk_i and m' with m_i as in G_4^i, does not change the winning probability. We further drop the condition $(\mathsf{pk}', m') = (\mathsf{pk}_i, m_i)$, which does not decrease the winning probability.

Finally, it remains to upper bound the winning probability of G_4^i for each $i \in [q_D]$. By a lazy sampling argument, we note that conditioned on KGen not querying with $(\cdot, m_i)$ and $\mathsf{pk}_i \neq \mathsf{pk}$, the output $H(k_i, m_i)$ is uniform and independent of $(H[(\cdot, m_i) \mapsto \perp], k, H(k, m_i), \mathsf{aux}(m_i))$, and hence, y' generated by $\mathcal{A}^{H[(\cdot, m_i) \mapsto \perp]}(k, H(k, m_i), \mathsf{aux}(m_i))$ is equal to $H(k_i, m_i)$ with probability at most $1/|\mathcal{Y}|$, i.e.

$$\Pr\left[1 \leftarrow \mathsf{G}_4^i\right] \leq 1/|\mathcal{Y}| \,,$$

which concludes (4.12), (4.13). □

Remark 4.21. *We point out that the claim on* $\bar{\mathcal{D}}$ *and* $\bar{\mathcal{A}}$ *being PPT if* $\mathcal{D}$ *and* $\mathcal{A}$ *are, fails to hold when aiming for a variant of Lemma 4.20 that considers* $\mathsf{NR}^{H,\perp}$ *instead of* $\mathsf{sNR}^{H,\perp}$*. The reason is that, on input* $H(m, \mathsf{pk})$ *and* z*, the seeker* $\bar{\mathcal{A}}$ *needs to run* $\mathcal{A}$ *on* a signature of $H(m, \mathsf{pk})$*, which it can do efficiently if given* sk *(which is part of* z *here, exploiting that* $\mathcal{D}$ *is given* sk*), but not if only given* pk*. This is the reason why in the* computational *setting, treated in Section 4.8, our proof for showing that BUFF satisfies* $\mathsf{sNR}^{H,\perp}$ *does not carry over to* $\mathsf{NR}^{H,\perp}$ *(in line with the counter example given in Section 4.4.4).*

4.5.5 Hide-and-Seek for Random Oracles

The following provides a bound on the Hide-and-Seek property of the random oracle.

Theorem 4.22 (The RO satisfies HnS^H)**.** *Let $\mathcal{D} : \{\perp\} \to \mathcal{X} \times \mathcal{Z}$ and $\mathcal{A} : \mathcal{Y} \times \mathcal{Z} \to \mathcal{X}$ be HnS^H-adversaries satisfying (4.10) for some $0 < \epsilon < 1$, where $\mathcal{A}$ makes q queries to H. Then for $\mathcal{A}^H$ making quantum queries in general, we have*

$$\mathbf{Adv}_H^{\mathsf{HnS}^H}(\mathcal{D}, \mathcal{A}) \leq 4(q+1)\big(\log |\mathcal{Z}| + \log(1/\epsilon) + 20q\big)\epsilon + 10\epsilon \tag{4.15}$$

In the case where $\mathcal{A}^H$ is restricted to classical queries, we have

$$\mathbf{Adv}_H^{\mathsf{HnS}^H}(\mathcal{D}, \mathcal{A}) \leq 2(q+1)\big(\log |\mathcal{Z}| + \log(1/\epsilon) + 1\big)\epsilon + \epsilon\,. \tag{4.16}$$

To prove the above statement, below, we show that a Hide-and-Seek seeker for any fixed hash function $\mathcal{F} : \mathcal{X} \to \mathcal{Y}$ is a successful adversary that can invert multiple hashes simultaneously.

Lemma 4.23. *Let $\mathcal{F} : \mathcal{X} \to \mathcal{Y}$ be a fixed function, $\mathcal{D} : \{\perp\} \to \mathcal{X} \times \mathcal{Z}$ and $\mathcal{A}$ be any Hide-and-Seek adversaries against $\mathcal{F}$. Then for every $(k, T) \in \mathbb{Z}_{>0} \times \mathbb{R}_{>0}$, for $(x, z) \leftarrow \mathcal{D}$ and for $x_1^u, \ldots, x_k^u \leftarrow \mathcal{Y}$ sampled uniformly and independently, we have*

$$\Pr\left[\, x = \mathcal{A}(\mathcal{F}(x), z) \,\right] \leq T \cdot \mathsf{guess}(x \mid z) + \left(\frac{|\,\mathcal{X}\,|}{T}\right)^k \Pr\left[x_i^u = \mathcal{A}(\mathcal{F}(x_i^u), z)\ \forall i \in [k]\right].$$

Proof. Here, for any $z^\circ \in \mathcal{Z}$, we define the following "weighted set"

$$S_{z^\circ}^* := \left\{ \big(x^\circ, w_{z^\circ}(x^\circ)\big) \;\middle|\; x^\circ \in \mathcal{X} \right\},$$

where each element x° comes with a weight, given by

$$w_{z^\circ}(x^\circ) := \Pr\big[x^\circ = \mathcal{A}(\mathcal{F}(x^\circ), z^\circ)\big]\,.$$

The total weight of $S_{z^\circ}^*$ is defined as $W(S_{z^\circ}^*) := \sum_{x^\circ} w_{z^\circ}(x^\circ)$.

Here, we consider the guesser $\mathcal{G}$ that, on input z, chooses its guess $\hat{x}$ by picking it from $\mathcal{X}$ according to the renormalized weights, i.e., according to the distribution

$$p_z(\hat{x}) := \frac{w_z(\hat{x})}{W(S_z^*)}\,.$$

First, we note that

$$\begin{aligned}\Pr[\hat{x} = x] &\geq \Pr[\hat{x} = x \wedge W(S_z^*) \leq T] \\ &\geq \frac{1}{T} \Pr[\mathcal{A}(\mathcal{F}(x), z) = x \wedge W(S_z^*) \leq T] \\ &\geq \frac{1}{T} \big(\Pr[\mathcal{A}(\mathcal{F}(x), z) = x] - \Pr[W(S_z^*) > T] \big) ,\end{aligned}$$

where here, for the second inequality, we exploit that for any fixed choices of x and z, if $W(S_z^*) \leq T$ then $\Pr[\hat{x} = x] = p_z(x) \geq w_z(x)/T$, and so the inequality is obtained by averaging over these choices. Rearranging the terms, we have

$$\mathbf{Adv}^{\mathsf{HnS}^H}(\mathcal{D}, \mathcal{A}) \leq T \cdot \mathsf{guess}(x \mid z) + \Pr[W(S_z^*) > T] . \tag{4.17}$$

Observe that, for every $k \in \mathbb{Z}_{>0}$ and for $x^u, x_1^u, \ldots, x_k^u \leftarrow \mathcal{X}$ sampled uniformly and independently,

$$\left(\frac{W(S_{z^\circ}^*)}{|\mathcal{X}|} \right)^k = \left(\Pr\left[x^u = \mathcal{A}(\mathcal{F}(x^u), z^\circ) \right] \right)^k = \Pr\left[x_i^u = \mathcal{A}(\mathcal{F}(x_i^u), z^\circ) \; \forall i \in [k] \right] . \tag{4.18}$$

Hence, by averaging over $z^\circ \sim z$, for every $T \in \mathbb{R}_{>0}$, we immediately obtain

$$\Pr\left[W(S_z^*) > T \right] \leq \frac{\mathbb{E}\left[|\, W(S_z^*) \,|^k \right]}{T^k} \leq \frac{\Pr\left[x_i^u = \mathcal{A}(\mathcal{F}(x_i^u), z) \; \forall i \in [k] \right]}{T^k} ,$$

where the first inequality follows Markov's bound. Plugging the above back to (4.17), we conclude the proof. □

Lemma 4.24. *Let $\mathcal{F} : \mathcal{X} \to \mathcal{Y}$ be a fixed function, and $\mathcal{A}$ be any algorithm. Then, for $(x^u, y^u) \leftarrow \mathcal{X} \times \mathcal{Y}$ sampled uniformly,*

$$\Pr\left[x^u = \mathcal{A}(\mathcal{F}(x^u)) \right] \leq \frac{|\, \mathcal{Y} \,| \cdot \Pr\left[y^u = \mathcal{F}\left(\mathcal{A}(y^u)\right) \right]}{|\, \mathcal{X} \,|} .$$

Proof. Without loss of generality, let $\mathcal{A}$ be deterministic. Observe that, by symmetry, for every $y^\circ \in \mathcal{F}(\mathcal{X}) \subseteq \mathcal{Y}$,

$$\begin{aligned}\Pr\left[x^u = \mathcal{A}(y^\circ) \;\middle|\; \mathcal{F}(x^u) = y^\circ \right] &= \begin{cases} 1/\left|\, \mathcal{F}^{-1}(y^\circ) \,\right| & \text{if } \mathcal{F}(\mathcal{A}(y^\circ)) = y^\circ \\ 0 & \text{otherwise} \end{cases} \\ &= \frac{\mathbb{1}_{\mathcal{F}(\mathcal{A}(y^\circ)) = y^\circ}}{|\, \mathcal{F}^{-1}(y^\circ) \,|} .\end{aligned}$$

Averaging both sides over $y^\circ \sim \mathcal{F}(x^u)$, we obtain

$$\begin{aligned}
\Pr\left[\, x^u = \mathcal{A}(\mathcal{F}(x^u)) \,\right] &= \sum_{y^\circ \in \mathcal{F}(\mathcal{X})} \Pr\left[\, \mathcal{F}(x^u) = y^\circ \,\right] \cdot \frac{\mathbb{1}_{\mathcal{F}(\mathcal{A}(y^\circ))=y^\circ}}{|\, \mathcal{F}^{-1}(y^\circ) \,|} \\
&= \sum_{y^\circ \in \mathcal{F}(\mathcal{X})} \frac{\left|\, \mathcal{F}^{-1}(y^\circ) \,\right|}{|\, \mathcal{X} \,|} \cdot \frac{\mathbb{1}_{\mathcal{F}(\mathcal{A}(y^\circ))=y^\circ}}{|\, \mathcal{F}^{-1}(y^\circ) \,|} \\
&\leq \frac{|\, \mathcal{Y} \,|}{|\, \mathcal{X} \,|} \sum_{y^\circ \in \mathcal{Y}} \frac{\mathbb{1}_{\mathcal{F}(\mathcal{A}(y^\circ))=y^\circ}}{|\, \mathcal{Y} \,|} \\
&= \frac{|\, \mathcal{Y} \,| \cdot \Pr\left[y^u = \mathcal{F}\left(\mathcal{A}(y^u)\right)\right]}{|\, \mathcal{X} \,|} \,.
\end{aligned}$$

This concludes the proof. □

Substituting $\mathcal{F}$ with $\mathcal{F}^k(x_1, \ldots, x_k) := \big(\mathcal{F}(x_1), \ldots, \mathcal{F}(x_k)\big)$, and $\mathcal{A}$ with $(y_1, \ldots, y_k) \mapsto \big(\mathcal{A}(y_1, z), \ldots, \mathcal{A}(y_k, z)\big)$ for z as in the right-hand-side of Lemma 4.23, we immediately obtain the following.

Corollary 4.25. *Let $\mathcal{F} : \mathcal{X} \to \mathcal{Y}$ be a fixed function, $\mathcal{D} : \{\bot\} \to \mathcal{X} \times \mathcal{Z}$ and $\mathcal{A}$ be any Hide-and-Seek adversaries against $\mathcal{F}$. Then for every $(k, T) \in \mathbb{Z}_{>0} \times \mathbb{R}_{>0}$, for $(x, z) \leftarrow \mathcal{D}$ and for $y_1^u, \ldots, y_k^u \leftarrow \mathcal{Y}$ sampled uniformly and independently, we have*

$$\Pr\left[\, x = \mathcal{A}(\mathcal{F}(x), z) \,\right] \leq T \cdot \mathsf{guess}(x \mid z) + \left(\frac{|\, \mathcal{Y} \,|}{T}\right)^k \Pr\left[y_i^u = \mathcal{F}\big(\mathcal{A}(y_i^u, z)\big)\ \forall i \in [k]\right] .$$

Finally, it suffices to bound the probability of simultaneously finding RO preimages of multiple uniformly and independently chosen target. We prove it for the classical case here, and refer to [CGLQ20] for the quantum case.[14]

Lemma 4.26. *Let $\mathcal{A}^H$ be an oracle algorithm making at most q classical queries to H. Then for $y_1^u, \ldots, y_k^u \leftarrow \mathcal{Y}$ sampled uniformly and independently, we have*

$$\Pr\left[H\left(\mathcal{A}^H(y_i^u)\right) = y_i^u\ \forall i \in [k]\right] \leq \big(k(q+1)/|\, \mathcal{Y} \,|\big)^k .$$

Proof. Let $\mathcal{A}$ be deterministic, and $\mathcal{B}^H(y_1^u, \ldots, y_k^u)$ be running every instance of $x_i \leftarrow \mathcal{A}^H(y_i^u)$, making one more query per instance to check if the produced output is valid $H(x_i) \stackrel{?}{=} y_i^u$ and output a bit that is 1 if and only if all checks are passed. Then, of course

$$\Pr\left[H\left(\mathcal{A}^H(y_i^u)\right) = y_i^u\ \forall i \in [k]\right] = \Pr\left[\mathcal{B}^H(y_1^u, \ldots, y_k^u) = 1\right] .$$

[14] Strictly speaking, since we cannot fully extract and verify the bound as in [CGLQ20], we use a bound that we obtain by using a similar approach.

Toward bounding the right-hand side, it suffices to consider the case where the domain of H is of size at least $k(q+1)$ and $\mathcal{A}$ makes exactly $k(q+1)$ distinct queries $x_1, \ldots, x_{k(q+1)}$ to H. Observe that $y_1^u, \ldots, y_k^u, H(x_1), \ldots, H(x_{k(q+1)})$ are mutually independent, and hence we have

$$\begin{aligned}
&\Pr[\mathcal{B}^H(y_1^u, \ldots, y_k^u) = 1] \leq \Pr\left[y_i^u \in \{H(x_1), \ldots, H(x_{k(q+1)})\}\ \forall i \in [k]\right] \\
&= \mathop{\mathbb{E}}_{\substack{y_i^\circ \sim H(x_i) \\ \forall i \in [k]}} \left[\Pr\left[y_i^u \in \{y_1^\circ, \ldots, y_{k(q+1)}^\circ\}\ \forall i \in [k] \;\middle|\; H(x_j) = y_j^\circ\ \forall j \in [k(q+1)]\right]\right] \\
&= \mathop{\mathbb{E}}_{\substack{y_i^\circ \sim H(x_i) \\ \forall i \in [k]}} \left[\prod_{i \in [k]} \Pr\left[y_i^u \in \{y_1^\circ, \ldots, y_{k(q+1)}^\circ\} \;\middle|\; H(x_j) = y_j^\circ\ \forall j \in [k(q+1)]\right]\right] \\
&\leq \mathop{\mathbb{E}}_{\substack{y_i^\circ \sim H(x_i) \\ \forall i \in [k]}} \left[\prod_{i \in [k]} k(q+1)/|\,\mathcal{Y}\,|\right] \leq \left(k(q+1)/|\,\mathcal{Y}\,|\right)^k .
\end{aligned}$$

□

Lemma 4.27 (A concrete version of [CGLQ20, Lemma 5.6])**.** *Let $\mathcal{A}^H$ be an oracle algorithm making at most q quantum queries to H. Then for $y_1^u, \ldots, y_k^u \leftarrow \mathcal{Y}$ sampled uniformly and independently, we have*

$$\Pr\left[H\left(\mathcal{A}^H(y_i^u)\right) = y_i^u\ \forall i \in [k]\right] \leq \left(\frac{2k(q+1) + 40q^2 + 4}{|\,\mathcal{Y}\,|}\right)^k .$$

Now we are ready to wrap up the proof of Theorem 4.22

Proof of Theorem 4.22. For every fixed choice H° of H, evoke Corollary 4.25 and we obtain

$$\begin{aligned}
&\Pr\left[x = \mathcal{A}^{H^\circ}(\mathcal{F}^{H^\circ}(x), z) \;\middle|\; H = H^\circ\right] \\
&\leq T \cdot \mathsf{guess}(x \mid z, H = H^\circ) + \left(\frac{|\,\mathcal{Y}\,|}{T}\right)^k \Pr\left[y_i^u = \mathcal{A}^{H^\circ}(y_i^u, z)\ \forall i \in [k] \;\middle|\; H = H^\circ\right] \\
&\leq T \cdot \mathsf{guess}(x \mid z, H = H^\circ) + \left(\frac{|\,\mathcal{Y}\,|}{T}\right)^k \sum_{z^\circ \in \mathcal{Z}} \Pr\left[y_i^u = \mathcal{A}^{H^\circ}(y_i^u, z^\circ)\ \forall i \in [k] \;\middle|\; H = H^\circ\right] .
\end{aligned}$$

Note that $\mathsf{guess}(x \mid z, H) \leq \epsilon$; averaging the above over $H^\circ \sim H$, we thus obtain

$$\Pr\left[\, x = \mathcal{A}^H(\mathcal{F}^H(x), z)\,\right] \leq T\epsilon + \left(\frac{|\,\mathcal{Y}\,|}{T}\right)^k \sum_{z^\circ \in \mathcal{Z}} \Pr\left[y_i^u = \mathcal{A}^H(y_i^u, z^\circ)\ \forall i \in [k]\right] \tag{4.19}$$

In the case where $\mathcal{A}$ makes classical queries only, substitute $\mathcal{A}$ as in Lemma 4.26 with $\mathcal{A}(\cdot, z^\circ)$ in (4.19); we obtain

$$\Pr\big[\, x = \mathcal{A}^H(\mathcal{F}^H(x), z)\,\big] \le T\epsilon + |\,\mathcal{Z}\,|\left(\frac{k(q+1)}{T}\right)^k .$$

Plugging in $T = 2k(q+1)$ and $k = \lceil \log|\,\mathcal{Z}\,| + \log(1/\epsilon)\rceil$, the above is bounded by

$$\le T\epsilon + |\,\mathcal{Z}\,| \cdot 2^{-k} \le T\epsilon + \epsilon \le 2\big(\log|\,\mathcal{Z}\,| + \log(1/\epsilon) + 1\big)(q+1) + \epsilon .$$

Similarly, in the case where $\mathcal{A}$ makes quantum queries in general, substitute $\mathcal{A}$ as in Lemma 4.27 with $\mathcal{A}(\cdot, z^\circ)$ in (4.19); we obtain

$$\Pr\big[\, x = \mathcal{A}^H(\mathcal{F}^H(x), z)\,\big] \le T\epsilon + |\,\mathcal{Z}\,|\left(\frac{2k(q+1) + 40q^2 + 4}{T}\right)^k .$$

Plugging in $T = 2(2k(q+1) + 40q^2 + 4)$ and $k = \lceil \log|\,\mathcal{Z}\,| + \log(1/\epsilon)\rceil$ and simplifying the bound, the above is bounded by

$$\le T\epsilon + |\,\mathcal{Z}\,|\,2^{-k} \le T\epsilon + \epsilon \le 4(q+1)\big(\log|\,\mathcal{Z}\,| + \log(1/\epsilon) + 20q\big)\epsilon + 10\epsilon$$

This concludes the proof. □

4.5.6 Wrapping up the Proof of Theorem 4.18

Theorem 4.18. *Let $\mathcal{D}^H : \mathcal{SK} \to \mathcal{M}$ and $\mathcal{A}^H : \mathcal{SK} \times \mathcal{SGN} \times \mathcal{AUX} \to \mathcal{PK} \times \mathcal{SGN}$ be $\mathsf{sNR}^{H,\perp}$-adversaries against $\mathsf{BUFF}[\mathcal{S}, H]$ for some $\mathsf{aux} : \mathcal{SK} \times \mathcal{M} \to \mathcal{AUX}$, making at most q_D and q_A queries to H, respectively, where (4.9) is satisfied for h such that $0 < \epsilon := 2^{-h} \le \frac{1}{2}$. Then*

$$\mathbf{Adv}^{\mathsf{sNR}^{H,\perp}}_{\mathsf{BUFF}[\mathcal{S},H]}(\mathcal{D}, \mathcal{A}, \mathsf{aux}) \le 6(q_A + q_S + 1)^2 \log\left(\frac{|\mathcal{SK}| \cdot |\mathcal{AUX}|}{\epsilon}\right)\epsilon + q_K\epsilon + \frac{q_D + 1}{|\mathcal{Y}|}$$

and in the case where $\mathcal{A}$ makes quantum queries, we have $\mathbf{Adv}^{\mathsf{sNR}^{H,\perp}}_{\mathsf{BUFF}[\mathcal{S},H]}(\mathcal{D}, \mathcal{A}, \mathsf{aux})$

$$\le 18\sqrt{\left(\log\frac{|\mathcal{SK}| \cdot |\mathcal{AUX}|}{\epsilon} + q_{\mathcal{A}} + q_S\right)(q_{\mathcal{A}} + q_S + 1)^3\,\epsilon} + q_K\epsilon + \frac{q_D + 1}{|\mathcal{Y}|} .$$

Proof. In the case where $\mathcal{A}$ is restricted to classical queries, we combine (4.12)

and (4.16), simplify the bound, and get

$$\begin{aligned}
&\mathbf{Adv}^{\mathsf{sNR}^{H,\perp}}_{\mathsf{BUFF}[\mathcal{S},H]}(\mathcal{D},\mathcal{A},\mathsf{aux}) \\
&\leq (q_A+q_S)\cdot\mathbf{Adv}^{\mathsf{HnS}^H}_{H}(\bar{\mathcal{D}},\bar{\mathcal{A}})+q_K\epsilon+\frac{q_D+1}{|\mathcal{Y}|} \\
&\leq 6(q_A+q_S+1)^2\log\left(\frac{|\mathcal{SK}|\cdot|\mathcal{AUX}|}{\epsilon}\right)\epsilon+q_K\epsilon+\frac{q_D+1}{|\mathcal{Y}|}
\end{aligned}$$

where $\bar{\mathcal{D}}$ and $\bar{\mathcal{A}}$ are as specified in Lemma 4.20. Similarly, in the case where $\mathcal{A}$ is allowed to make quantum queries, we combine (4.13) and (4.15), simplify the bound, and get

$$\begin{aligned}
&\mathbf{Adv}^{\mathsf{sNR}^{H,\perp}}_{\mathsf{BUFF}[\mathcal{S},H]}(\mathcal{D},\mathcal{A},\mathsf{aux}) \\
&\leq 2(q_A+q_S)\cdot\sqrt{\mathbf{Adv}^{\mathsf{HnS}^H}(\bar{\mathcal{D}},\bar{\mathcal{A}})}+q_K\epsilon+\frac{q_D+1}{|\mathcal{Y}|} \\
&\leq 18\sqrt{\left(\log\frac{|\mathcal{SK}|\cdot|\mathcal{AUX}|}{\epsilon}+q_{\mathcal{A}}+q_S\right)(q_{\mathcal{A}}+q_S+1)^3\epsilon}+q_K\epsilon+\frac{q_D+1}{|\mathcal{Y}|}
\end{aligned}$$

where $\bar{\mathcal{D}}$ and $\bar{\mathcal{A}}$ are as specified in Lemma 4.20. □

4.6 BUFF via Iterative Hash Functions

4.6.1 sNR Relative to (Function-like) Oracles

We recall the *secret-key non-resignability (sNR)* defined in Section 4.5.1, slightly relaxed by not restricting to the random oracle model, but instead considering an oracle O that is to be instantiated with a function chosen according to an *arbitrary* distribution. The relevant examples are when $O = H$ is a uniformly random function with a given domain and range, for capturing the random oracle model, or $O = P^{\pm 1} : \{-1,1\}\times\mathcal{Y}\to\mathcal{Y}$, which maps $(d,x)\mapsto P^d(x)$ for a random permutation $P\leftarrow\mathsf{Sym}(\mathcal{Y})$, capturing the random-permutation model then. We may then write $\mathcal{A}^O$ in order to make the oracle access to O explicit.

Let $\mathcal{S} = (\mathsf{KGen}^O, \mathsf{Sign}^O, \mathsf{Vrfy}^O)$ be a signature scheme where the underlying algorithms are given query access to O. The security game $\mathsf{sNR}^{O,\perp}$ depicted in Fig. 4.15 is played by two oracle algorithms

$$\mathcal{D}^O : \mathcal{SK}\to\mathcal{M}\ ,\ \text{and}\quad \mathcal{A}^H : \mathcal{SK}\times\mathcal{SGN}\times\mathcal{AUX}\to\mathcal{PK}\times\mathcal{SGN}$$

each with bounded queries to O, and an arbitrary function $\mathsf{aux} : \mathcal{SK}\times\mathcal{M}\to\mathcal{AUX}$. It is crucial here that aux has no access to O; otherwise, the strong negative result from [DFHS24] applies.

$\mathsf{sNR}^{O,\perp}_{\mathcal{S}}(\mathcal{D},\mathcal{A},\mathsf{aux})$:
1: $(\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{KGen}^O$
2: $m \leftarrow \mathcal{D}^O(\mathsf{sk})$
3: $(\mathsf{pk}',\sigma') \leftarrow \mathcal{A}^O\left(\mathsf{sk}, \mathsf{Sign}^O(\mathsf{sk},m), \mathsf{aux}(m,\mathsf{sk})\right)$
4: **return** $\mathsf{pk}' \neq \mathsf{pk} \ \wedge \ \mathsf{Vrfy}^O(\mathsf{pk}',m,\sigma')$

Figure 4.15: The game $\mathsf{sNR}^{O,\perp}$.

In addition, for the game to be non-trivial, we have the entropy requirement

$$\mathsf{H}_\infty(m \mid \mathsf{sk}, \mathsf{aux}(m,\mathsf{sk}), O) \geq \log(1/\epsilon) \tag{4.20}$$

for some small $\epsilon > 0$ is satisfied. We informally say that the signature scheme $\mathcal{S}$ satisfies $\mathsf{sNR}^{O,\perp}$ if for every such $\mathcal{D},\mathcal{A},\mathsf{aux}$

$$\mathbf{Adv}^{\mathsf{sNR}^{O,\perp}}_{\mathcal{S}}(\mathcal{D},\mathcal{A},\mathsf{aux}) := \Pr\left[\mathsf{sNR}^{O,\perp}_{\mathcal{S}}(\mathcal{D},\mathcal{A},\mathsf{aux}) = 1\right]$$

is small. Later in Section 4.8, we will consider a computational variant of the entropy condition (4.20).

4.6.2 Iterative Hash Functions

Here and for the remainder, we set $\mathcal{X} := \{0,1\}^r$ and $\mathcal{Y} := \{0,1\}^{n_f}$ for parameters r and n_f, and let $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{Y}$ be a function, referred to as the *round function*. An element $x \in \mathcal{X}$ is called a *block*, and a bit string in $x \in \{0,1\}^{rB}$ is said to have *block length* B. Clearly, a bit string x with block length B can be naturally parsed as $(x_1,\ldots,x_B) \in (\{0,1\}^r)^B$. Finally, we write $\mathcal{X}^{\leq B}$ for the set of non-empty strings with block length at most B.
For a bit string s, define

$$|\, s \,|_{\mathrm{bl}} := \lceil\, |\, s \,| \,/\, r \,\rceil,$$

as the number of blocks of length r that s takes up, rounded up when the last block is not full.

The following captures the general notion of an iterative hash function, to which our negative result applies.

Definition 4.28. *An* iterative hash function $\mathcal{F}\colon \{0,1\}^* \to \{0,1\}^{n_{\mathcal{F}}}$ *that is specified by a round function* $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{Y}$*, an initialization vector* $\mathsf{IV} \in \mathcal{Y}$*, a padding function* $\mathsf{pad} : \{0,1\}^* \to \{0,1\}^{\leq 2r}$ *with the property that* $|x|+|pad(x)|$ *is a multiple of* r *for any* $x \in \{0,1\}^*$*, and (optionally) a post-processing function* $p : \mathcal{Y} \to \{0,1\}^{n_{\mathcal{F}}}$*.* $\mathcal{F}$ *is then defined to map* $x \in \{0,1\}^*$ *to* $\mathcal{F}(x) = p(z_B)$*, where*

z_i is iteratively given by

$$z_i = f(x_i, z_{i-1}) \qquad \text{and} \qquad z_0 \coloneqq \mathsf{IV}$$

and the x_i's are obtained by parsing the string $x \| \mathsf{pad}(x) \in \{0,1\}^{rB}$ naturally as $(x_1, \ldots, x_B) \in (\{0,1\}^r)^B$.

Remark 4.29. *We allow the round function f and the post-processing function p to be given as oracle algorithms, with access to an oracle O (for instance the random oracle, or a random permutation); we will then write f^O and p^O, as well as $\mathcal{F}^O$, to make this explicit. This does not apply to* pad *and* IV*.*

Instantiating the round function f by a cryptographic compression function (which is then typically modelled as a random oracle H), and a suitable padding function pad, we obtain the the Merkle-Damgård construction, which we will be referring to as $\mathsf{MD}^H_{\mathsf{pad}} : \{0,1\}^* \to \mathcal{Y}$ for short.

If we instantiate the round function as $f(x, y) = P\big((x \| 0^c) \oplus y\big)$ for $c = n_f - r$, where $P \in \mathsf{Sym}(\mathcal{Y})$ is a cryptographic permutation (typically modelled as a random permutation), and take a suitable padding function pad (that appends $10 \ldots 01$ with the appropriate number of 0s) and an appropriate post-processing, we recover the Sponge construction $\mathsf{SPNG}^{P^{\pm 1}}_{\mathsf{pad}} : \{0,1\}^* \to \{0,1\}^{n_{\mathsf{SPNG}}}$ with rate r and capacity c. We are considering the Sponge hash function with one squeezing round; this means that $p(z)$ merely outputs the first n_{SPNG} bits of z where for technical reasons we require that $n_{\mathsf{SPNG}} \leq c$.

In the case where no padding is performed and instead the domain is restricted an exact multiple of blocks, we denote the corresponding padding-free variants as $\mathsf{MD}^H_{\perp} : \mathcal{X}^* \to \mathcal{Y}$ and $\mathsf{SPNG}^{P^{\pm 1}}_{\perp} : \mathcal{X}^* \to \{0,1\}^{n_{\mathsf{SPNG}}}$ respectively.

4.6.3 BUFF via Iterative Hash Functions is not sNR

Let $\mathcal{F}^O$ be an iterative hash function, as in Definition 4.28, where the round function f^O has access to an oracle O (which is instantiated by a function chosen according to a given distribution, e.g. the random oracle). Furthermore, let $\mathcal{S}^O = (\mathsf{KGen}, \mathsf{Sign}^O, \mathsf{Vrfy}^O)$ be a signature scheme, where signing and verifying (but not KGen) may also have query access to O.

Proposition 4.30. *There are $\mathsf{sNR}^{O,\perp}$ adversaries $\mathcal{D}^O, \mathcal{A}^O$ and a function* aux *such that for every choice of the security parameter $\lambda \in \mathbb{Z}_{>0}$ we have*

$$\mathsf{H}_\infty\limits_{\substack{(\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{KGen}^O \\ m \leftarrow \mathcal{D}^O}} \Big(m \;\Big|\; \mathsf{sk}, \mathsf{aux}(m, \mathsf{sk}), O \Big) \geq (\lambda - 2) \cdot r - n_f \; . \tag{4.21}$$

yet

$$\mathbf{Adv}^{\mathsf{sNR}^{O,\perp}}_{\mathsf{BUFF}[\mathcal{S},\mathcal{F}]}(\mathcal{D}, \mathcal{A}, \mathsf{aux}) \geq 1 - \mathsf{guess}(\mathsf{pk}) \; ,$$

for $(\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{KGen}$. *Moreover the algorithms* $\mathcal{D}$ *and* $\mathcal{A}$ *make* $Q_{\mathcal{D}}$ *and* $Q_{\mathcal{A}}$ *queries to* O *where*

$$Q_{\mathcal{D}} + Q_{\mathcal{A}} \leq Q_{\mathcal{F}} + Q_S \ ,$$

where Sign *makes at most* Q_S *queries to* O*, and evaluating* $\mathcal{F}^O(m\|\mathsf{pk})$ *for* $(\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{KGen}$ *and* $m \leftarrow \mathcal{D}^O(\mathsf{sk})$ *takes at most* $Q_{\mathcal{F}}$ *queries to* O *in the worst case. If the round function* f *and post-processing* p *of the hash function* $\mathcal{F}$ *are polynomial-time computable, then so is* aux *and* $\mathcal{D},\mathcal{A}$ *are PPT.*

Proof. For simplicity, we give the proof for the case that the padding only depends on the length of the input, i.e. $\mathsf{pad}(x) = \mathsf{pad}(y)$ for all x, y with $|\,x\,| = |\,y\,|$.

We describe the adversaries $\mathcal{D}^O, \mathcal{A}^O$ and the auxiliary function aux. $\mathcal{D}^O$ first samples a message prefix $x = (x_1, \ldots, x_\lambda) \leftarrow \mathcal{X}^\lambda$ and then computes the intermediate digests z_i towards computing the hash of x, i.e., it sets $z_0 = \mathsf{IV}$ and $z_i = f^O(x_i, z_{i-1})$ for $i = 1, \ldots, \lambda$. In the end, it outputs the message $m := x\|z_\lambda$. Furthermore, for any message and secret key, the auxiliary function aux is defined to output the last n_f bits of the message, so that here $\mathsf{aux}(m,\mathsf{sk}) = z_\lambda$, which is sufficient information to compute that hash $\mathcal{F}^O(m\|\mathsf{pk}')$ for any pk'.

Thus, the adversary $\mathcal{A}^O(\mathsf{sk}, \sigma, \mathsf{aux}(m,\mathsf{sk}))$ simply parses $\mathsf{aux}(m,\mathsf{sk}) = z_\lambda$ and samples $(\mathsf{sk}',\mathsf{pk}') \leftarrow \mathsf{KGen}$ and aborts if $\mathsf{pk} = \mathsf{pk}'$. Then it computes the hash $y' = \mathcal{F}^O(m\|\mathsf{pk}')$ in the obvious way: First, it splits $\mathsf{pk}'\|\mathsf{pad}(m\|\mathsf{pk}')$ into blocks of length r.[15] Denote the number of blocks by ℓ and the ith block by b_i. $\mathcal{A}$ then sets $z'_0 = z_\lambda$ and computes $z'_i = f(b_i, z'_{i-1})$ for $i = 1, \ldots, \ell$. It sets $y' = z'_\ell$ (or in case of post-processing $y' = p(z'_\ell)$). Finally, it computes the signature $\sigma' = (\mathcal{S}.\mathsf{Sign}^O(\mathsf{sk}', y'), y')$ and outputs (pk', σ') to the challenger.

For the reader's convenience, we show how the adversaries $\mathcal{D}$ and $\mathcal{A}$ share the computation of the hash in Fig. 4.16.

It is easy to see that $\mathcal{D}^O$ calls the round function f λ times to compute the value z_λ, and $\mathcal{A}$ makes at most $\lceil(|\,\mathsf{pk}\,| + |\,\mathsf{pad}\,|)/r\rceil$ calls the round function and one call to the post-processing p to evaluate the rest of the function.

It is easy to see that all algorithms make at most one call to KGen (expected) and one call to Sign each and thus are efficient.

The min-entropy H_∞ can be computed by

$$\begin{aligned}
&\mathop{\mathsf{H}_\infty}_{\substack{(\mathsf{sk},\mathsf{pk})\leftarrow\mathsf{KGen}^O\\ m\leftarrow\mathcal{D}^O}} \Big(m \;\Big|\; \mathsf{sk}, \mathsf{aux}(m,\mathsf{sk}), O\Big) = \mathop{\mathsf{H}_\infty}_{\substack{(\mathsf{sk},\mathsf{pk})\leftarrow\mathsf{KGen}^O\\ m\leftarrow\mathcal{D}^O}} \big(x\|z_\lambda \big| \mathsf{sk}, z_\lambda, O\big) \\
\geq &\mathop{\mathsf{H}_\infty}_{\substack{(\mathsf{sk},\mathsf{pk})\leftarrow\mathsf{KGen}^O\\ m\leftarrow\mathcal{D}^O}} \big(x\big|O\big) - \big|\,\mathsf{aux}(m,\mathsf{sk})\|\mathsf{pad}(m\|\mathsf{pk}')\,\big| \geq k\cdot r - (n_f + 2r)
\end{aligned}$$

This proves the claim. □

[15] Here we use the assumption that $\mathsf{pad}(m\|\mathsf{pk}')$ can be computed from $|\,m\|\mathsf{pk}'\,|$ which is known to $\mathcal{A}$.

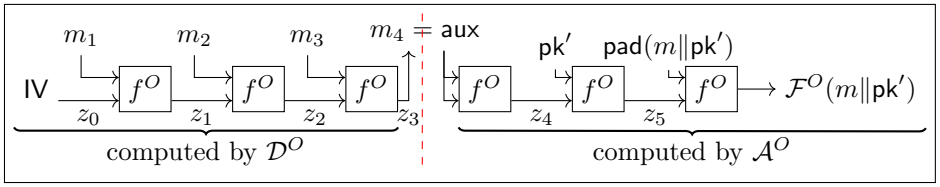


Figure 4.16: A simplified description of our attack for $\lambda = 3$ and without post-processing.

4.7 Sandwich BUFF (sBUFF) Transformation

Motivated by the above attack against the original BUFF transform, we introduce here Sandwich BUFF (sBUFF). The main technical challenge is to show that when instantiated with the Merkle-Damgård hash function, the Sandwich BUFF transform satisfies sNR. Motivated by the approach in [DFH+24], we first reduce the sNR property to the hardness of Hide-and-Seek for the considered hash function, and then we show the hardness of Hide-and-Seek in Sect Section 4.7.3.

For a signature scheme $\mathcal{S} = (\mathsf{KGen}^O, \mathsf{Sign}^O, \mathsf{Vrfy}^O)$ we define the Sandwich BUFF transform of $\mathcal{S}$ with hash function $\mathcal{F}^O$ to be signature scheme $\mathsf{sBUFF}[\mathcal{S}, \mathcal{F}] = (\mathsf{KGen}'^O, \mathsf{Sign}'^O, \mathsf{Vrfy}'^O)$ with algorithms as follows:

KGen'^O:	$\mathsf{Sign}'^O(\mathsf{sk}, m)$:	$\mathsf{Vrfy}'^O(\mathsf{pk}, m, \sigma')$:
1: $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}^O$	1: $y := \mathcal{F}^O(m\|\mathsf{pk}\|m)$	1: $(\sigma, y) := \sigma'$
2: **return** $(\mathsf{sk}, \mathsf{pk})$	2: $\sigma \leftarrow \mathsf{Sign}^O(\mathsf{sk}, y)$	2: **return** $\mathsf{Vrfy}^O(\mathsf{pk}, y, \sigma) \wedge$
	3: **return** (σ, y)	3: $\quad y = \mathcal{F}^O(m\|\mathsf{pk}\|m)$

Figure 4.17: The Sandwich BUFF Transformation

4.7.1 Sandwich BUFF via Merkle-Damgård is sNR

Recall that $\mathcal{X} := \{0,1\}^r$ and $\mathcal{Y} := \{0,1\}^{n_f}$. Towards, the main statement for $\mathsf{sBUFF}[\cdot, \mathsf{MD}]$, let $\mathcal{S} = (\mathsf{KGen}, \mathsf{Sign}^H, \mathsf{Vrfy}^H)$ be a signature scheme in the random oracle model, i.e., with access to a random oracle $H : \mathcal{X} \times \mathcal{Y} \to \mathcal{Y}$ (though, for simplicity, we assume that KGen does not query H), and let $\mathsf{MD}_\perp^H$ be the padding-free Merkle-Damgård hash function with H as the round-function. We allow Sign to make at most $q_\mathcal{S} \in \mathbb{Z}_{>0}$ queries for signing each message. For technical reasons, we restrict the domain of $\mathsf{MD}_\perp^H$ to $\mathcal{X}^{\leq \bar{B}}$ for an arbitrary but fixed bound $\bar{B}$. As a consequence, the message space $\mathcal{M}'$ of $\mathcal{S}' = \mathsf{sBUFF}[\mathcal{S}, \mathsf{MD}]$ is then restricted so that

$$m\|\mathsf{pk}\|m\|\mathsf{pad}(m\|\mathsf{pk}\|m) \in \mathcal{X}^{\leq \bar{B}}$$

for all $m \in \mathcal{M}'$ and $\mathsf{pk} \in \mathcal{PK}' = \mathcal{PK}$. Additionally, we assume (without loss of generality) that any $\mathsf{pk} \in \mathcal{PK}'$ is at least r bits long. We then have the following statement on the sNR property of $\mathcal{S}' = \mathsf{sBUFF}[\mathcal{S}, \mathsf{MD}]$.

Theorem 4.31. *Let $\mathcal{D}^H$ and $\mathcal{A}^H$ be $\mathsf{sNR}^{H,\perp}$-adversaries against $\mathsf{sBUFF}[\mathcal{S}, \mathsf{MD}]$ making at most $q_{\mathcal{D}}$ and $q_{\mathcal{A}} > 0$ classical queries to H respectively such that (4.10) holds for some $0 < \epsilon < 1$; let $\mathsf{aux} : \mathcal{SK} \times \mathcal{M} \to \mathcal{AUX}$ be any (possibly randomized) function. Then for $k := \lceil \log |\, \mathcal{SK} \times \mathcal{AUX} \,| + \log(1/\epsilon) \rceil$ and $q_{\mathcal{B}} := q_{\mathcal{A}} + q_{\mathcal{S}}$ we have*

$$\mathbf{Adv}^{\mathsf{sNR}^{H,\perp}}_{\mathsf{sBUFF}[\mathcal{S},\mathsf{MD}_{\mathsf{pad}}]}(\mathcal{D}, \mathcal{A}) \leq \frac{(q_{\mathcal{D}}^2 \cdot r + 1) \cdot \bar{B}\bar{L} + q_{\mathcal{D}} + 2\bar{L}^2}{|\, \mathcal{Y} \,|} + 6kr^2 q_{\mathcal{B}}(q_{\mathcal{B}} + \bar{B})\epsilon \ ,$$

where $\bar{L} = q_{\mathcal{A}} + q_{\mathcal{D}} + q_{\mathcal{S}} + 2\bar{B}$.

Proof. We put together Lemma 4.32, which reduces sNR of Sandwich BUFF to the harness of Hide-and-Seek, and Theorem 4.37 (with $\mathcal{Z} = \mathcal{SK} \times \mathcal{AUX}$ and $q = q_{\mathcal{A}}$), which shows the hardness of Hide-and-Seek, to obtain the bound. □

4.7.2 Reducing sNR of Sandwich BUFF to Hide-and-Seek

We reduce the sNR property for Sandwich BUFF with MD to the hardness of Hide-and-Seek for MD.

Lemma 4.32. *Let $\mathcal{D}^H$ and $\mathcal{A}^H$ be $\mathsf{sNR}^{H,\perp}$-adversaries against $\mathsf{sBUFF}[\mathcal{S}, \mathsf{MD}]$, making at most $q_{\mathcal{D}}$ and $q_{\mathcal{A}} \in \mathbb{Z}_{>0}$ classical queries to H respectively; let $\mathsf{aux} : \mathcal{SK} \times \mathcal{M} \to \mathcal{AUX}$ be any (possibly randomized) function. Then there exists a hider $\bar{\mathcal{D}} : \{\perp\} \to \mathcal{X}^{\leq \bar{B}} \times \mathcal{Z}$ and a seeker $\bar{\mathcal{A}} : \mathcal{Y} \times \mathcal{Z} \to \mathcal{X}^{\leq \bar{B}}$ and $\mathcal{Z} = \mathcal{SK} \times \mathcal{AUX}$, where $\bar{\mathcal{A}}$ makes at most $q_{\mathcal{B}} := q_{\mathcal{A}} + q_{\mathcal{S}}$ and $\bar{\mathcal{D}}$ makes at most $q_{\mathcal{D}}$ queries to H, and such that*

$$\mathsf{H}^{\infty}_{(x,z) \leftarrow \bar{\mathcal{D}}^H}(x \mid H, z) = \mathsf{H}^{\infty}_{\substack{(\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{KGen}^H \\ m \leftarrow \mathcal{D}^H(\mathsf{sk})}}(m \mid H, \mathsf{sk}, \mathsf{aux}(\mathsf{sk}, m)) \tag{4.22}$$

and $\mathbf{Adv}^{\mathsf{sNR}^{H,\perp}}_{\mathsf{sBUFF}[\mathcal{S},\mathsf{MD}]}(\mathcal{D}, \mathcal{A}, \mathsf{aux})$

$$\leq 2q_{\mathcal{B}} \cdot r^2 \cdot \mathbf{Adv}^{\mathsf{HnS}^H}_{\mathsf{MD}_{\perp}}(\bar{\mathcal{D}}, \bar{\mathcal{A}}) + \frac{(q_{\mathcal{D}}^2 \cdot r + 1) \cdot \bar{B}\bar{L} + q_{\mathcal{D}} + 2\bar{L}^2}{|\, \mathcal{Y} \,|}. \tag{4.23}$$

where $\bar{B}$ and $q_{\mathcal{S}}$ are as described in Section 4.7.1 and $\bar{L} = q_{\mathcal{D}} + q_{\mathcal{A}} + q_{\mathcal{S}} + 2\bar{B}$. Moreover, if aux is polynomial-time computable and $\mathcal{D}, \mathcal{A}$ are PPT, then so are $\bar{\mathcal{D}}, \bar{\mathcal{A}}$.

Proof. We present here a slightly simplified proof, where we omit the padding, i.e., consider the padding-free $\mathsf{MD}^H_{\perp}$, and instead assume that the message m

output by $\mathcal{D}$ is a multiple of the block length r of the hash function; furthermore, we assume that all public keys have the same length, also a multiple of r. The full proof can be found in Appendix A.3, and we state state the auxiliary claims in the simplified proof in full generality, but mark in colour the parts that are only relevant in the general case.

First, we note that

$$\mathbf{Adv}^{\mathsf{sNR}^{H,\perp}}_{\mathsf{sBUFF}[\mathcal{S},\mathsf{MD}_\perp]}(\mathcal{D},\mathcal{A},\mathsf{aux}) \le \Pr\left[\mathsf{MD}^H_\perp(m\|\mathsf{pk}'\|m) = y' \wedge \mathsf{pk}' \neq \mathsf{pk}\right]$$

with the random variables $\mathsf{pk}, \mathsf{pk}', m$ and y defined by the experiment

$$(\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{KGen},\ m \leftarrow \mathcal{D}^H(\mathsf{sk}),\ (\mathsf{pk}',y') \leftarrow \mathcal{B}^H\left(\mathsf{sk}, \mathsf{MD}^H_\perp(m\|\mathsf{pk}\|m), \mathsf{aux}(\mathsf{sk},m)\right)$$

where $\mathcal{B}(\mathsf{sk},y,a) := \mathcal{A}^H\left(\mathsf{sk},(\mathsf{Sign}^H(\mathsf{sk},y),y),a\right)$. We note that the random choice of H is understood and left implicit. We recall that $\mathcal{D}$ and $\mathcal{B}$ make at most $q_\mathcal{D}$ and $q_\mathcal{B} := q_\mathcal{A} + q_\mathcal{S}$ queries to the random oracle respectively. We introduce the following additional random variables, implicitly defined by the above experiment.

We denote by B the block number of $m\|\mathsf{pk}\|m$ and $m\|\mathsf{pk}'\|m$. We denote by B' the block number of $m\|\mathsf{pk}'$. We denote by m_i the ith block of the message m and by $B'' = |\,m\,|_{\mathrm{bl}}$, that is $m = m_1\|\ldots\|m_{B''}$.

We define

$$z'_1 := \mathsf{MD}^H_\perp(m\|\mathsf{pk}) \qquad \text{and} \qquad z'_{i+1} := \mathsf{MD}^H_\perp(m\|\mathsf{pk}\|m_1\|\ldots\|m_i) = H(m_i, z'_i)$$

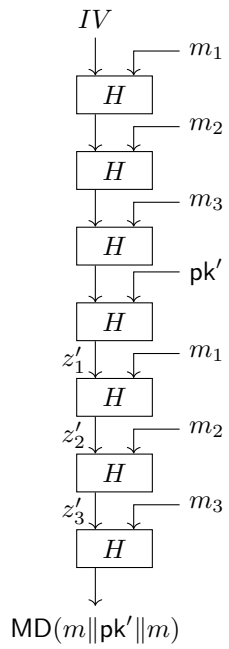
for $i = 1,\ldots,B''$, with $z_{B''+1} = \mathsf{MD}^H_\perp(m\|\mathsf{pk}'\|m)$ then. The z_i's thus form the intermediate digests in the second part of the hash chain when computing $\mathsf{MD}^H_\perp(m\|\mathsf{pk}'\|m)$; for illustration see Fig. 4.18a.

Finally, we let $\tau_1,\ldots,\tau_L$ with $L = q_\mathcal{D} + B + q_\mathcal{B} + B$ be the list of inputs to all the hash computations performed during the experiment, listed in the performed order; see Fig. 4.18b. Hence, $Q_\mathcal{D} = \{\tau_1,\ldots,\tau_{q_\mathcal{D}}\}$ consists of the hash queries made by $\mathcal{D}$, $Q_{\mathsf{MD}_\perp(m\|\mathsf{pk}\|m)} = \{\tau_{q_\mathcal{D}+1},\ldots,\tau_{q_\mathcal{D}+B}\}$ consists of the hash queries made by the challenger when computing $\mathsf{MD}_\perp(m\|\mathsf{pk}\|m)$, $Q_\mathcal{B} = \{\tau_{q_\mathcal{D}+B+1},\ldots,\tau_{q_\mathcal{D}+B+q_\mathcal{B}}\}$ of the queries made by $\mathcal{B}$, and the remaining τ_ℓ's are the inputs to the hash computations done towards computing $\mathsf{MD}^H_\perp(m\|\mathsf{pk}'\|m)$, in particular $\tau_{q_\mathcal{D}+B+q_\mathcal{B}+B'+1} = (m_1, z'_1)$, $\tau_{q_\mathcal{D}+B+q_\mathcal{B}+B'+2} = (m_2, z'_2)$, etc. For any τ_ℓ with $\ell \in [L]$, we write $\mathsf{R}(\ell)$ for the right component of τ_ℓ, i.e., $\mathsf{R}(q_\mathcal{D}+q_\mathcal{B}+B+1) = z'_1$, etc.

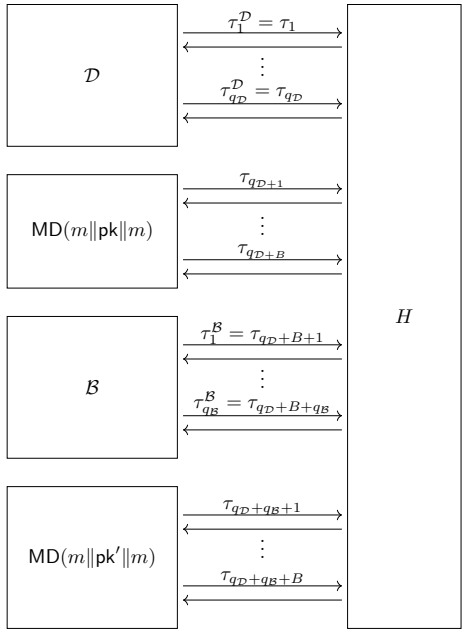
As explained, we are interested in upper-bounding the probability $\Pr[\Sigma]$ of the event

$$\Sigma := \left[\mathsf{MD}^H_\perp(m\|\mathsf{pk}'\|m) = y' \wedge \mathsf{pk}' \neq \mathsf{pk}\right].$$

We do this by introducing a sequence of further events, Γ, Λ and Δ, with the property that $\Pr[\Sigma]$ is close to $\Pr[\Sigma \wedge \Gamma \wedge \Lambda \wedge \Delta]$, assuming $\mathbf{Adv}^{\mathsf{HnS}^H}_{\mathsf{MD}_\perp}(\bar{\mathcal{D}},\bar{\mathcal{A}})$ is



(a) The hash chain for $\mathsf{MD}(m\|\mathsf{pk}'\|m)$ with three message blocks and one pk' block.



(b) Interaction between $\mathcal{D}, \mathcal{B}$, and the random oracle H, along with the queries made by the game in the end when computing the MD hash for the Sandwich BUFF transform.

Figure 4.18: Our notation for the proof of Lemma 4.32

small (for suitable choices of $\bar{\mathcal{D}}$ and $\bar{\mathcal{A}}$), and such that we can upper bound the latter probability.

We start off avoiding some atypical behavior of H. Formally, we consider the good event $\Gamma := \Gamma_1 \wedge \Gamma_2 \wedge \Gamma_3$ with

$$\Gamma_1 := \left[\forall\, \ell, \ell' \in [L] : H(\tau_\ell) = H(\tau_{\ell'}) \;\Rightarrow\; \tau_\ell = \tau_{\ell'}\right]$$
$$\Gamma_2 := \left[\forall\, \ell, \ell' \in [L] : H(\tau_\ell) = \mathsf{R}(\ell') \;\Rightarrow\; (\exists\, \ell_\circ < \ell' : \tau_\ell = \tau_{\ell_\circ})\right] \qquad \text{and}$$
$$\Gamma_3 := \left[\forall\, \ell \in [q_{\mathcal{D}}] : H(\tau_\ell) \neq IV\right].$$

Informally, Γ_1 states that there are no collisions for the points that H was queried on, Γ_2 states that a hash output does not "bump into" a previous hash input, thus retroactively connecting hash chains, and lastly, Γ_3 states that the initialization vector is never a hash output (this will be helpful later on to identify the start of a hash chain).

Claim 4.33. *It holds that* $\Pr[\Sigma] \leq \Pr[\Sigma \wedge \Gamma] + q_{\mathcal{D}}/|\mathcal{Y}| + 2\bar{L}^2/|\mathcal{Y}|$.

Proof. It follows from the collision resistance bound and the zero-preimage resistance bound that $\Pr[\neg\Gamma_1] \leq \bar{L}^2/|\mathcal{Y}|$ and $\Pr[\neg\Gamma_3] \leq q_{\mathcal{D}}/|\mathcal{Y}|$. Also, we immediately obtain $\Pr[\neg\Gamma_2] \leq \bar{L}^2/|\mathcal{Y}|$. Hence we have

$$\Pr[\neg\Gamma] \leq q_{\mathcal{D}}/|\mathcal{Y}| + 2\bar{L}^2/|\mathcal{Y}| \ , \tag{4.24}$$

and thus $\Pr[\Sigma] \leq \Pr[\Sigma \wedge \Gamma] + \Pr[\neg\Gamma] \leq \Pr[\Sigma \wedge \Gamma] + q_{\mathcal{D}}/|\mathcal{Y}| + 2\bar{L}^2/|\mathcal{Y}|$. □

Next, exploiting hide-and-seek, we argue that it is unlikely that $\mathcal{B}$ has queried the entire MD hash chain for $m\|\mathsf{pk}'\|m$ *and* provides the correct output $y' = \mathsf{MD}^H_\perp(m\|\mathsf{pk}'\|m)$. Formally, we consider the event

$$\Lambda := \left[\exists\, i \in [B''] : (m_i, z'_i) \notin Q_{\mathcal{B}}\right]$$

that $\mathcal{B}$ has not made a hash query to one of the high-order intermediate digests z'_i, together with the corresponding message block m_i.

Claim 4.34. *There exist hide-and-seek adversaries* $\bar{\mathcal{D}}, \bar{\mathcal{A}}$ *such that*

$$\Pr[\Sigma \wedge \Gamma \wedge \neg\Lambda] \leq q_{\mathcal{B}} \cdot 2r^2 \cdot \mathbf{Adv}^{\mathsf{HnS}^H_{\mathsf{MD}_\perp}}(\bar{\mathcal{D}}, \bar{\mathcal{A}}) \ , \tag{4.25}$$

where $\bar{\mathcal{A}}$ *makes at most* $q_{\mathcal{B}}$ *and the value* x *chosen by* $\bar{\mathcal{D}}$ *preserves the entropy:*

$$\mathsf{H}_\infty(x \mid z, H) = \mathsf{H}_\infty(m \mid H, \mathsf{sk}, \mathsf{aux}(\mathsf{sk}, m)) \ .$$

Moreover, aux *is polynomial-time computable and* $\mathcal{D}, \mathcal{A}$ *are PPT, then so are* $\bar{\mathcal{D}}$ *and* $\bar{\mathcal{A}}$.

Simplified Proof. [16] We construct adversaries $\bar{\mathcal{D}}$ and $\bar{\mathcal{A}}$ against hide and seek. First, the adversary $\bar{\mathcal{D}}$ simulates the sNR game to $\mathcal{D}$ by sampling a key pair $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}$ and giving sk to $\mathcal{D}$. It forwards all queries and responses by $\mathcal{D}$ to the random oracle and back. When $\mathcal{D}$ outputs a message m, the adversary $\bar{\mathcal{D}}$ outputs $x = m\|\mathsf{pk}\|m$ and $z = (\mathsf{sk}, \mathsf{aux}(\mathsf{sk}, m))$ as its output.

The adversary $\bar{\mathcal{A}}$ takes as input the hash y and $z = \mathsf{sk}, \mathsf{aux}(\mathsf{sk}, m)$. It parses z into sk and $\mathsf{aux}(\mathsf{sk}, m)$ and runs $\mathcal{B}$ on $\mathsf{sk}, y, \mathsf{aux}(\mathsf{sk}, m)$. It forwards all queries to H and their responses. Here, we observe that if Γ and Σ hold but Λ does not hold, then $\bar{\mathcal{A}}$ is able to restore the entire message m (and thus win the corresponding hide-and-seek game against $\mathsf{MD}_\perp$ by recomputing $m\|\mathsf{pk}\|m$), via inspecting $\mathcal{B}$'s queries to H and its output y'. Indeed, $z'_{B''+1} = y'$ (by Σ), and $\mathcal{B}$ has queried $(m_{B''}, z'_{B''})$ such that $H(m_{B''}, z'_{B''}) = z'_{B''+1} = y'$ (by $\neg\Lambda$), and $(m_{B''}, z'_{B''})$ is unique with that property (by Γ_1), and so $\bar{\mathcal{A}}$ can find it. By the same argument, $\bar{\mathcal{A}}$ can then find $(m_{B''-1}, z'_{B''-1}), (m_{B''-2}, z'_{B''-2}), \ldots, (m_1, z'_1)$

[16] This proof is simplified with the assumption that there is no padding and the messages and keys line up with the blocks. For the unsimplified version see Appendix A.3.

in the queries if it knows B'', which is at most $q_{\mathcal{B}}$ and can be guessed with probability $1/q_{\mathcal{B}}$.

Finally, in terms or computational efficiency, $\bar{\mathcal{D}}$ is PPT as long as $\mathcal{D}$ is PPT and aux is polynomial-time computable. Also, if $\mathcal{A}$ is PPT, then $\mathcal{B}$ is PPT and hence so is $\bar{\mathcal{A}}$. □

It remains to to bound the success probability of the adversaries in the case that Λ holds. For this purpose we define m_0 to be the last block of pk' in the sandwich, m_{-1} the second last block etc. and z'_i for $i = 0, -1, -2 \ldots$ accordingly. To this end, consider the events

$$\Delta := \left[\exists\, i \geq -\left|\,\mathsf{pk}\,\right|_{\mathrm{bl}} + 1 : (m_i, z'_i) \notin Q_{\mathcal{B}} \cup Q_{\mathcal{D}} \cup Q_{\mathsf{MD}_\perp(m\|\mathsf{pk}\|m)}\right]$$

and

$$\Delta'_k := \left[\exists\, i \in [B''] : \tau k = (m_i, z'_i) \notin Q_{\mathcal{B}} \cup Q_{\mathsf{MD}_\perp(m\|\mathsf{pk}\|m)} \cup \{\tau k'\}_{k'<k}\right]$$

for $k \in \{1, \ldots, q_{\mathcal{D}}\}$. It is not too hard to see that $\Delta \vee \Delta'_1 \vee \ldots \vee \Delta'_{q_{\mathcal{D}}} \Leftarrow \Sigma \wedge \Lambda \wedge \Gamma$, and thus by basic manipulations

$$\begin{aligned}\Pr[\Sigma] &= \Pr[\Sigma \wedge \Gamma \wedge \Lambda] + \Pr[\Sigma \wedge \Gamma \wedge \neg\Lambda] + \Pr[\Sigma \wedge \neg\Gamma] \\ &\leq \Pr[\Sigma \wedge \Delta] + \sum_k \Pr[\Sigma \wedge \Gamma \wedge \Delta'_k] + \Pr[\Sigma \wedge \Gamma \wedge \neg\Lambda] + \Pr[\neg\Gamma]\,,\end{aligned}$$

where we already have bounds for the last two terms.

First, we argue that $\Pr[\Sigma \wedge \Delta]$ is small. For that purpose, we introduce

$$\Delta^i := \left[i \leq B'' + \left|\,\mathsf{pk}'\,\right|_{\mathrm{bl}} \;\wedge\; (m_{B''-i+1}, z'_{B''-i+1}) \notin Q_{\mathcal{B}} \cup Q_{\mathcal{D}} \cup Q_{\mathsf{MD}_\perp(m\|\mathsf{pk}\|m)}\right]$$

for $i \in [\bar{B}]$.[17] Furthermore, we write $\Delta^{>i} = \Delta^{i+1} \vee \Delta^{i+1} \vee \ldots \vee \Delta^{\bar{B}}$. The crucial observation now is that conditioned on Δ^i, the hash value of $z'_{B''-i+2} = H(m_{B''-i+1}, z'_{B''-i+1})$ is uniformly random and independent of y' (and of prior hash queries etc.), which makes it unlikely for Σ to be satisfied. We formalize this in Claim 4.35 below, and can then conclude that

$$\begin{aligned}&\Pr[\Sigma \wedge \Delta] \\ \leq &\sum_i \Pr\left[\Sigma \wedge \Delta^i \wedge \neg\Delta^{>i}\right] = \sum_i \Pr\left[\Delta^i \wedge \neg\Delta^{>i}\right] \cdot \Pr\left[\Sigma \;\middle|\; \Delta^i \wedge \neg\Delta^{>i}\right] \\ \leq &\sum_i \Pr\left[\Delta^i \wedge \neg\Delta^{>i}\,\right] \cdot \frac{\bar{B}\bar{L}}{|\,\mathcal{Y}\,|} \;\leq\; \frac{\bar{B}\bar{L}}{|\,\mathcal{Y}\,|}\,,\end{aligned}$$

[17] We note that B'' is a random variable (given that m may have variable size), and so the condition $i \leq B''$ ensures $(m_{B''-i+1}, z'_{B''-i+1})$ to be well defined, or else the event is not satisfied.

where it is understood that the sum is over all $i \in [\bar{B}]$ with $\Pr\left[\Delta^i \wedge \neg\Delta^{>i}\right] > 0$, and where $\bar{L} := q_{\mathcal{D}} + \bar{B} + q_{\mathcal{B}} + \bar{B}$; the last inequality follows by the disjointness of $\Delta^i \wedge \neg\Delta^{>i}$ across $i \in [\bar{B}]$.

Claim 4.35. *It holds for every $i \in [\bar{B}]$ with $\Pr[\Delta^i \wedge \neg\Delta^{>i}] > 0$ that*

$$\Pr\left[\Sigma \;\middle|\; \Delta^i \wedge \neg\Delta^{>i}\right] \leq (i-1) \cdot \frac{\bar{L}}{|\mathcal{Y}|} + \frac{1}{|\mathcal{Y}|} \leq \frac{\bar{B}\bar{L}}{|\mathcal{Y}|}\,,$$

where $\bar{L} := q_{\mathcal{D}} + \bar{B} + q_{\mathcal{B}} + \bar{B}$.

Proof. We compute the probability

$$\begin{aligned}
\Pr\left[\Sigma \;\middle|\; \Delta^i \wedge \neg\Delta^{>i}\right] &\leq \Pr\left[\Sigma \wedge \Delta^{i-1} \middle| \Delta^i \wedge \neg\Delta^{>i}\right] + \Pr\left[\neg\Delta^{i-1} \middle| \Delta^i \wedge \neg\Delta^{>i}\right] \\
&\overset{(*)}{\leq} \Pr\left[\Sigma \;\middle|\; \Delta^{i-1} \wedge \Delta^i \wedge \neg\Delta^{>i}\right] + \frac{\bar{L}}{\mathcal{Y}} \\
&\leq \Pr[\Sigma \wedge \Delta^{i-2} | \Delta^{i-1} \wedge \Delta^i \wedge \neg\Delta^{>i}] + \Pr[\neg\Delta^{i-2} | \Delta^{i-1} \wedge \Delta^i \wedge \neg\Delta^{>i}] + \frac{\bar{L}}{\mathcal{Y}} \\
&\leq \Pr[\Sigma \mid \Delta^{i-2} \wedge \Delta^{i-1} \wedge \Delta^i \wedge \neg\Delta^{>i}] + 2 \cdot \frac{\bar{L}}{\mathcal{Y}} \\
&\leq \cdots \\
&\leq \Pr\left[\Sigma \;\middle|\; \Delta^1 \wedge \ldots \wedge \Delta^i \wedge \neg\Delta^{>i}\right] + (i-1) \cdot \frac{\bar{L}}{\mathcal{Y}} \\
&\leq \Pr\left[y' = H(m_{B''}, z'_{B''}) \;\middle|\; \Delta^1 \wedge \ldots \wedge \Delta^i \wedge \neg\Delta^{>i}\right] + (i-1) \cdot \frac{\bar{L}}{\mathcal{Y}} \\
&\overset{(**)}{\leq} \frac{1}{|\mathcal{Y}|} + (i-1) \cdot \frac{\bar{L}}{|\mathcal{Y}|}\,.
\end{aligned}$$

The statement $(*)$ follows the fact that $\neg\Delta^{i-1}$ implies $H(m_{B''-i+1}, z'_{B''-i+1}) \in \{\mathsf{R}(\tau) \mid \tau \in Q_{\mathcal{B}} \cup Q_{\mathcal{D}} \cup Q_{\mathsf{MD}_\perp(m\|\mathsf{pk}\|m)}\}$, which only happens with probability $\bar{L}/\mathcal{Y}$ since $H(m_{B''-i+1}, z'_{B''-i+1})$ is uniform and independent of $Q_{\mathcal{B}} \cup Q_{\mathcal{D}} \cup Q_{\mathsf{MD}_\perp(m\|\mathsf{pk}\|m)}$ conditioned on $\Delta^i \wedge \neg\Delta^{>i}$. For $(**)$ we used the same kind of argument, exploiting that $H(m_{B''}, z'_{B''})$ is uniformly random and independent of y' if $(m_{B''}, z'_{B''})$ has not been queried yet.

In the above series of inequalities we assume the conditions always have non-zero probability. Otherwise, the claim is trivial. In the case that $i \leq 2$ the claim follows directly from the last two rows. □

Towards controlling $\Pr[\Sigma \wedge \mathsf{GD} \wedge \Delta'_k]$, the obstacle is that $\mathcal{D}$ has made a hash query to (m_i, z'_i) and can thus, potentially, make its output m dependent on the hash (e.g., by choosing $m_{i+1} := H(m_i, z'_i)$ then), and so $H(m_i, z'_i)$ may not be "freshly random" anymore.[18] However, by our "sandwich structure" of

[18] Indeed, that is what our attack in Section 4.6.3 exploits.

the hash computation, this is actually not possible. Indeed, since z'_i is a point in *the second part of* the hash chain, all the points in the first part of the chain, i.e., $z_2 := H(m_1, \mathsf{IV})$, $z_3 := H(m_2, z_2)$ up to $z_{B''+1} := H(m_{B''}, z_{B''})$, must be determined already, and hence all of m as well, *before* $\mathcal{D}$ learns the hash of (m_i, z'_i).
We bound the probability in the following claim.

Claim 4.36. $\Pr[\Sigma \wedge \mathsf{GD} \wedge \Delta'_k] \leq q_{\mathcal{D}} \cdot r \cdot \bar{B}\bar{L}/|\mathcal{Y}|$.

Simplified Proof. [19] Formally, for a fixed choice of k, we consider the following procedure to (try to) extract m from the first k queries made by $\mathcal{D}$ and the replies to the first $k-1$ of these queries: Start with the kth query τk and look for a query within $\{\tau 1, \ldots, \tau k - 1\}$ that hashes into $\mathsf{R}(k)$, and then continuing iteratively with that query, until no further such query exists. By construction, this procedure finds $n \leq k$ and $j_1 < \ldots < j_n = k$ such that $H(\tau j_1) = \mathsf{R}(j_2)$, $H(\tau j_2) = \mathsf{R}(j_3), ..., H(\tau j_{n-1}) = \mathsf{R}(j_n)$. The procedure then guesses a value $B^\circ \leftarrow [q_{\mathcal{D}}]$ for the message length. The output of the procedure is then defined to be $\hat{m} := (\mathsf{L}(j_1), \ldots, \mathsf{L}(j_{B^\circ}))$.

We note that $\Sigma \wedge \Gamma \wedge \Delta'_k$ implies that m is a prefix of $\mathsf{L}(j_1)\| \ldots \|\mathsf{L}(j_n)$, and thus, as $n \leq q_{\mathcal{D}}$, it holds that

$$\Pr[m = \hat{m} | \Sigma \wedge \Gamma \wedge \Delta'_k] \geq \frac{1}{q_{\mathcal{D}}}. \tag{4.26}$$

Indeed, Δ'_k implies that $\tau k = (m_i, z'_i)$ for some i, and so $\mathsf{R}(k) = z'_i = \mathsf{MD}^H_\perp(m_1\| \ldots \|m_B\|\mathsf{pk}'\|m_1\| \ldots \|m_{i-1}) = H(m_{i-1}, z'_{i-1}) = H(\tau q_{\mathcal{D}} + q_{\mathcal{B}} + B + i + 1)$. Hence, by Γ_2, there exists $j_{n-1} < k$ so that $\tau j_{n-1} = (m_{i-1}, z'_{i-1})$, and thus $H(\tau j_{n-1}) = \mathsf{R}(k)$. Furthermore, $\mathsf{R}(j_{n-1}) = z'_{i-1} = \mathsf{MD}^H_\perp(m_1\| \ldots \|m_B\|\mathsf{pk}'\|m_1\| \ldots \|m_{i-2})$, and so by repeating the argument, the procedure extracts, in this reversed order, $m_i, m_{i-1}, \ldots, m_1, \mathsf{pk}, m_{B''}, \ldots, m_1$, until $\tau j_1 = (m_1, \mathsf{IV})$, which is when the procedure stops (by Γ_3).

Now we make the following "game hop", by replacing the experiment

$$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen},\ m \leftarrow \mathcal{D}^H(\mathsf{sk}),\ (\mathsf{pk}', y') \leftarrow \mathcal{B}^H\big(\mathsf{sk}, \mathsf{MD}^H_\perp(m\|\mathsf{pk}\|m), \mathsf{aux}(\mathsf{sk}, m)\big),$$

which defined all the above random variables and probabilities, by

$$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen},\ \hat{m} \leftarrow \hat{\mathcal{D}}^H(\mathsf{sk}),\ (\mathsf{pk}', y') \leftarrow \mathcal{B}^H\big(\mathsf{sk}, \mathsf{MD}^H_\perp(\hat{m}\|\mathsf{pk}\|\hat{m}), \mathsf{aux}(\mathsf{sk}, \hat{m})\big),$$

where $\hat{\mathcal{D}}$ runs $\mathcal{D}$, but then stops before sending the kth query to H and instead tries to extract m by means of the above procedure from the prior queries. Correspondingly, we denote its output by $\hat{m}$. We stress that $\hat{\mathcal{D}}$ has now query

[19] We give a simplified proof here, for the full version see Appendix A.3.

complexity $q_{\hat{\mathcal{D}}} = k - 1$. The crucial observation is that

$$\Pr[\Sigma \wedge \Delta'_k \wedge \hat{m} = m] \leq \widehat{\Pr}[\Sigma \wedge \Delta]$$

where we use $\widehat{\Pr}$ to denote probabilities in the new experiment. Indeed, in case $\hat{m} = m$ there is no difference in the new experiment, except that now $\hat{\mathcal{D}}$ stops before doing the kth query, and so if Δ'_k is satisfied in the original experiment then Δ is satisfied in the new one. Thus, we can recycle the bound from above. Using the bound $\widehat{\Pr}[\Sigma \wedge \Delta] \leq \bar{B}\bar{L}/|\mathcal{Y}|$ from Claim 4.35 (which holds for any choice of $\mathcal{D}$ and thus also for $\hat{\mathcal{D}}$) we obtain using the above (4.26) that

$$\Pr[\Sigma \wedge \mathsf{GD} \wedge \Delta'_k] \leq q_{\mathcal{D}} \cdot \Pr[\Sigma \wedge \Gamma \wedge \Delta'_k \wedge \hat{m} = m] \leq q_{\mathcal{D}} \cdot \widehat{\Pr}[\Sigma \wedge \Delta] \leq q_{\mathcal{D}} \bar{B}\bar{L}/|\mathcal{Y}|,$$

which concludes the proof of Claim 4.36. □

We wrap up the proof by adding up the probabilities

$$\begin{aligned}
\Pr[\Sigma] \leq& \Pr[\Sigma \wedge \Delta] + \sum_{k=1}^{q_{\mathcal{D}}} \Pr[\Sigma \wedge \Gamma \wedge \Delta'_k] + \Pr[\Sigma \wedge \Gamma \wedge \neg\Lambda] + \Pr[\neg\Gamma] \\
\leq& \frac{\bar{B}\bar{L}}{|\mathcal{Y}|} + \frac{q_{\mathcal{D}}^2 \cdot r \cdot \bar{B}\bar{L}}{|\mathcal{Y}|} + q_{\mathcal{B}} \cdot 2r^2 \cdot \mathbf{Adv}^{\mathsf{HnS}^H}_{\mathsf{MD}_\perp}(\bar{\mathcal{D}}, \bar{\mathcal{A}}) + \frac{q_{\mathcal{D}} + 2\bar{L}^2}{|\mathcal{Y}|} \\
=& q_{\mathcal{B}} \cdot 2r^2 \cdot \mathbf{Adv}^{\mathsf{HnS}^H}_{\mathsf{MD}_\perp}(\bar{\mathcal{D}}, \bar{\mathcal{A}}) + \frac{(q_{\mathcal{D}}^2 \cdot r + 1) \cdot \bar{B}\bar{L} + q_{\mathcal{D}} + 2\bar{L}^2}{|\mathcal{Y}|}
\end{aligned}$$

which shows the claimed bound. □

4.7.3 Hide-and-Seek for Merkle-Damgård

In this section, we provide a (classical) bound on the Hide-and-Seek property of Merkle-Damgård hash functions. For the purpose of using our bounds together with Lemma 4.32, it is sufficient to consider the padding-free versions $\mathsf{MD}^H_\perp : \mathcal{X}^{\leq \bar{B}} \to \mathcal{Y}$ with a bounded number of (at most $\bar{B}$) input blocks, where $\mathcal{X} = \{0,1\}^r$.

Theorem 4.37 ($\mathsf{MD}^H_\perp$ satisfies HnS^H)**.** *Let $\mathcal{D} : \{\perp\} \to \mathcal{X}^{\leq \bar{B}} \times \mathcal{Z}$ and $\mathcal{A} : \mathcal{Y} \times \mathcal{Z} \to \mathcal{X}^{\leq \bar{B}}$ be $\mathsf{HnS}^H_{\mathsf{MD}_\perp}$-adversaries satisfying (4.10) for some $0 < \epsilon < 1$, where $\mathcal{A}$ makes q classical queries to H. Then for $k := \lceil \log|\mathcal{Z}| + \log(1/\epsilon) \rceil$ we have*

$$\mathbf{Adv}^{\mathsf{HnS}^H}_{\mathsf{MD}_\perp}(\mathcal{D}, \mathcal{A}) \leq 2k(q + \bar{B})\epsilon + \epsilon \leq 3k(q + \bar{B})\epsilon\,.$$

To prove the above statement, we recall Corollary 4.25, which reduces the Hide-and-Seek security of any hash function $\mathcal{F}$, to the probability of simultaneously finding preimages of multiple randomly chosen targets. Plugging in $\mathcal{F} = \mathsf{MD}^H_\perp$, we then control the latter in Lemma 4.38 further below, which can

be obtained from the observation that inverting $\mathsf{MD}_{\perp}^{H}$ necessarily inverts H at the last round.

Corollary 4.25. *Let $\mathcal{F} : \mathcal{X} \to \mathcal{Y}$ be a fixed function, $\mathcal{D} : \{\perp\} \to \mathcal{X} \times \mathcal{Z}$ and $\mathcal{A}$ be any Hide-and-Seek adversaries against $\mathcal{F}$. Then for every $(k, T) \in \mathbb{Z}_{>0} \times \mathbb{R}_{>0}$, for $(x, z) \leftarrow \mathcal{D}$ and for $y_1^u, \ldots, y_k^u \leftarrow \mathcal{Y}$ sampled uniformly and independently, we have*

$$\Pr\left[\, x = \mathcal{A}(\mathcal{F}(x), z)\,\right] \le T \cdot \mathsf{guess}(x \mid z) + \left(\frac{|\,\mathcal{Y}\,|}{T}\right)^{k} \Pr\left[y_i^u = \mathcal{F}\big(\mathcal{A}(y_i^u, z)\big)\ \forall i \in [k]\right].$$

Lemma 4.38. *Let $\mathcal{A}^H$ be an oracle algorithm making at most q classical queries to H. Then for $y_1^u, \ldots, y_k^u \leftarrow \mathcal{Y}$ sampled uniformly and independently, we have*

$$\Pr\left[\mathsf{MD}_{\perp}^{H}\left(\mathcal{A}^H(y_i^u)\right) = y_i^u\ \forall i \in [k]\right] \le \big(k(q+\bar{B})/\,|\,\mathcal{Y}\,|\big)^{k}.$$

Proof. For every $i \in [k]$, let $x_i' \| x_i'' := \mathcal{A}^H(y_i^u)$ where $x_i'' \in \mathcal{X}$. Then, with the convention that $\mathsf{MD}_{\perp}^{H}(x_i') := \mathsf{IV}$ if $x_i' = \perp$ is the empty string,

$$\begin{aligned}\Pr\left[\mathsf{MD}_{\perp}^{H}\left(\mathcal{A}^H(y_i^u)\right) = y_i^u\ \forall i \in [k]\right] &\le \Pr\left[H(x_i'', \mathsf{MD}_{\perp}^{H}(x_i')) = y_i^u\ \forall i \in [k]\right] \\ &\le \big(k(q+\bar{B})/\,|\,\mathcal{Y}\,|\big)^{k},\end{aligned}$$

where the second inequality is by Lemma 4.26. □

We wrap up the proof of the Hide-and-Seek property for $\mathsf{MD}_{\perp}^{H}$ (Theorem 4.37) below.

Proof of Theorem 4.37. Combining Corollary 4.25 (for $\mathcal{F} = \mathsf{MD}_{\perp}^{H}$ with $\mathcal{X}_{\mathcal{F}} := \mathcal{X}^{\le B}$ and $\mathcal{Y}_{\mathcal{F}} = \mathcal{Y}$) and Lemma 4.38 immediately gives us

$$\begin{aligned}\mathbf{Adv}_{\mathsf{MD}_{\perp}}^{\mathsf{HnS}^H}(\mathcal{D}, \mathcal{A}) &\le T \cdot \epsilon + \frac{|\,\mathcal{Y}\,|^k}{T^k} \cdot \sum_{z^\circ \in \mathcal{Z}} \Pr\left[\mathsf{MD}_{\perp}^{H}(\mathcal{A}(y_i^u, z^\circ)) = y_i^u\ \forall i \in [k]\right], \\ &\le T \cdot \epsilon + |\,\mathcal{Z}\,| \cdot \left(\frac{k(q+\bar{B})}{T}\right)^{k} \qquad (4.27)\end{aligned}$$

for every $k, T \in \mathbb{Z}_{>0}$, where $y_1^u, \ldots, y_k^u \leftarrow \mathcal{Y}_f$ are sampled uniformly and independently. Plugging in $T = 2k(q+\bar{B})$ and $k = \lceil \log |\,\mathcal{Z}\,| + \log(1/\epsilon) \rceil$, the above is then bounded by

$$\le 2k(q+\bar{B})\epsilon + |\,\mathcal{Z}\,| \cdot 2^{-k} \le 2k(q+\bar{B})\epsilon + \epsilon\,,$$

which concludes the proof. □

4.8 Positive Results with Computational Entropy

Here, we consider the computational variant of $\mathsf{sNR}^{H,\perp}$, where we restrict $\mathcal{D}^H, \mathcal{A}^H$ and aux to be PPT (oracle) algorithms. Furthermore, the entropy requirement (4.9) is replaced by

$$\mathsf{HILL}^{H}_{\infty} \underset{\substack{(\mathsf{sk},pk)\leftarrow\mathsf{KGen}^H \\ m\leftarrow\mathcal{D}^H(\mathsf{sk})}}{(m \mid \mathsf{sk}, \mathsf{aux}(\mathsf{sk}, m))} \geq \omega(\log \lambda)\,, \tag{4.28}$$

and we then naturally demand that the game $\mathsf{sNR}^{H,\perp}$ can be won with negligible probability $\mathsf{negl}(\lambda)$ only.

Remark 4.39. *Interestingly, and maybe somewhat surprisingly, in the computational setting* $\mathsf{sNR}^{H,\perp}$ *does not imply* $\mathsf{NR}^{H,\perp}$*, in contrast to the statistical setting, as explained in Section 4.5.1. Indeed, in Section 4.4.4 we showed that the BUFF transform* $\mathsf{BUFF}[\mathcal{S}, H]$ *does in general not satisfy* $\mathsf{NR}^{H,\perp}$ *in the computational setting, while below we show that it does satisfy* $\mathsf{sNR}^{H,\perp}$*. See Remark 4.21 for why our proof does not carry over to* $\mathsf{NR}^{H,\perp}$*. We suspect that the two notions are incomparable in the computational setting.*

4.8.1 BUFF via RO is sNR, Computationally

We get the following positive result on the computational $\mathsf{sNR}^{H,\perp}$ security of the BUFF transform $\mathsf{BUFF}[\mathcal{S}, H]$ when using a random oracle H.

Theorem 4.40. *Let* $\mathcal{S} = (\mathsf{KGen}, \mathsf{Sign}^H, \mathsf{Vrfy}^H)$ *be a signature scheme in ROM, where* KGen *makes no query to* H*, and let* $\mathsf{BUFF}[\mathcal{S}, H]$ *be the signature scheme obtained by applying the BUFF transform that uses* H*. Then for every PPT hint function* aux*, and for any PPT adversaries* $\mathcal{D}^H, \mathcal{A}^H$ *against* $\mathsf{sNR}^{H,\perp}_{\mathsf{BUFF}[\mathcal{S},H]}$ *that satisfy* (4.28)*, we have*

$$\mathbf{Adv}^{\mathsf{sNR}^{H,\perp}}_{\mathsf{BUFF}[\mathcal{S},H]}(\mathcal{D}, \mathcal{A}, \mathsf{aux}) \leq \mathsf{negl}(\lambda)\,.$$

In spirit, we can recycle Lemma 4.20 to reduce the computational variant of $\mathsf{sNR}^{H,\perp}$ to the computational variant of Hide-and-Seek, and then we show in Lemma Lemma 4.41 that the latter is hard as well, which follows rather directly from the statistical hardness and the definition of the HILL entropy.

Proof. Take $(\bar{\mathcal{D}}, \bar{\mathcal{A}})$ as in Lemma 4.20, for which

$$\mathbf{Adv}^{\mathsf{sNR}^{H,\perp}}_{\mathsf{BUFF}[\mathcal{S},H]} \leq \mathsf{poly}(\lambda) \cdot \mathbf{Adv}^{\mathsf{HnS}^H}(\bar{\mathcal{D}}, \bar{\mathcal{A}}) + \mathsf{negl}(\lambda)\,,$$

where we exploit that the numbers of queries made by Sign, $\mathcal{D}$, and $\mathcal{A}$ are bounded by their (polynomial) running times, respectively, and that the addi-

tive term $q_K\epsilon$ in (4.13) vanishes due to the assumption that KGen makes no query to H. Hence it suffices to control the HnS^H advantage of $(\bar{\mathcal{D}}, \bar{\mathcal{A}})$.

Towards this end, we first note that, by inspecting the construction of $\bar{\mathcal{D}}$ with $x = (\mathsf{pk}, m)$ and $z = (\mathsf{sk}, \mathsf{aux}(\mathsf{sk}, m))$, the HILL entropy variant of (4.11) follows:

$$\mathsf{HILL}^H_{\infty,(x,z)\leftarrow\bar{\mathcal{D}}^H}(x \mid z) \geq k(\lambda) \iff \mathsf{HILL}^H_{\infty,\substack{(\mathsf{sk},pk)\leftarrow\mathsf{KGen}^H \\ m\leftarrow\mathcal{D}^H(\mathsf{sk})}}(m \mid \mathsf{sk}, \mathsf{aux}(\mathsf{sk}, m)) \geq k(\lambda) \ ,$$

where the equivalence is due to the public key pk being *efficiently derivable* from its corresponding secret key sk, and so (4.14) also holds for the HILL entropy. Combining the above with (4.28), we obtain

$$\mathsf{HILL}^H_{\infty,(x,z)\leftarrow\bar{\mathcal{D}}^H}(x \mid z) \geq \omega(\log\lambda) \ .$$

Moreover, by Lemma 4.20, $\bar{\mathcal{D}}\colon \{\bot\} \to \mathcal{X} \times \mathcal{Z}$ with $\mathcal{Z} = \mathcal{SK} \times \mathcal{AUX}$. Hence

$$\log|\mathcal{Z}| = \log|\mathcal{SK}| + \log|\mathcal{AUX}| \leq \mathsf{poly}(\lambda)$$

due to both KGen and aux being poly-time. Finally, Lemma 4.20 ensures that $\bar{\mathcal{A}}$ is PPT whenever $\mathcal{A}$ is, which is satisfied by assumption. Thus, the assumptions for Lemma 4.41 below (the hardness of Hide-and-Seek in the computational setting) are all satisfied, and so

$$\mathbf{Adv}^{\mathsf{HnS}^H}(\bar{\mathcal{D}}, \bar{\mathcal{A}}) \leq \mathsf{negl}(\lambda) \ ,$$

which concludes the proof. □

The following provided the computational hardness of Hide-and-Seek.

Lemma 4.41. *Let $\mathcal{D}^H\colon \{\bot\} \to \mathcal{X} \times \mathcal{Z}$ and $\mathcal{A}^H : \mathcal{Y} \times \mathcal{Z} \to \mathcal{X}$ be adversaries against HnS^H, with $\mathcal{A}$ being PPT, $\log|\mathcal{Z}| < \mathsf{poly}(\lambda)$, and*

$$\mathsf{HILL}^H_{\infty,(x,z)\leftarrow\mathcal{D}^H}(x \mid z) \geq \omega(\log\lambda) \ .$$

Then $\mathbf{Adv}^{\mathsf{HnS}^H}(\mathcal{D}, \mathcal{A}) \leq \mathsf{negl}(\lambda)$.

Proof. Let $(x, z) \leftarrow \mathcal{D}^H$. Via the entropy condition, there is an (H-dependent) random variable $x^* \in \mathcal{X}$ such that $\mathsf{guess}(x^* \mid H, z) \leq \mathsf{negl}(\lambda)$ and moreover (x^*, z) and (x, z) are computationally indistinguishable. Without loss of generality, we may assume (x^*, z) is sampled via a (possibly unbounded) hider $\mathcal{D}^{*H}$. Now, inspect the displayed games $\mathsf{HnS}^H(\mathcal{D}, \mathcal{A})$ and $\mathsf{HnS}^H(\mathcal{D}^*, \mathcal{A})$ below.

$\mathsf{HnS}^H(\mathcal{D}, \mathcal{A})$
1: $(x, z) \leftarrow \mathcal{D}^H$
2: **return** $x = \mathcal{A}^H(H(x), z)$

$\mathsf{HnS}^H(\mathcal{D}^*, \mathcal{A})$:
1: $(x^*, z) \leftarrow \mathcal{D}^{*H}$
2: **return** $x^* = \mathcal{A}^H(H(x^*), z)$

By the computational indistinguishability, it follows that

$$|\mathbf{Adv}^{\mathsf{HnS}^H}(\mathcal{D}, \mathcal{A}) - \mathbf{Adv}^{\mathsf{HnS}^H}(\mathcal{D}^*, \mathcal{A})| \leq \mathsf{negl}(\lambda)\,.$$

Finally, we can apply Theorem 4.22 to the HnS^H adversaries $\mathcal{D}^*$ and $\mathcal{A}$, which satisfy the statistical entropy condition, and so we have $\mathbf{Adv}^{\mathsf{HnS}}(\mathcal{D}^*, \mathcal{A}) \leq \mathsf{negl}(\lambda)$. This concludes the proof. □

Remark 4.42. *Interestingly, towards proving* $\mathsf{sNR}^{H,\perp}$ *of the BUFF transform in the* statistical *setting, as we did earlier in the thesis, it would have been sufficient to show that the random oracle satisfies (the statistical variant of)* HnS *for a* query bounded *hider* $\mathcal{D}$*. However, for the above line of reasoning in the computational setting, it is essential that Theorem 4.22 holds for a query unbounded hider; indeed, above,* x^* *may be arbitrarily dependent on* H*, and so might not be producible by a query bounded hider* $\mathcal{D}^*$*.*

4.8.2 Sandwich BUFF via MD is sNR, Computationally

We obtain a similar positive result on the computational $\mathsf{sNR}^{H,\perp}$ security of the Sandwich BUFF $\mathsf{sBUFF}[\mathcal{S}, \mathsf{MD}]$ when using a Merkle-Damgård hash function. Recall that, we denote by $\mathsf{MD}^H : \{0,1\}^* \to \mathcal{Y}$ the Merkle-Damgård hash function that is allowed to query a random oracle $H : \mathcal{X} \times \mathcal{Y} \to \mathcal{Y}$ as its round function, where $\mathcal{X} = \{0,1\}^r$ and $\mathcal{Y} = \{0,1\}^{n_f}$ satisfy $\omega(\log \lambda) \leq r, n_f \leq \mathsf{poly}(\lambda)$.

Theorem 4.43. *Let* $\mathcal{S} = (\mathsf{KGen}, \mathsf{Sign}^H, \mathsf{Vrfy}^H)$ *be a signature scheme in ROM, where* KGen *makes no query to* H*, and let* $\mathsf{sBUFF}[\mathcal{S}, \mathsf{MD}]$ *be the signature scheme obtained by applying the Sandwich BUFF that uses* MD^H*. Then for every PPT hint function* aux*, and* $\mathsf{sNR}^{H,\perp}$ *adversaries* $\mathcal{D}$ *and* $\mathcal{A}$ *that satisfy* (4.28)*, we have*

$$\mathbf{Adv}^{\mathsf{sNR}^{H,\perp}}_{\mathsf{sBUFF}[\mathcal{S},\mathsf{MD}]}(\mathcal{D}, \mathcal{A}, \mathsf{aux}) \leq \mathsf{negl}(\lambda)\,.$$

Similarly, once we have Lemma 4.32 and Theorem 4.37, the proof follows line by line as that of Theorem 4.40. Indeed, the reductions in Lemma 4.32 carry the HILL entropy from the $\mathsf{sNR}^{H,\perp}$ game to the $\mathsf{HnS}^H_{\mathsf{MD}}$ game, and the proven Hide-and-Seek property in Theorem 4.37 carries over to the computational setting.

Proof of Theorem 4.43. Since $\mathcal{D}, \mathcal{A}$ runs in polynomial-time, for $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}$ and $m \leftarrow \mathcal{D}^O(\mathsf{sk})$ the message length $|m| \leq \mathsf{poly}(\lambda)$ is polynomially bounded. Also we assume (up to a negligible security loss) that m is always

non-empty via (4.28). Hence we consider the domain of MD only consists of non-empty bit strings that are at most $\bar{B}$ blocks long, for some $\bar{B} \leq \mathsf{poly}(\lambda)$.

Let $\bar{\mathcal{D}}$ and $\bar{\mathcal{A}}$ be the PPT algorithms as specified in Lemma 4.32, so that

$$\begin{aligned}
&\mathbf{Adv}^{\mathsf{sNR}^{H,\perp}}_{\mathsf{sBUFF}[\mathcal{S},\mathsf{MD}]}(\mathcal{D},\mathcal{A},\mathsf{aux}) \\
&\leq 2q_{\mathcal{B}}{\cdot}r^2 \cdot \mathbf{Adv}^{\mathsf{HnS}^H_{\mathsf{MD}_\perp}}(\bar{\mathcal{D}},\bar{\mathcal{A}}) + \frac{(q_{\mathcal{D}}^2{\cdot}r+1)\cdot \bar{B}\bar{L} + q_{\mathcal{D}} + 2\bar{L}^2}{|\,\mathcal{Y}\,|}. \\
&\leq \mathsf{poly}(\lambda) \cdot \mathbf{Adv}^{\mathsf{HnS}^H_{\mathsf{MD}_\perp}}(\bar{\mathcal{D}},\bar{\mathcal{A}}) + \mathsf{negl}(\lambda) \qquad (4.29)
\end{aligned}$$

where parameters $\bar{L}, q_{\mathcal{B}}, k$ are as specified in the lemma, and the last inequality exploits the fact that $r, c, \bar{B}, \bar{L}, q_{\mathcal{B}}, q_{\mathcal{D}}, q_{\mathcal{A}}, q_{\mathcal{S}} \leq \mathsf{poly}(\lambda)$ and that $|\,\mathcal{Y}\,| \geq \lambda^{\omega(1)}$. Moreover by inspecting the construction of $\bar{\mathcal{D}}$ it follows that

$$\mathsf{HILL}^H_{\infty}_{(x,z)\leftarrow\bar{\mathcal{D}}^O} \geq k(\lambda) \iff \mathsf{HILL}^H_\infty(m \mid \mathsf{sk}, \mathsf{aux}(\mathsf{sk}, m)) \geq k(\lambda)\ .$$

Combining the above with (4.28), we thus have

$$\mathsf{HILL}^H_{\infty}_{(x,z)\leftarrow\bar{\mathcal{D}}^H}(x \mid z) \geq \lambda^{\omega(1)}\ ,$$

which implies the existence of an H-dependent random variable x^* for $(x,z) \leftarrow \mathcal{D}^H$ such that

$$\mathsf{HILL}_\infty(x^* \mid z, H) \geq \omega(\log\lambda)\ ,$$

and yet (x^*, z) and (x, z) are computationally indistinguishable for oracle algorithms querying H. Without loss of generality, let $\mathcal{D}^*$ be the Hide-and-Seek seeker that samples x^*, and note that the seeker $\bar{\mathcal{A}}$ is PPT and makes only polynomially many queries to H. Hence,

$$\mathbf{Adv}^{\mathsf{HnS}^H_{\mathsf{MD}_\perp}}(\bar{\mathcal{D}},\bar{\mathcal{A}}) \leq \mathbf{Adv}^{\mathsf{HnS}^H_{\mathsf{MD}_\perp}}(\mathcal{D}^*,\bar{\mathcal{A}}) + \mathsf{negl}(\lambda) \leq \mathsf{negl}(\lambda)\ ,$$

where the first inequality follows from the indistinguishability between (x^*, z) and (x, z), and the last inequality follows from Theorem 4.37 respectively in the Merkle-Damgård and the Sponge case. Putting things together, we conclude the proof. □