Schnorr-like Signatures in the Non-Observable Random Oracle Model

Marian Dietz and Dennis Hofheinz

Department of Computer Science
ETH Zurich
Zurich, Switzerland
{marian.dietz,hofheinz}@inf.ethz.ch

Abstract. Schnorr's signature scheme and many of its variants are among the most efficient group-based digital signature schemes. Schnorr's scheme has very compact signatures (consisting of only two exponents in its most compact form). However, its security reduction is notoriously non-tight and requires a strong ("programmable") version of the random oracle model. Variants with a tight(er) security proof in a more realistic model exist, but are less compact and efficient.
In this work, we investigate whether these disadvantages are inherent to Schnorr's signatures and its variants. In particular, we define a family of "Schnorr-like" signature schemes, which contains group-based signature schemes with verification similar to Schnorr's scheme. To explore the necessity of (heavy) random oracle abstractions for such schemes, we allow only for a very weak ("non-programmable, non-observable") version of a random oracle in the security proof.
Our main result is that **there is no tight reduction of the security of any such "Schnorr-like" scheme to any group-based assumption that holds generically.**
We also show that this result itself is tight, in the sense that non-tightly secure schemes exist. Similarly, already for a slightly generalized definition of "extended Schnorr-like" schemes, tightly secure schemes exist.
Our main result employs a meta-reduction with a new "filtering" technique that may be of independent interest.

1 Introduction

Schnorr signatures. Schnorr signatures [Sch90] are among the most compact and efficient group-based digital signature schemes. They use a group in a black-box manner, except that group elements are hashed to bitstrings at one point during signature generation and verification. Schnorr signatures can be proven secure assuming one-wayness of exponentiation in the used group, but the known proof only works in the programmable random oracle model [PS96, FH17]. Furthermore, this is inherently non-tight, i.e., suffers from a large security loss [Seu12, FJS14].[1] [2] On the other hand, variants of Schnorr's scheme are even tightly secure in the so-called non-programmable random oracle model [FHJ20]. However, it is somewhat unsatisfying that the only known analyses of a highly practical family of signatures schemes rely on a proof heuristic such as the random oracle methodology.

Random oracles, and how they differ. The random oracle model, introduced in complexity theory [BG81], has proved to be a highly useful heuristic for security proofs [BR93]. In it, a hash function $\mathcal{O}_{\mathsf{H}}$ used by a cryptographic scheme is modeled to be a truly random function in the security analysis (although in practice, $\mathcal{O}_{\mathsf{H}}$ will have to be implemented by a concrete function such as SHA-3).

[1] Cryptographic reductions construct from any given adversary $\mathcal{A}$ on the investigated scheme a solver $\mathcal{B}$ for the underlying (presumably hard) computational problem. If $\mathcal{B}$ takes more resources or is less successful than $\mathcal{A}$, then that reduction is called non-tight. Tight reductions are preferable from a concrete security standpoint.

[2] We remark that it *is* possible to prove Schnorr signatures even tightly (or "somewhat tightly" [BD20, RS21]) secure, under an ad-hoc complexity assumption [CFO24] or in idealized models of (group) computation [FKL18, FPS20].

Specifically, most known random oracle proofs assume that $\mathcal{O}_{\mathsf{H}}$ is *observable* [FLR+10, AB13] by the security reduction R, i.e., that R sees what $\mathcal{O}_{\mathsf{H}}$-queries an adversary makes. Many proofs also assume that $\mathcal{O}_{\mathsf{H}}$ is *programmable* [BR93], i.e., that R can intercept adversarial $\mathcal{O}_{\mathsf{H}}$-queries and answer them arbitrarily.

A proof in the non-observable (or non-programmable) random oracle model is always preferable to one in the observable (or programmable) random oracle model. The fewer idealizations are necessary for the analysis, the more realistic the derived security guarantees become. In particular for the case of Schnorr signatures and its variants, the absence of security analyses (under standard assumptions) in the non-observable random oracle may be considered a bit worrying. At least for *unmodified* Schnorr signatures, we know that programmable random oracles are necessary [FJS14], and similar results for other cryptographic schemes or tasks are known as well [Nie02, FLR+10]. But already for slight variants of Schnorr signatures, it is unclear whether a proof in the non-observable random oracle model exists.

Our definition of Schnorr-like signatures. To state our contribution, we formulate a class of "Schnorr-like" signatures that generalizes Schnorr signatures.

Recall that verification in the original Schnorr signature scheme works as follows: given public key $\mathrm{pk} = (\mathsf{g}_0, \mathsf{g}_1)$ (consisting of two group elements) and a signature $\sigma = (\mathsf{h}, t)$ (consisting of a group element h and an exponent t) for a message m, verification is successful iff

$$t \cdot \mathsf{g}_0 - r \cdot \mathsf{g}_1 \;=\; \mathsf{h},$$

where $r = \mathcal{O}_{\mathsf{H}}(m, \mathsf{h})$ is computed using the hash function $\mathcal{O}_{\mathsf{H}}$. (Throughout the paper, we write the group operation *additively* as above.)

Schnorr-like signatures follow a similar, but more general, pattern: they are group-based (formalized in a variant of Maurer's generic group model [Mau05]), and they may utilize a hash function $\mathcal{O}_{\mathsf{H}}$ that maps group elements to bitstrings. Verification may only check a single system of linear equations

$$A \cdot \mathbf{g} \;=\; \mathbf{h},$$

where $\mathbf{g}$ and $\mathbf{h}$ are vectors of group elements from public key and signature, respectively, and A is an exponent matrix that may depend on the message, the signature, and a single hash value (derived from public key and the signature).

Beyond Schnorr signatures, many other schemes that follow the Fiat-Shamir paradigm can be formalized in this way [FS87].[3] However, unlike the negative results from e.g., [KMP16, FHJ20], we do not *require* that the signature is derived from the Fiat-Shamir paradigm; we only require a certain algebraic form.

Our results. Our main result is a lower bound:

(1) There is no tight security reduction of any Schnorr-like scheme to any computational assumption in the generic group model.

Our result relies on a new type of meta-reduction (that goes far beyond the techniques used in [FJS14] for the unmodified Schnorr scheme): in our setting, the distribution of forgeries that a straightforward (computationally unbounded) ideal adversary would produce differs significantly from the distribution that an efficient meta-reduction (who does not know the dlog of the public key elements $\mathbf{g}$) can simulate. We use what we call a "filtering" technique that, using just a few signature queries, extracts information about sk and the reduction's distribution of signatures. We then show that this information implies some forgery distribution that both the ideal adversary and the meta-reduction are able to adhere to if they have the filtering step built-in (see 2 for a technical overview).

We also show that our main result is essentially optimal:

[3] However, the *proofs* of Fiat-Shamir-style require programmable and/or observable random oracles, which we explicitly disallow.

(2a) There is a Schnorr-like signature scheme whose security can be *non-tightly* reduced to the Decisional Diffie-Hellman assumption. In fact, this scheme can be viewed as a variation of Waters' [Wat05] signature scheme, with pairings replaced by non-interactive zero-knowledge proofs.[4] The main difficulty here is to find a suitably weak notion of non-interactive zero-knowledge that does not require observable or programmable random oracles, but still suffices to show unforgeability.

(2b) We can also circumvent our main result by generalizing the class of Schnorr-like signatures slightly, to "extended Schnorr-like signatures". Such signatures schemes are identical to Schnorr-like schemes, except that verification may be of the form

$$A \cdot \mathbf{g} \;=\; B \cdot \mathbf{h},$$

where $\mathbf{g}$, $\mathbf{h}$, and A are as before, and B is another exponent matrix that may depend on message, signature, and the single hash value. Concretely, we present a simple, tightly secure extended Schnorr-like signature in our model. To obtain our scheme, we take a suitable algebraic tightly secure signature scheme [AHN+23], and replace its pairing operations with non-interactive zero-knowledge proofs, as in (2a).

To summarize, our definition of Schnorr-like signatures captures precisely which group-based signatures with hash can be proven secure (tightly or not) when idealizing the hash function only weakly.

For context, we compare our main result with existing lower bounds on group-based signature schemes in 1 (also see related works below for a discussion). To complement this, in 2 we further list existing signature schemes that fit into our class of Schnorr-like signatures (and others mentioned above), and compare those with our new constructions.

Reference	class	assumption	reduction restrictions	
			tight	ROM
[FJS14]	{Schnorr}	non-interactive	yes	–
[FJS14]	{Schnorr}	non-interactive	–	NP
[KMP16]	Fiat-Shamir[ID]	security of ID	yes	–
[FHJ20]	Parallel-OR	non-interactive	–	NP
This work	Schnorr-like	any	yes	NP and NO

Fig. 1: Lower bounds for *reductions* proving signature schemes secure. "Class" denotes the class of signature schemes covered, "assumption" the class of assumptions that the reduction is allowed to be based on, "tight" denotes whether only tightly secure reductions are covered, and "ROM" denotes whether only reductions treating the ROM as **N**on-**P**rogrammable resp. **N**on-**O**bservable are covered. The first two lower bounds only apply to the original Schnorr signature scheme, and the third one to a generalization of Schnorr in which *any* identification scheme may be used. The fourth lower bound applies to a specific class of schemes (which does not include Schnorr signatures but uses similar techniques), and our lower bound applies to any Schnorr-like scheme.

More related work. [FHJ20] show that proofs for Schnorr variants of a specific form ("Parallel-OR signatures") require programmability. The signature schemes they consider are different from ours (and technically do not even cover the original Schnorr signature scheme): their parallel-OR signatures are not necessarily group-based, but must be the result of the Fiat-Shamir transformation [FS87] applied to a specific, "OR-type" Sigma-protocol [CDS94, Cra96]. In contrast, our Schnorr-like protocols are group-based, and we require that verification has a particular algebraic form, but we do not stipulate

[4] We stress that this idea is not new, and can be traced back to how [GJ03] replaced pairings in the BLS signature scheme [BLS01]. For a newer take in the setting of blind signatures, see also [CATZ24].

Reference	$\lvert pk\rvert$	$\lvert\sigma\rvert$	sec. loss	assumption	(E)SL	ROM
[Sch90, PS96]	$1 \cdot \mathbb{G}$	$2 \cdot \mathbb{Z}_p$	$O(q_{\mathcal{A}}/\varepsilon_{\mathcal{A}})$	DLog	SL*	P,O
[Sch90, FPS20]	$1 \cdot \mathbb{G}$	$2 \cdot \mathbb{Z}_p$	$O(1)$	AGM+DLog	SL*	P,O
[Sch90, BD20]	$1 \cdot \mathbb{G}$	$2 \cdot \mathbb{Z}_p$	$O(1)$	1-MBDL	SL*	P,O
[Sch90, CFO24]	$1 \cdot \mathbb{G}$	$2 \cdot \mathbb{Z}_p$	$O(1)$	CDL	SL*	P,O
[Oka93, Seu12]	$1 \cdot \mathbb{G}$	$1 \cdot \mathbb{G} + 2 \cdot \mathbb{Z}_p$	$O(q/\varepsilon_{\mathcal{A}})$	DLog	SL	P,O
[CP93, GJ03]	$1 \cdot \mathbb{G}$	$1 \cdot \mathbb{G} + 2 \cdot \mathbb{Z}_p$	$O(1)$	CDH	no	P,O
[KW03]	$3 \cdot \mathbb{G}$	$1 \cdot \mathbb{G} + 1 \cdot \mathbb{Z}_p$	$O(1)$	DDH	SL*	P,O
[Che05]	$1 \cdot \mathbb{G}$	$1 \cdot \mathbb{G} + 2 \cdot \mathbb{Z}_p$	$O(1)$	CDH	no	P,O
[FHJ20]	$2 \cdot \mathbb{G}$	$2 \cdot \mathbb{G} + 2 \cdot \mathbb{Z}_p$	$O(1)$	DDH	no	NP,O
Ours (§4.1)	$(2\ell_M + 6) \cdot \mathbb{G}$	$2 \cdot \mathbb{G} + 3 \cdot \mathbb{Z}_p$	$O(q_{\mathcal{A}})$	DDH	SL	NP,NO
Ours (§4.2)	$25 \cdot \mathbb{G}$	$16 \cdot \mathbb{G} + 22 \cdot \mathbb{Z}_p$	$O(\log q_{\mathcal{A}})$	DDH	ESL	NP,NO

Fig. 2: Group-based (but pairing-free) signature schemes in the random oracle model. When calculating the sizes $\lvert pk\rvert$ and $\lvert\sigma\rvert$ of public key and signature, we assume that a generator of $\mathbb{G}$ is publicly known, and we let $\varepsilon_{\mathcal{A}}$ and $q_{\mathcal{A}}$ denote the success probability and number of signing queries of a given adversary $\mathcal{A}$. ℓ_M denotes the (bit)length of the signed message. In the "assumption" column, "AGM"="Algebraic Group Model" [FKL18], "MBDL"="Multi-Base Discrete Logarithm problem" (an interactive but falsifiable assumption that holds generically), "CDL"="Circular Discrete Logarithm problem" (a non-interactive assumption that holds generically). "SL" and "ESL" stand for "Schnorr-like" and "Extended Schnorr-like", and an asterisk (as in SL*) means that the scheme must be syntactically (slightly, and without affecting the security analysis) altered to make it formally SL/ESL. "ROM" denotes the type of random oracle used in the analysis: (**N**on-)**P**rogrammable, (**N**on-)**O**bservable. We note that [RS21] also provide a security analysis of Schnorr's scheme that only requires non-programmable (but observable) random oracles, but makes nonstandard assumptions about the used group (and whose loss cannot easily be expressed as above).

a particular design paradigm. [FHJ20] also uses a meta-reduction, but with different ideas than ours (and in particular they do not require a technique similar to our filtering).

In a similar vein, [KMP16] show and relate new and known upper and lower bounds for Fiat-Shamir-based signature schemes. Their results cover the Schnorr scheme specifically, and are independent of the used mathematical platform (such as groups, RSA, or lattices). They consider both programmable and non-programmable random oracles, but their lower bounds only apply to a particular class (i.e., Fiat-Shamir-based) of signature schemes.

A number of works use meta-reductions to provide lower bounds for Schnorr signatures specifically (e.g., [PV05, GBL08, Seu12]). The most general result [FJS14] in this line of works shows that Schnorr signatures cannot be proven tightly secure in a generic group model with a programmable (and observable) random oracle from any non-interactive assumption. They also show that with only a non-programmable (observable) random oracle, no reduction at all (tight or not) exists for the Schnorr scheme. Their results are incomparable to ours, since they show stronger results, but only for a single signature scheme. Their techniques, and in particular the way in which they design ideal and simulated forgers, are quite different from ours.

Follow-up work. We remark that the techniques from our constructive results (Section 4.1) have also proved useful in the context of message franking schemes [GTH26].

Roadmap. After a technical overview in 2 and some preliminaries in 3, we present our main result in 5. We present our non-tightly secure Schnorr-like signature scheme in 4.1, and our tightly secure extended Schnorr-like construction in 4.2.

2 Technical overview

We are interested in a computational model that is *just* expressive enough to construct secure (group-based) signatures. We start by describing this model, consisting of a generic group and a random oracle.

Generic group. We assume a generic group [Nec94, BL96, Sho97, Mau05, Zha22] in the formalization of Maurer [Mau05].[5] Intuitively, this means that every algorithm has access to an abstract prime-order group, but only through explicit oracle queries (for group operation and equality tests).

In Maurer's formalization, there exist no binary representations of group elements. An algorithm only gets a "reference" supplied by the group oracle for each group element that is operated on. This reference is generally different for different users of the group oracle. This stands in contrast to e.g. Shoup's model [Sho97], where group elements have random encodings. The issue with such a model is that the encodings yield a source of entropy and can be exploited to obtain an unlimited random oracle functionality, hence immediately allowing to construct OWF-based signature schemes. These subtleties are the reason why our result (in line with existing impossibility results on purely group-based signatures [DHH+21, CFGG22]) are all based on Maurer's model.

Random oracle. Since the group oracle alone is insufficient to construct secure signatures [DHH+21] (see also [CFGG22, Theorem 3]), we additionally consider a random oracle. This means that all algorithms have oracle access to a function $\mathcal{O}_{\mathsf{H}}$ that takes as input any combination of group elements and bitstrings, and outputs a truly random bitstring. Because our goal is to find a meaningful dividing line between secure and insecure signature schemes, we model $\mathcal{O}_{\mathsf{H}}$ in a very weak, non-programmable and non-observable form. That means that a security reduction can neither influence (or "program") $\mathcal{O}_{\mathsf{H}}$ on particular inputs, nor even "see" inputs $\mathcal{O}_{\mathsf{H}}$-queries made by an internally simulated adversary.[6]

Schnorr-like signatures. The "Schnorr-like" signature schemes we study have a restricted verification procedure. Specifically, say (without loss of generality) that the public keys and signatures are of the form $\mathrm{pk} = (\mathbf{g}, s)$ and $\sigma = (\mathbf{h}, t)$, where $\mathbf{g}$ and $\mathbf{h}$ are vectors of group elements, and s and t are arbitrary bitstrings. Then, we mandate that verification, on input pk, σ, and a (bitstring) message m, consists of a single check of the form

$$A(s, m, r, t) \cdot \mathbf{g} \;=\; \mathbf{h}, \tag{1}$$

where $r = \mathcal{O}_{\mathsf{H}}(C(s, m, t) \cdot \mathbf{h}, f(s, m, t))$, and A, C, and f are arbitrary (but fixed) functions. In other words, verification makes only a (potentially large) number of non-adaptive group equality checks, whose coefficients may involve the result r of a single hash query. We call such signatures "Schnorr-like", since Schnorr signatures are of this form.

Note that (1) already allows to express the verification of non-interactive zero-knowledge proofs derived from the Fiat-Shamir paradigm. This will be a great aid in our positive results.[7] Still, our lower bound will show that (1) is quite limiting, and we thus also define "extended Schnorr-type signatures", where

$$A(s, m, r, t) \cdot \mathbf{g} \;=\; B(s, m, r, t) \cdot \mathbf{h} \tag{2}$$

replaces form (1), using another arbitrary (but fixed) function B. We will exhibit an interesting signature scheme with a verification of the form (2).

[5] We remark that what we denote "Maurer's GGM" is in other references called the "type-safe GGM" [Zha22].
[6] Note $\mathcal{O}_{\mathsf{H}}$ is part of the machine model, not an external entity. Many types of random oracles have been studied (e.g., [Unr07, FLR+10, BDF+11, AB13, CJS14, FRS22]).
[7] However, *simulating proofs* is not possible with a non-programmable random oracle.

2.1 Impossibility of tightly secure Schnorr-like signature schemes

We start with an overview of the techniques used in our main result: the impossibility of Schnorr-like signature schemes tightly reducible to any assumption that holds in the GGM. (See Section 5 for a formal statement and proof.)

Meta-reduction strategy. Suppose there is some generic reduction R (treating the random oracle as *non-observable and non-programmable*) that reduces EUF-CMA security of some Schnorr-like signature scheme Σ to an assumption that holds in the GGM. This means that for any generic (and even unbounded[8]) adversary A that has non-negligible advantage in the EUF-CMA game, R^{A} breaks the underlying (generically true) assumption.

Our impossibility result excludes the existence of such a reduction R. To this end, we will follow the meta-reduction technique [Cor00]. This requires first defining some adversary A that has non-negligible EUF-CMA advantage, which guarantees us that R^{A} breaks the underlying assumption. Second, we need to transform R^{A} into a so-called *meta-reduction* M, and prove that M breaks the assumption whenever R^{A} does. The crucial point is that this argument does not require the adversary A to be efficient: as long as the meta-reduction M is efficient, the assumption is broken and hence one of the premises (i.e., existence of a reduction R or validity of the assumption itself) must be false.

On a high level, we leverage the following idea: Recall that the reduction R must be *non-programming*, i.e., it may only access the RO $\mathcal{O}_{\mathsf{H}}$ explicitly through some provided interface. This interface is made available only to the reduction R and to the adversary A that R is interacting with. Normally, the interface would just keep state of a uniformly random function, and answer all queries from R and A accordingly. However, the meta-reduction M only needs to be an adversary against the underlying assumption, *which is a pure group-based assumption that is true in the GGM and cannot refer to the RO in any way.*

Therefore, M entirely takes control of the random oracle, which we can use to our advantage: say we first construct an *unbounded* adversary $\mathsf{A}_{\mathsf{unbounded}}$ against EUF-CMA security, which accesses the RO in the same way as reduction R. We know that $\mathsf{R}^{\mathsf{A}_{\mathsf{unbounded}}}$ (which, so far, makes a super-polynomially number of group operations) breaks the underlying group assumption. Then, we can construct another *programming* adversary $\mathsf{A}_{\mathsf{prog}}$ against EUF-CMA, which has the capability to program the RO (but is *not* unbounded anymore). Then, $\mathsf{M} := \mathsf{R}^{\mathsf{A}_{\mathsf{prog}}}$ is a well-formed meta-reduction. As long as we show that $\mathsf{R}^{\mathsf{A}_{\mathsf{prog}}}$ and $\mathsf{R}^{\mathsf{A}_{\mathsf{unbounded}}}$ behave statistically very close (including the behavior of the RO itself), then we successfully broke the underlying assumption using $\mathsf{R}^{\mathsf{A}_{\mathsf{prog}}}$. See Figure 3.

We point out that [FHJ20] follows a similar high-level strategy to show that signature schemes using "parallel-OR techniques" cannot be proven secure using a non-programming reduction R. They also define a meta-reduction M that programs the ROM, but their techniques for constructing M are entirely different from ours.

In the following, for simplicity we assume that in (1), $C(s, m, t) = I$ is always the identity matrix and we always have $f(s, m, t) = m$. In the main body, we show the more general result for any C and f.

Strawman Adversaries. Designing successful adversaries $\mathsf{A}_{\mathsf{unbounded}}$ and $\mathsf{A}_{\mathsf{prog}}$ *by themselves* is indeed very simple. Each of them has some "superpower" that makes forging easy:

- $\mathsf{A}_{\mathsf{unbounded}}$, given public key $\mathsf{pk} = (\mathbf{g}, s)$, would first break the dlog of $\mathbf{g}$, i.e., compute $\mathbf{x} \in \mathbb{Z}_p^{|\mathbf{g}|}$ with $\mathbf{g} = \mathbf{x} \cdot \mathsf{g}_0$. Given some message m, $\mathsf{A}_{\mathsf{unbounded}}$ needs to pick some vector $\mathbf{h} := \mathbf{y} \cdot \mathsf{g}_0$ suitably, s.t. for

[8] Unbounded here refers to the number of group operations, but *not* the number of calls to the random oracle. A reduction in the *non-observable and non-programmable* ROM should not be required to be successful given an adversary with super-polynomially many RO-calls. Otherwise it would have no leverage over the adversary it is interacting with and the random oracle appears to be useless altogether.

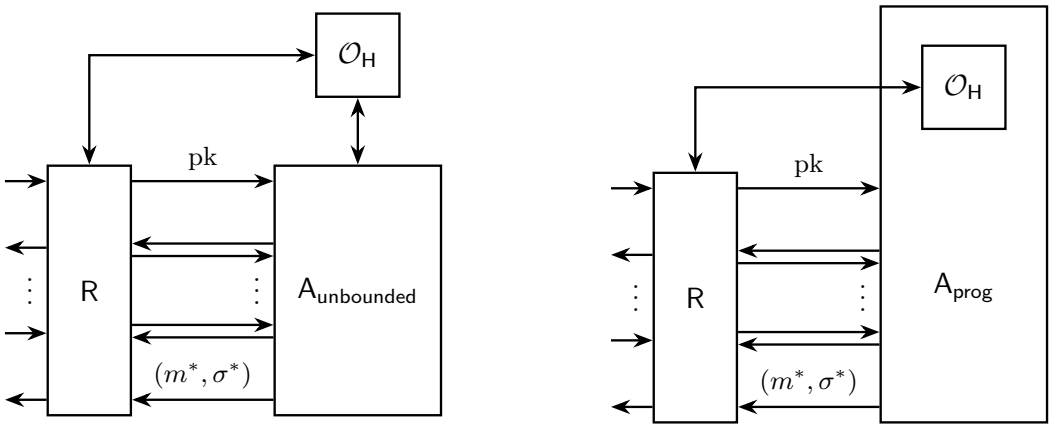

Fig. 3: Illustration of the two adversaries in our meta-reduction.

the corresponding RO output $r \leftarrow \mathcal{O}_\mathsf{H}(\mathbf{h}, m)$, the existence of some t with $A(s, m, r, t) \cdot \mathbf{x} = \mathbf{y}$ is likely. The forged signature for m will be $\sigma := (\mathbf{h}, t)$.
The success of this strategy depends on whether a "suitable" choice of $\mathbf{h}$ even exists. This is guaranteed by correctness, i.e., the existence of some (to $\mathsf{A}_\mathsf{unbounded}$ unknown) secret key sk for which $\textsc{Sign}(\mathrm{sk}, m)$ always produces valid signatures.

- A_prog, given public key $\mathrm{pk} = (\mathbf{g}, s)$, is even simpler: pick any message m, any bitstring t, some "RO output" r, and compute $\mathbf{h} := A(s, m, r, t) \cdot \mathbf{g}$. Programming $\mathcal{O}_\mathsf{H}(\mathbf{h}, m) := r$ makes the signature $\sigma := (\mathbf{h}, t)$ valid.

Unfortunately, these two strawman adversaries do not necessarily behave similarly (since their respective distributions of forgeries may differ), which is necessary for a successful meta-reduction. Resolving this issue is the main challenge and technical contribution of our lower bound.

"Suitable" group elements. Given a message m, let us call a vector $\mathbf{h}$ of group elements *suitable* for $\mathrm{pk} = (\mathbf{g}, s)$, if

$$\text{for at least } \frac{|\mathcal{R}_\mathsf{H}|}{\mathrm{poly}_\mathsf{bound}(\lambda)} \text{ choices of } r \in \mathcal{R}_\mathsf{H} : \exists t \text{ s.t. } \mathbf{h} = A(s, m, r, t) \cdot \mathbf{g} \; .$$

Here, $\mathrm{poly}_\mathsf{bound}(\lambda)$ is some (for now unspecified) polynomial, and $\mathcal{R}_\mathsf{H}$ denotes the output space of the random oracle.

Note that $\mathsf{A}_\mathsf{unbounded}$ is very unlikely to produce a signature $\sigma = (\mathbf{h}, t)$ for a message m where $\mathbf{h}$ is unsuitable (because $\mathsf{A}_\mathsf{unbounded}$ receives a uniformly random $r \in \mathcal{R}_\mathsf{H}$ from the RO *after* picking $\mathbf{h}$, and hence by unsuitability it is unlikely that a t exists that makes the signature valid). On the other hand, A_prog cannot easily check whether some $\mathbf{h}$ is suitable without breaking the dlog of $\mathbf{g}$; at the same time, it is very easy for A_prog to produce a signature containing $\mathbf{h}$. Hence, in our meta-reduction, we must find a strategy to ensure that $\mathsf{A}_\mathsf{unbounded}$ and A_prog use the same "distribution" of $\mathbf{h}$'s. This requires making signature queries.

Characterizing the Adversary's Distribution. We start with the observation that, given a fixed RO $\mathcal{O}_\mathsf{H}$, there is a one-to-one correspondence between

$$\begin{aligned} &\text{forgeries } (m, \sigma = (\mathbf{h}, t)) \quad \text{with} \quad \mathbf{h} = A(s, m, \mathcal{O}_\mathsf{H}(\mathbf{h}, m), t) \cdot \mathbf{g} \quad \text{and} \\ &\text{triples } (m, r, t) \quad \text{with} \quad r = \mathcal{O}_\mathsf{H}(A(s, m, r, t) \cdot \mathbf{g}, m) \; . \end{aligned}$$

Hence, instead of talking about forgeries, we will discuss how to make the distributions of triples (m, r, t) generated by $\mathsf{A}_{\mathsf{prog}}$ and $\mathsf{A}_{\mathsf{unbounded}}$ close, while they must always always satisfy $r = \mathcal{O}_{\mathsf{H}}(A(s, m, r, t) \cdot \mathbf{g}, m)$.

Main goal. Let us denote by $\mathcal{T}$ the set of all triples (m, r, t), and for convenience let $\mathcal{T}_m := \{(r, t) \mid (m, r, t) \in \mathcal{T}\}$ denote the pairs (r, t) corresponding to m. As a basic idea, we want our adversaries to *filter* the set $\mathcal{T}$ (i.e., delete elements from it) in such a way that afterwards it mostly contains "suitable" triples (m, r, t). In this context, with a triple (m, r, t) being suitable w.r.t. $\mathcal{T}$ and $\mathrm{pk} = (\mathbf{g}, s)$ we mean that[9]

$$\begin{aligned} &\text{for at least } \frac{|\mathcal{R}_{\mathsf{H}}|}{\mathrm{poly}_{\mathsf{bound}}(\lambda)} \text{ choices of } r' \in \mathcal{R}_{\mathsf{H}} : \\ &\exists t' \text{ s.t. } A(s, m, r, t) \cdot \mathbf{g} = A(s, m, r', t') \cdot \mathbf{g} \text{ and } (m, r', t') \in \mathcal{T} \ . \end{aligned} \tag{3}$$

In general, without knowing the dlog of $\mathbf{g}$, one cannot check whether a triple (m, r, t) is suitable using a polynomial number of group operations.

One challenge with computing a set $\mathcal{T}$ that fulfills the above is that suitability of a triple $(m, r, t) \in \mathcal{T}$ not only depends on values m, r, t, but even on all *other triples in* $\mathcal{T}$. Hence, when finding $\mathcal{T}$, we need to strike a balance between (1) filtering out unsuitable triples while simultaneously (2) not removing too many triples that are required to keep other triples suitable.

Jumping ahead, if we could find such a filtered set $\mathcal{T}$ containing only suitable triples w.r.t. $\mathcal{T}$ and pk, then there is hope to construct two adversaries $\mathsf{A}_{\mathsf{prog}}$ and $\mathsf{A}_{\mathsf{unbounded}}$ with similar behavior: both adversaries first compute $\mathcal{T}$, and then use the following strategies for sampling (m, r, t):

$$\begin{array}{ll} \underline{\mathsf{A}_{\mathsf{prog}}}: & \underline{\mathsf{A}_{\mathsf{unbounded}}}: \\ m \leftarrow\!\!\$\ \mathcal{M} & m \leftarrow\!\!\$\ \mathcal{M} \\ (r, t) \leftarrow\!\!\$\ \mathcal{T}_m & (r', t') \leftarrow\!\!\$\ \mathcal{T}_m \\ \mathcal{O}_{\mathsf{H}}(A(s, m, r, t) \cdot \mathbf{g}, m) := r & r \leftarrow \mathcal{O}_{\mathsf{H}}(A(s, m, r', t') \cdot \mathbf{g}, m) \\ & \text{find } t \text{ with } (m, r, t) \in \mathcal{T} \\ & \quad \text{and } A(s, m, r', t') \cdot \mathbf{x} = A(s, m, r, t) \cdot \mathbf{x} \end{array} \tag{4}$$

Note that $\mathsf{A}_{\mathsf{unbounded}}$ is allowed to use $\mathbf{x}$ because it can break the dlog of $\mathbf{g}$. Strictly speaking, $\mathsf{A}_{\mathsf{unbounded}}$ also needs to repeat its steps if no t was found, but this only happens with inverse-polynomial probability due to the "suitability" property. (The two distributions are still not entirely identical due to more subtle reasons. This will be addressed later.)

Learning Suitable Triples. So how can we filter $\mathcal{T}$ efficiently (in terms of number of group operations) without knowing the dlog of $\mathbf{g}$, i.e., without being able to check suitability of triples? The adversaries will first attempt to find a set $\mathcal{G} \subseteq \mathbb{Z}_p^{|\mathbf{g}|}$ of *plausible* dlogs of $\mathbf{g}$ (where, importantly, the real dlog should remain in $\mathcal{G}$). Then, we will filter the set $\mathcal{T}$ so that its triples are suitable not only w.r.t. pk, but w.r.t. *all plausible public keys* $\mathrm{pk} = (\mathbf{g}', s)$ (i.e,. $\mathbf{g}' = \mathbf{x}' \cdot \mathsf{g}_0$ for some $\mathbf{x}' \in \mathcal{G}$).

To find $\mathcal{G}$, the main insight is that not only $\mathsf{A}_{\mathsf{unbounded}}$, but also the *signing oracle is incapable of producing a triple* (m, r, t) *that is unsuitable* w.r.t. the initial set $\mathcal{T}$ containing all triples. In particular, if we query $\textsc{Sign}(m)$ for some $m \leftarrow\!\!\$\ \mathcal{M}$, and the signing oracle (i.e., either the honest signer, or the reduction R) makes at most q_{H} RO queries, then with probability no more than $q_{\mathsf{H}}/\mathrm{poly}_{\mathsf{bound}}(\lambda) + \mathsf{negl}(\lambda)$, its returned signature σ induces a triple (m, r, t) that is unsuitable w.r.t. the set $\mathcal{T}$ of all possible triples and pk. (The term $\mathsf{negl}(\lambda)$ comes from the fact that the signing oracle may have already made an RO query $\mathcal{O}_{\mathsf{H}}(\mathbf{h}, m)$ before we even sampled $m \leftarrow\!\!\$\ \mathcal{M}$.)

Given this, any adversary can (using only a polynomial number of signature queries and RO queries) compute a set $\mathcal{G} \subseteq \mathbb{Z}_p^{|\mathbf{g}|}$ by making many signature queries m and deleting all dlog-candidates

[9] The difference to before is that suitability now asserts the existence of sufficiently many RO outputs r' for which the corresponding triple still is *in the set* $\mathcal{T}$.

$\mathbf{x}'$ from $\mathcal{G}$ for which $A(s,m,r,t)\cdot\mathbf{x}'\cdot\mathsf{g}_0$ would be unsuitable w.r.t. $\mathcal{T}$ and public key $(\mathbf{x}'\cdot\mathsf{g}_0, s)$ (where (m,r,t) is the triple corresponding to the obtained signature σ). When performing k such deletion steps (for independent messages), then with probability $1-k\cdot q_{\mathsf{H}}/\mathrm{poly}_{\mathsf{bound}}(\lambda)-\mathsf{negl}(\lambda)$, the *real* dlog of $\mathbf{g}$ is still contained in $\mathcal{G}$.

After obtaining the set $\mathcal{G}$ as above, we can make progress towards filtering the set $\mathcal{T}$ of suitable triples: Let us start with $\mathcal{T}$ containing *all* triples (m,r,t). Then we pick *any* of the remaining dlog-candidates, say $\mathbf{x}'\in\mathcal{G}$, and update $\mathcal{T}$ to be the following set $\mathcal{T}'$:

$$\mathcal{T}' := \{(m,r,t) \mid (m,r,t) \text{ suitable w.r.t. } \mathcal{T} \text{ and } \mathrm{pk}' := (\mathbf{x}'\cdot\mathsf{g}_0, s)\} \tag{5}$$

Iterating. The above process must be iterated as follows, as there is no guarantee that $\mathcal{T}$ is well-filtered after a single step: Initialize $\mathcal{G}=\mathbb{Z}_p^{|\mathbf{g}|}$ and $\mathcal{T}$ to be the set of all triples. Perform deletions on $\mathcal{G}$, and use this to update $\mathcal{T}$ as in (5), provided that there exists some plausible dlog-candidate $\mathbf{x}'\in\mathcal{G}$ for which not sufficiently many triples in $\mathcal{T}$ are suitable. Then perform deletions on $\mathcal{G}$ again, and so on, until all dlog-candidates in $\mathcal{G}$ are such that most (say, a $1-1/\mathrm{poly}_{\mathsf{filter}}(\lambda)$ fraction) triples in $\mathcal{T}$ are suitable for them.

Note that by deleting a $1/\mathrm{poly}_{\mathsf{filter}}(\lambda)$ fraction of triples each round, this process is guaranteed to terminate after a polynomial number of steps. Equipped with this filtering strategy, two things remain to be shown for our meta-reduction argument: (1) $\mathsf{A}_{\mathsf{unbounded}}$, if it employs the filtering process above and then samples triples from $\mathcal{G}$ as per (4), indeed breaks EUF-CMA security, and (2) $\mathsf{R}^{\mathsf{A}_{\mathsf{prog}}}$ and $\mathsf{R}^{\mathsf{A}_{\mathsf{unbounded}}}$ are statistically close.

Success of $\mathsf{A}_{\mathsf{unbounded}}$. Consider $\mathsf{A}_{\mathsf{unbounded}}$ engaging in an EUF-CMA game, which in particular means that the signing oracle is honest. The only reason why $\mathsf{A}_{\mathsf{unbounded}}$ may lose the game is when $\mathcal{T}$ becomes empty during filtering, in which case it wouldn't have anything to sample from.

Looking at a single iteration of the filtering process, suppose we perform $\frac{\lambda+\ell_{\mathsf{g}}\cdot\lceil\log p\rceil}{\mathrm{poly}(\lambda)}$ deletions on $\mathcal{G}$ (for independent random messages $m\leftarrow\!\!\$\ \mathcal{M}$) before moving on to updating $\mathcal{T}$. Then, with overwhelming probability (taken over the attack), we will still have

$$\Pr\left[(m,r,t)\in\mathcal{T}\;\middle|\;\begin{array}{l} m\leftarrow\!\!\$\ \mathcal{M}\\ \sigma\leftarrow \textsc{Sign}^{\mathcal{O}_{\mathsf{H}}}(\mathrm{sk},m)\end{array}\right]\geq 1-\frac{1}{\mathrm{poly}(\lambda)}$$

(This probability treats $\mathcal{G}$ and $\mathcal{T}$ as fixed. It is *not* taken over the attack, but only over the triple-sampling procedure that corresponds to Sign.)

This bound shows "soundness" of our $\mathcal{G}$-deletion technique given an honest signer, in the sense that most triples (m,r,t) that would ever be produced by the signing oracle are likely to remain in $\mathcal{T}$. This, in turn, means that $\mathcal{T}$ is likely to remain non-empty after filtering, even when repeated for many steps.

Statistical closeness. It remains to argue that $\mathsf{R}^{\mathsf{A}_{\mathsf{prog}}}$ and $\mathsf{R}^{\mathsf{A}_{\mathsf{unbounded}}}$ are statistically close. Once the filtering process converges, we obtain a state of $\mathcal{T}$ where most (i.e., a $1-1/\mathrm{poly}_{\mathsf{filter}}(\lambda)$ fraction) of its triples are suitable for all remaining dlog-candidates in $\mathcal{G}$, which holds in particular even for the real dlog of $\mathbf{g}$ with sufficiently large probability.

Intuitively, such a $\mathcal{T}$ should allow us to then sample triples in both adversaries in a way that differs statistically by at most $1/\mathrm{poly}_{\mathsf{filter}}(\lambda)$ as per (4), i.e., by having both adversaries construct signatures that induce a roughly uniform distribution over $\mathcal{T}$. However, there are some details that make the sampling strategy for $\mathsf{A}_{\mathsf{unbounded}}$ quite technical. In fact, in the main body we will need an even stronger notion of suitability to ensure that for each (m,r,t), there are many r' for which the number of t' resulting in the same $\mathbf{h}$ is almost the same as for r. Furthermore, we require *artificial*

aborts for ensuring that our sampling strategy for $\mathsf{A}_{\mathsf{unbounded}}$ is indeed uniform over $\mathcal{T}$. See Section 5 for details.

Note that we can now see why the lower bound only applies to *tight* reductions: the number of required signature queries (needed for deletions in $\mathcal{G}$) increases with our desired $\mathrm{poly}_{\mathsf{filter}}(\lambda)$. However, a non-tight reduction can have an advantage $\mathrm{Adv}_{\mathsf{R}^{\mathsf{A}_{\mathsf{unbounded}}}}$ against its underlying game that decreases with the number of signature queries, and therefore the statistical distance $1/\mathrm{poly}_{\mathsf{filter}}(\lambda)$ between $\mathsf{R}^{\mathsf{A}_{\mathsf{prog}}}$ and $\mathsf{R}^{\mathsf{A}_{\mathsf{unbounded}}}$ could be larger than $\mathrm{Adv}_{\mathsf{R}^{\mathsf{A}_{\mathsf{unbounded}}}}$, making our meta-reduction argument void.

Necessity of restrictions. While our filtering techniques could potentially be of independent interest, it cannot be applied to verification that has the more general form (2). (In fact, it *must* fail due to the upper bound presented in Section 2.2.) The reason is that previously, the suitability notion (3) implicitly partitioned $\mathcal{T}$'s triples into equivalence classes with a one-to-one correspondence to RO inputs $(\mathbf{h}, m)$, which is crucial for efficient sampling inside $\mathsf{A}_{\mathsf{unbounded}}$ (because it allows to first pick the RO input $(\mathbf{h}, m)$, and then after receiving r' to find a triple (m, r, t) from the corresponding equivalence class). Transitivity would not hold for the general form (2) anymore.

We also point out that we indeed require R to be *non-observing*: Our sampling strategy inside $\mathsf{A}_{\mathsf{unbounded}}$ makes many RO queries that may leak something about about the dlog of pk. Thus, we cannot show $\mathsf{A}_{\mathsf{unbounded}}$ and $\mathsf{A}_{\mathsf{prog}}$ to behave statistically close given an observing reduction.

2.2 Possibility of (extended) Schnorr-like signature schemes

To complement our lower bound just described, we present a Schnorr-like signature scheme with a *loose* security reduction, and a tightly secure *extended* Schnorr-like scheme. Our schemes can be seen as pairing-free variants of Waters signatures [Wat05], resp. the structure-preserving signature scheme from [AHN+23].

We begin with an overview of the Schnorr-like scheme with loose security reduction. (Its formal version can be found in Section 4.1.)

Waters' signature scheme. To outline our strategy, we recall Waters' scheme. It uses pairings only to check a single pairing equation during verification. Specifically, public keys and signatures have the form

$$\mathrm{pk} = (\mathsf{g}^{\alpha}, \mathsf{g}^{\beta}, \mathsf{h}_0, \ldots, \mathsf{h}_{\ell}) \quad \text{and} \quad \sigma = (\sigma_1, \sigma_2) = (\mathsf{g}^{r}, (\mathsf{h}_0 \cdot \prod_{i=1}^{\ell} \mathsf{h}_i^{m_i})^{r} \cdot \mathsf{g}^{\alpha\beta}),$$

where the signed message is $m = (m_1, \ldots, m_{\ell}) \in \{0,1\}^{\ell}$. Using a suitable pairing equation, verification checks that signatures are indeed of this form. Note that validity of signatures means that for the discrete logarithms α, β, r of $\mathsf{g}^{\alpha}, \mathsf{g}^{\beta}, \sigma_1$, we have $\sigma_2 = \mathsf{h}_m^{r} \cdot \mathsf{g}^{\alpha\beta}$, where h_m is a shorthand for the publicly computable element $\mathsf{h}_0 \cdot \prod_{i=1}^{\ell} \mathsf{h}_i^{m_i}$. Since this equation is linear for a fixed public key, we can replace this check with a Fiat-Shamir-based non-interactive zero-knowledge proof (which in turn can be checked without pairings, but with an equation like (1)).

Difficulty: no programming, no (easy) simulation. A similar idea has been used to replace pairings with random oracle queries in the context of blind signatures [CATZ24, KRW24, BHKR25] (building on ideas from [GJ03, Che05]). One difference in our setting is the security reduction: we cannot directly simulate zero-knowledge proofs by programming the random oracle (since our computational model does not allow such programming). Instead, we rely only on the witness-indistinguishability of our proof system, much like (plain) Groth-Sahai proofs [GS08].

In fact, we propose an abstraction of the Groth-Sahai commit-and-prove system from [GS08], and instantiate this abstraction without pairings, and in the (non-programmable, non-observable) random

oracle model.[10] Our proof system supports arbitrary disjunctions of (systems of) linear statements over the group, and has a verification that is compatible with (2). Hence, we can use our proof system in the construction of extended Schnorr-like signatures (such that signature verification may check proofs). Moreover, for "quasi-adaptive" languages [JR13] which are determined at setup time, verification is even compatible with (1), and we can construct Schnorr-like signature schemes from our proof system.

Equipped with a witness-indistinguishable commit-and-prove system as above, we can prove, e.g., that a given signature has been produced with a secret key (using as witness α, β, r). However, it is not clear how to simulate such non-interactive proofs in a security reduction. (Indeed, Waters' reduction from [Wat05] can produce valid signatures, but would not be able to supply any of the corresponding witnesses α, β, r.)[11]

Our solution is to simplify Waters' scheme, and align its signatures closer to the simulation of [Wat05]. Concretely, in our scheme, pk consists of a number of commitments to exponents $a_0, \ldots, a_\ell$, and a signature consists (only) of a witness-indistinguishable proof to the fact that $a_m := a_0 + \sum_{m_i=1} a_i \neq 0$.[12] Since the involved commitments to the a_i can be publicly and homomorphically turned into a commitment to a_m, this amounts to only proving a linear equality about the content of a commitment, an exponent. Hence, this results in a compact signature that conforms to our property (1) for Schnorr-likeness.

In our scheme, we will set up $a_0 = 1$ and $a_i = 0$ for $i > 0$, so that $a_m = 1 \neq 0$, and we can sign arbitrary messages. In our security reduction, we will set up the a_i differently, such that (with probability proportional to $1/q$), we can sign q messages, while being unable to sign a potential forgery m^*. This switch in the setup of the a_i is undetectable thanks to the witness-indistinguishability of our proof system, and allows to prove unforgeability of the signature scheme. The details, and in particular a suitable distribution of the a_i, are conceptually identical to the proof strategy of [Wat05], see also [HK08]. We omit the details in this overview. However, we note that this proof strategy is generally well-understood, and lower bounds exist for the corresponding reduction loss [Cor00, HJK12].

Extended Schnorr-like signatures with a tight security reduction. We also offer a second signature scheme, which enjoys a *tight* security reduction, but only satisfies extended Schnorr-like verification as in (2). Our strategy is again very similar, but this time applied to the more complex, structure-preserving signature scheme of [AHN+23].[13] For space reasons, and because this scheme is conceptually similar to the one from [AHN+23] (instantiated with a different proof system), we omit the details here and refer the reader to Section 4.2.

3 Preliminaries

Throughout the paper, we use standard cryptographic notation and terminology, such as λ to denote a security parameter, "ppt" for "probabilistic polynomial-time", $\mathsf{negl}(\lambda)$ for a negligible function, and $\approx_{f(\lambda)}$ to denote statistical distance of at most $f(\lambda)$ between two distributions.

We use $y \leftarrow A(x)$ to denote the execution of (potentially randomized) algorithm A on input x with output y, and we use $y \leftarrow\!\!\$\ \mathcal{D}$ to denote that y is sampled uniformly at random from distribution

[10] We note that [BHKR25] follow a very similar strategy, also with the intent to convert pairing-based signature schemes to pairing-free ones as in [CATZ24], and also using the abstraction of commit-and-prove systems. They rely on a programmable random oracle, however, and their schemes are (beyond of course being *blind* signature schemes) far from Schnorr-like.

[11] There are known ways to convert witness-indistinguishable proof systems into zero-knowledge proofs [FS90, Gro06]. However, the properties needed in our case (and in particular simulation-soundness) appear to require signature techniques [Gro06], which likely result in more complex verification equations of the form (2). We do not claim that this strategy cannot work; but here we choose a different, simpler route.

[12] This is not a linear equation, but we can transform it into one by requiring that there is a scalar w with $a_m \cdot w = 1$.

[13] This scheme is also the basis for the tightly secure, pairing-free blind signature scheme of [BHKR25].

$\mathcal{D}$. The notation $A^{O}(x)$ means that the execution of A has oracle access to a (possibly stateful) O. In some cases, an oracle may also alter the state of other oracles, e.g., when writing $A^{\mathcal{O}_{\mathsf{H}},B^{\mathcal{O}_{\mathsf{H}}}}(x)$, then B potentially alters the internal state of $\mathcal{O}_{\mathsf{H}}$ (for example, this will be the case when B programs the random oracle $\mathcal{O}_{\mathsf{H}}$).

3.1 The Model

We use $\mathbb{G}$ to denote an (arbitrary but fixed) group of prime order $p > \lambda^{\omega(1)}$ (written in *additive* notation) with generator g_0, and neutral element $\mathsf{0}$. We distinguish group elements from other objects using sans-serif font, e.g. $\mathsf{g}, \mathsf{h}, \ldots$ A group elements vector is additionally boldface, e.g. $\mathbf{g}, \mathbf{h}, \ldots$

Definition 1 (Maurer's Generic Group Model). *A* generic *algorithm* A *in Maurer's Generic Group Model (GGM) is an algorithm with special properties: When* A *is run, we first initialize a GGM environment that contains a counter* k *and a table* T *to* $k = 0$ *and* $T[0] = \mathsf{g}_0$*,* $T[i] = \bot$ *(for* $i > 0$*). The algorithm gets access to the following two oracles:*

- ***Group operation oracle*** $\mathcal{O}_{\mathsf{grp}}(i, j)$*: Given integers* $i, j \in \mathbb{N}$*, this oracle returns* $\bot$ *if* $T[i] = \bot$ *or* $T[j] = \bot$*. Otherwise, it sets* $k := k + 1$*, stores the group element* $T[i] + T[j]$ *in* $T[k]$*, and returns* k*.*
- ***Group equality oracle*** $\mathcal{O}_{\mathsf{eq}}(i, j)$*: Given integers* $i, j \in \mathbb{N}$*, this oracle returns* 1 *if the two elements* $T[i]$ *and* $T[j]$ *are equal, or* 0 *otherwise.*

We call an integer i *a "label" of some group element* $\mathsf{g} \in \mathbb{G}$ *if* $T[i] = \mathsf{g}$*.*

The generic algorithm A *may accept inputs (either at the beginning of the algorithm or later when it interacts with other algorithm) of the form* $(\mathbf{g}, x)$*, where* $\mathbf{g} \in \mathbb{G}^{\ell}$ *are group elements and* $x \in \{0,1\}^*$ *is a bitstring. However, the GGM environment* converts *the input by copying* $\mathbf{g}$ *into the table (*$T[k+i] := \mathbf{g}[i]$ *for all* $i \in [\ell]$*; and define a vector of labels* $\mathbf{v} \in \mathbb{N}^{\ell}$ *s.t.* $\mathbf{v}[i] = k + i$*, and updating* $k := k + \ell$*). Finally, what the algorithm* A *receives is the pair* $(\mathbf{v}, x)$*, i.e., only labels in place of group elements.*

Similarly, A *may produce outputs of the form* $(\mathbf{v}, x) \in \mathbb{N}^{\ell} \times \{0,1\}^*$*, which is first converted by the GGM environment into* $(\mathbf{g}, x)$*, with group elements* $\mathbf{g} \in \mathbb{G}^{\ell}$ *and* $\mathbf{g}[i] = T[\mathbf{v}[i]]$*, before being passed on to the recipient.*

If two or more generic algorithms are run at the same time, each of them has their own individual GGM environment. In particular, when P_1 *sends a message* $(\mathbf{v}, x) \in \mathbb{N}^* \times \{0,1\}^*$ *to another algorithm* P_2*, the environment of* P_1 *first converts it into a pair* $(\mathbf{g}, x) \in \mathbb{G}^* \times \{0,1\}^*$*, and* P_2 *then receives a pair* $(\mathbf{v}', x) \in \mathbb{N}^* \times \{0,1\}^*$ *constructed by* P_2*'s environment.*

Definition 2 (Group Assumption). *Let* $\mathcal{P}$ *be some (not necessarily efficient or generic) algorithm, providing an interface to interact with some other algorithm* $\mathcal{A}$ *(the "adversary"), where each message being communicated is of the form* $(\mathbf{g}, x) \in \mathbb{G}^* \times \{0,1\}^*$*. We write* $\langle \mathcal{P}(\lambda), \mathcal{A}(1^{\lambda}) \rangle$ *to denote their joint execution, with* $\mathcal{P}$ *determining the final output (either* 0 *or* 1*). We call* $(\mathcal{P}, \epsilon)$*, for any* $0 \leq \epsilon < 1$*, an* assumption*.*

The assumption $(\mathcal{P}, \epsilon)$ *is* true in the standard model*, if for every ppt adversary* $\mathcal{A}$*, we have* $\mathrm{Adv}^{\mathcal{P}}_{\mathcal{A}}(\lambda) \leq \epsilon + \mathsf{negl}(\lambda)$*, where the advantage is*

$$\mathrm{Adv}^{\mathcal{P}}_{\mathcal{A}}(\lambda) := \Pr\left[\langle \mathcal{P}(\lambda), \mathcal{A}(1^{\lambda}) \rangle = 1\right] - \epsilon \,.$$

The assumption $(\mathcal{P}, \epsilon)$ *is* true in the GGM*, if for every generic adversary* A *in the GGM that makes at most* $\mathrm{poly}(\lambda)$ *queries to the group operation oracle* $\mathcal{O}_{\mathsf{grp}}$ *(and is otherwise unbounded),* $\mathrm{Adv}^{\mathcal{P}}_{\mathsf{A}}(\lambda) \leq \mathsf{negl}(\lambda)$*.*

Example 1. The assumptions $(\mathsf{DLog}, 0)$, $(\mathsf{DDH}, \frac{1}{2})$ are defined as follows.

$\mathsf{DLog}(\lambda)$	$\mathsf{DDH}(\lambda)$
$x \leftarrow\!\!\$\ \mathbb{Z}_p$	$x, y, z \leftarrow\!\!\$\ \mathbb{Z}_p$
send $x\mathsf{g}_0$	$b \leftarrow\!\!\$\ \{0,1\}$
receive x'	**if** $b = 0$, **send** $(x\mathsf{g}_0, y\mathsf{g}_0, z\mathsf{g}_0)$
return 1 iff $x = x'$	**else send** $(x\mathsf{g}_0, y\mathsf{g}_0, (xy)\mathsf{g}_0)$
	receive b'
	return 1 iff $b = b'$

Similarly, one can also capture interactive assumptions such as OM-DLog.

Definition 3 (Random Oracle with Group Input). *A Random Oracle (RO) for group $\mathbb{G}$ with output space $\mathcal{R}_\mathsf{H}$ is a uniformly random function $\mathcal{O}_\mathsf{H} : \mathbb{G}^* \times \{0,1\}^* \to \mathcal{R}_\mathsf{H}$.*

Note that for a generic *algorithm A with access to an RO $\mathcal{O}_\mathsf{H}$, any RO query by A is treated in the same way as any other output of A. In particular, from the perspective of A itself, it makes queries of the form $(\mathbf{v}, x) \in \mathbb{N}^* \times \{0,1\}^*$, but the environment converts the labels $\mathbf{v}$ into group elements $\mathbf{h}$ and then returns the answer $\mathcal{O}_\mathsf{H}(\mathbf{h}, x)$ to A. Consequently, for two or more interacting generic algorithms, the same RO $\mathcal{O}_\mathsf{H}$ will be used for all of them, despite associating the group elements with different labels.*

3.2 Signature Schemes and their Security

Definition 4 (Group-Based Signature Scheme in ROM). *A signature scheme $\Sigma = (\textsc{KeyGen}, \textsc{Sign}, \textsc{Verify})$ for message space $\mathcal{M} = \mathcal{M}(\lambda)$ consists of the following ppt algorithms with access to an RO $\mathcal{O}_\mathsf{H}$:*

- $\textsc{KeyGen}^{\mathcal{O}_\mathsf{H}}(1^\lambda)$ *takes security parameter $\lambda \in \mathbb{N}$ and returns a keypair $(\mathrm{sk}, \mathrm{pk})$ with $\mathrm{sk}, \mathrm{pk} \in \mathbb{G}^* \times \{0,1\}^*$.*
- $\textsc{Sign}^{\mathcal{O}_\mathsf{H}}(\mathrm{sk}, m)$ *takes secret key $\mathrm{sk} \in \mathbb{G}^* \times \{0,1\}^*$ and message $m \in \mathcal{M}$, and returns a signature $\sigma \in \mathbb{G}^* \times \{0,1\}^*$.*
- $\textsc{Verify}^{\mathcal{O}_\mathsf{H}}(\mathrm{pk}, m, \sigma)$ *takes public key $\mathrm{pk} \in \mathbb{G}^* \times \{0,1\}^*$, message $m \in \mathcal{M}$, and signature $\sigma \in \mathbb{G}^* \times \{0,1\}^*$, and returns either 0 or 1.*

A signature scheme must fulfill the following ***correctness*** *property:*[14]

$$\Pr\left[\textsc{Verify}^{\mathcal{O}_\mathsf{H}}(\mathrm{pk}, m, \sigma) = 1 \;\middle|\; \begin{array}{c}(\mathrm{sk}, \mathrm{pk}) \leftarrow \textsc{KeyGen}^{\mathcal{O}_\mathsf{H}}(1^\lambda) \\ m \leftarrow\!\!\$\ \mathcal{M} \\ \sigma \leftarrow \textsc{Sign}^{\mathcal{O}_\mathsf{H}}(\mathrm{sk}, m)\end{array}\right] \geq 1 - \mathsf{negl}(\lambda)\ ,$$

taken over $\mathcal{O}_\mathsf{H}$ and internal randomness of $\textsc{KeyGen}$, $\textsc{Sign}$, *and* $\textsc{Verify}$. *The signature scheme is* perfectly correct *if the probability is exactly* 1.

Definition 5 (Schnorr-like Signature Scheme). *We call a signature scheme $\Sigma = (\textsc{KeyGen}, \textsc{Sign}, \textsc{Verify})$* Schnorr-like, *if every public key is of the form $\mathrm{pk} = (\mathbf{g}, s) \in \mathbb{G}^{\ell_\mathsf{g}} \times \{0,1\}^{\ell_\mathsf{s}}$, every signature is of the form $\sigma = (\mathbf{h}, t) \in \mathbb{G}^{\ell_\mathsf{h}} \times \{0,1\}^{\ell_\mathsf{t}}$ (with $\ell_\mathsf{g}, \ell_\mathsf{h}, \ell_\mathsf{s}, \ell_\mathsf{t}$ all being polynomials in λ), and if* $\textsc{Verify}$ *can be written as on the left-hand side:*

[14] Note that this correctness definition is quite weak because m is sampled at random. This is sufficient for our lower bound, and only makes our impossibility result stronger. Both of our constructions, however, do actually fulfill perfect correctness.

$\textsc{Verify}^{\mathcal{O}_\mathsf{H}}(\mathrm{pk}, m, \sigma)$
parse $\mathrm{pk} = (\mathbf{g}, s)$ and $\sigma = (\mathbf{h}, t)$
$r \leftarrow \mathcal{O}_\mathsf{H}(C(s,m,t) \cdot \mathbf{h}, f(s,m,t))$
return $A(s,m,r,t) \cdot \mathbf{g} \stackrel{?}{=} \mathbf{h}$

$\textsc{Verify}^{\mathcal{O}_\mathsf{H}}(\mathrm{pk}, m, \sigma)$
parse $\mathrm{pk} = (\mathbf{g}, s)$ and $\sigma = (\mathbf{h}, t)$
$r \leftarrow \mathcal{O}_\mathsf{H}(C(s,m,t) \cdot \mathbf{h}, f(s,m,t))$
return $A(s,m,r,t) \cdot \mathbf{g} \stackrel{?}{=} B(s,m,r,t) \cdot \mathbf{h}$

If it can be written as on the right-hand side, then Σ is extended *Schnorr-like. The functions f, A, B, and C must be efficiently computable, with*

- $f : \{0,1\}^{\ell_\mathsf{s}} \times \mathcal{M} \times \{0,1\}^{\ell_\mathsf{t}} \to \{0,1\}^*$,
- $A : \{0,1\}^{\ell_\mathsf{s}} \times \mathcal{M} \times \mathcal{R}_\mathsf{H} \times \{0,1\}^{\ell_\mathsf{t}} \to \mathbb{Z}_p^{*\times \ell_\mathsf{g}}$,
- $B : \{0,1\}^{\ell_\mathsf{s}} \times \mathcal{M} \times \mathcal{R}_\mathsf{H} \times \{0,1\}^{\ell_\mathsf{t}} \to \mathbb{Z}_p^{*\times \ell_\mathsf{h}}$, *and*
- $C : \{0,1\}^{\ell_\mathsf{s}} \times \mathcal{M} \times \{0,1\}^{\ell_\mathsf{t}} \to \mathbb{Z}_p^{*\times \ell_\mathsf{h}}$.

We further require the following technical restriction to hold for any (extended) Schnorr-like signature scheme Σ: For every $\mathrm{pk} = (\mathbf{g}, s)$*, every two distinct messages $m_1, m_2 \in \mathcal{M}$, corresponding signatures $\sigma_1 = (\mathbf{h}_1, t_1), \sigma_2 = (\mathbf{h}_2, t_2)$ and RO outputs $r_1, r_2 \in \mathcal{R}_\mathsf{H}$ such that the signatures are valid (i.e., $A(s, m_b, r_b, t_b) \cdot \mathbf{g} = B(s, m_b, r_b, t_b) \cdot \mathbf{h}$ for $b \in \{1,2\}$):*[15]

$$(C(s,m_1,t_1) \cdot \mathbf{h}_1, f(s,m_1,t_1)) \neq (C(s,m_2,t_2) \cdot \mathbf{h}_2, f(s,m_2,t_2))$$

Note that *extended* Schnorr-like are a generalization of what is called "fully algebraic signatures" in [DHH+21] (where verification has the form $A \cdot \mathbf{g} \stackrel{?}{=} B \cdot \mathbf{h}$ without any RO), where they are proven inherently insecure.[16]

Definition 6 (EUF-CMA Security). *For a group-based signature scheme $\Sigma = (\textsc{KeyGen}, \textsc{Sign}, \textsc{Verify})$ in the ROM and an adversary $\mathcal{A}$, we define the advantage in the* existential unforgeability under chosen message attack *(EUF-CMA) game as* $\mathrm{Adv}^{\mathsf{euf\text{-}cma}}_{\Sigma,\mathcal{A}}(\lambda) := \Pr[\mathsf{euf\text{-}cma}_{\Sigma,\mathcal{A}}(\lambda) = 1]$*, where the game is defined as follows:*

$\mathsf{euf\text{-}cma}_{\Sigma,\mathcal{A}}(\lambda)$
$\mathcal{O}_\mathsf{H} \leftarrow$ *uniformly random function* $\mathbb{G}^* \times \{0,1\}^* \to \mathcal{R}_\mathsf{H}$
$(\mathrm{sk}, \mathrm{pk}) \leftarrow \textsc{KeyGen}^{\mathcal{O}_\mathsf{H}}(1^\lambda)$
$(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}_\mathsf{H}, \textsc{Sign}^{\mathcal{O}_\mathsf{H}}(\mathrm{sk}, \cdot)}(1^\lambda, \mathrm{pk})$
if $\textsc{Verify}(\mathrm{pk}, m^*, \sigma^*) = 1 \wedge m^* \notin \{m_1, \ldots, m_k\}$ **then return** 1
else return 0

Here, m_i denotes the i-th query made by $\mathcal{A}$ to its signing oracle. The probability is taken over $\mathcal{O}_\mathsf{H}$ and all algorithms' internal randomness.

We say that Σ is generically *EUF-CMA secure, if for every generic adversary A that makes at most* $\mathrm{poly}(\lambda)$ *queries to the group operation oracle $\mathcal{O}_\mathsf{grp}$ and to the RO $\mathcal{O}_\mathsf{H}$ (but otherwise is not necessarily efficient), we have* $\mathrm{Adv}^{\mathsf{euf\text{-}cma}}_{\Sigma,\mathsf{A}}(\lambda) \leq \mathsf{negl}(\lambda)$.

[15] Intuitively, this restriction says that the message m must be somehow "encoded" into the input of the RO query made during verification. It seems that any useful scheme would have this property, since otherwise the same RO query could be used for multiple signatures, in which case the RO does not appear to have a purpose. But while RO-free algebraic signatures are known to be insecure [DHH+21], it is also not clear how to turn this intuition into a concrete attack on *any* given signature scheme that does not conform to our technical restriction.

[16] For fully algebraic signatures, our technical restriction can be trivially fulfilled by a vacuous RO query to the signed message m.

Definition 7 (Black-Box Reductions for EUF-CMA Security). *Let $(\mathcal{P}, \epsilon)$ be a group assumption, and let Σ be a group-based signature scheme in the ROM. A* black-box reduction $\mathcal{R}(1^\lambda, 1^{q_{\text{SIGN}}})$ *is a ppt algorithm that takes as input 1^λ and $1^{q_{\text{SIGN}}}$, interacts with $\mathcal{P}$, and simultaneously may invoke an EUF-CMA adversary $\mathcal{A}(1^\lambda, \cdot)$ by providing the* euf-cma*-interface (i.e., it outputs* pk*, answers* SIGN*-queries, and finally takes a forgery (m^*, σ^*) from $\mathcal{A}$). The reduction is in the*

- ***programmable ROM****, if $\mathcal{R}$ additionally provides an interface to $\mathcal{A}$ for answering $\mathcal{O}_{\mathsf{H}}$-queries (and can choose the $\mathcal{O}_{\mathsf{H}}$-outputs itself);*
- ***non-programmable but observable ROM****, if both $\mathcal{R}$ and $\mathcal{A}$ have access to a uniformly random RO $\mathcal{O}_{\mathsf{H}}$, and $\mathcal{R}$ is informed about every query that $\mathcal{A}$ makes to $\mathcal{O}_{\mathsf{H}}$;*
- ***non-programmable, non-observable ROM****, if both $\mathcal{R}$ and $\mathcal{A}$ have access to a uniformly random RO $\mathcal{O}_{\mathsf{H}}$, and all queries to $\mathcal{O}_{\mathsf{H}}$ are hidden from the other party.*

See Figure 4 for an illustration. By abuse of notation, independently of which ROM is used, we write $\mathcal{R}^{\mathcal{A}}$ to denote the algorithm that combines the execution of $\mathcal{R}$ with that of $\mathcal{A}$ (and $\mathcal{O}_{\mathsf{H}}$ according to which ROM we are in), and therefore has no external interaction except that with assumption $\mathcal{P}$. Note that $\mathcal{R}$ is allowed to initiate more than one euf-cma*-session with $\mathcal{A}$. Each time, $\mathcal{A}$ is invoked on the same security parameter λ.*

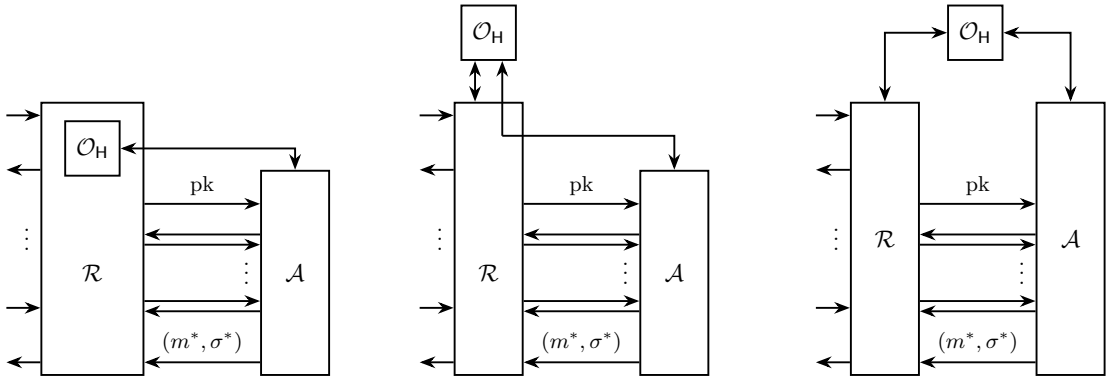

Fig. 4: The three different modes in which a security reduction may use the RO.

We say that the reduction $\mathcal{R}$ proves EUF-CMA security of Σ under the assumption $(\mathcal{P}, \epsilon)$*, if for every polynomial $q_{\text{SIGN}}(\lambda)$ and every adversary $\mathcal{A}$*

- *that is computationally unbounded,*
- *guaranteed to make $\leq q_{\text{SIGN}}$ queries to its signing oracle,*
- *makes at most* $\text{poly}(\lambda)$ *queries to the RO $\mathcal{O}_{\mathsf{H}}$, and*
- *has non-negligible advantage* $\text{Adv}^{\mathsf{euf\text{-}cma}}_{\Sigma,\mathcal{A}}(\lambda)$*,*

$\mathcal{R}^{\mathcal{A}}_{q_{\text{SIGN}}}$ breaks the assumption $(\mathcal{P}, \epsilon)$, i.e., $\text{Adv}^{\mathcal{P}}_{\mathcal{R}^{\mathcal{A}}_{q_{\text{SIGN}}}}(\lambda)$ is non-negligible[17]*. Here, $\mathcal{R}^{\mathcal{A}}_{q_{\text{SIGN}}}(1^\lambda)$ is defined as running $\mathcal{R}^{\mathcal{A}}(1^\lambda, 1^{q_{\text{SIGN}}(\lambda)})$.*

A generic *black-box reduction* R *has the same properties as above, except that* R *must be generic, is computationally unbounded, and makes $\leq \text{poly}(\lambda)$ queries to its group operation oracle $\mathcal{O}_{\mathsf{grp}}$. Furthermore, in all ROM modes (except the programmable ROM mode where* R *manages $\mathcal{O}_{\mathsf{H}}$ itself),* R *is allowed to make at most* $\text{poly}(\lambda)$ *queries to $\mathcal{O}_{\mathsf{H}}$.*

[17] Clearly, our definition of "the reduction proving EUF-CMA security" is sound: regardless of which ROM mode the reduction $\mathcal{R}$ is using, the assumption $(\mathcal{P}, \epsilon)$ being true implies that Σ is EUF-CMA secure. This is because every ppt adversary $\mathcal{A}$ would result in a ppt algorithm $\mathcal{R}^{\mathcal{A}}_{q_{\text{SIGN}}}$ breaking the assumption.

Definition 8 (*Tight* Generic Reductions). *Suppose* R *is a generic black-box reduction that proves EUF-CMA security of* Σ *under an assumption* $(\mathcal{P}, \epsilon)$*. We say that* R *is a* tight *reduction, if there exists some polynomial* $\mathrm{loss}(\lambda)$*, s.t. for every polynomial* $q_{\text{SIGN}}(\lambda)$ *and for every generic adversary* A *(with* $\leq q_{\text{SIGN}}(\lambda)$ *signature queries and* $\leq \mathrm{poly}(\lambda)$ $\mathcal{O}_{\mathsf{H}}$*-queries):*

$$\frac{\mathrm{Adv}^{\mathsf{euf\text{-}cma}}_{\Sigma,\mathsf{A}}(\lambda)}{\mathrm{Adv}^{\mathcal{P}}_{\mathsf{R}^{\mathsf{A}}_{q_{\text{SIGN}}}}(\lambda)} \cdot k(\lambda) \leq \mathrm{loss}(\lambda) + \mathsf{negl}(\lambda) \ ,$$

where $k(\lambda)$ *is an upper bound on the number of* $\mathsf{euf\text{-}cma}$*-sessions the reduction* $\mathsf{R}(1^\lambda, 1^{q_{\text{SIGN}}})$ *initiates with adversary* A*.*[18]

Note that we do not require a constant security loss $\mathrm{loss}(\lambda)$. Nevertheless, tightness implies that the security loss does not depend on q_{SIGN}.

3.3 Commit-and-Prove Systems

Definition 9 (Languages over Groups). *We call a set* $L \subseteq \mathbb{G}^{\ell_{\mathsf{u}}} \times \mathbb{G}^{\ell_{\mathsf{v}}}$ *a language over group* $\mathbb{G}$*, where*

$$L = \{(\mathbf{u}, \mathbf{v}) \in \mathbb{G}^{\ell_{\mathsf{u}}} \times \mathbb{G}^{\ell_{\mathsf{v}}} \mid W_L(\mathbf{u}, \mathbf{v}) \neq \emptyset\} \ ,$$

with $W_L(\mathbf{u}, \mathbf{v})$ *denoting the set of witnesses for element* $(\mathbf{u}, \mathbf{v})$*.*

A class of languages $\mathcal{L}$ *is a set of languages (which do not necessarily utilize the same lengths* ℓ_{u} *and* ℓ_{v}*).*

In the definition above, an element $(\mathbf{u}, \mathbf{v})$ of a language L consists of two parts, because in the following we consider commit-and-prove systems that are used for showing statements formulated with group elements $\mathbf{u}$ together with *commitments* of group elements $\mathbf{v}$.

We now define such commit-and-prove systems. Note that the definition is slightly non-standard in the sense that it supports proving/verifying *lists of statements*, which will be necessary for the signature scheme Σ_2, where several proofs (using different $\mathbf{crs}$) need to share a single RO query.

Definition 10 (Group-Based Commit-and-Prove System in ROM). *A Commit-and-Prove System* Π *consists of algorithms* BGen, HGen, Com, Prove, SimProve, Verify *for a class of languages* $\mathcal{L}$ *over* $\mathbb{G}$*, where the proof and verification algorithms all have access to an RO* $\mathcal{O}_{\mathsf{H}}$*:*

- $\text{BGen}(1^\lambda)$ *generates a common reference string* $\mathbf{crs}$ *in* binding *mode.*
- $\text{HGen}(1^\lambda)$ *generates a common reference string* $\mathbf{crs}$ *in* hiding *mode together with a trapdoor* $\mathbf{trap}$*.*
- $\text{Com}(\mathbf{crs}, \mathsf{v}; R)$ *takes a* $\mathbf{crs}$*, a group element* $\mathsf{v} \in \mathbb{G}$ *and randomness* $R \leftarrow\!\!\$\ \mathcal{R}_{\mathsf{com}}$ *(for some arbitrary set* $\mathcal{R}_{\mathsf{com}}$ *that depends on* λ*), and* deterministically *returns a commitment* $\mathrm{com} \in \mathbb{G}^* \times \{0,1\}^*$ *to* v*.*
- $\text{Prove}^{\mathcal{O}_{\mathsf{H}}}((\mathbf{crs}^{(j)}, L^{(j)}, \mathbf{u}^{(j)}, (\mathrm{com}_i^{(j)}, R_i^{(j)})_{i \in [\ell_{\mathsf{v}}^{(j)}]}, w^{(j)})_{j \in [N]})$ *proves/simulates a list of* N *statements, each of them consisting of a* $\mathbf{crs}^{(j)}$*, a language* $L \subseteq \mathbb{G}^{\ell_{\mathsf{u}}^{(j)}} \times \mathbb{G}^{\ell_{\mathsf{v}}^{(j)}}$*,* $L^{(j)} \in \mathcal{L}$*, a vector* $\mathbf{u}^{(j)} \in \mathbb{G}^{\ell_{\mathsf{u}}^{(j)}}$ *of group elements, and a list of commitments* $\mathrm{com}_i^{(j)}$ *that use randomness* $R_i^{(j)} \in \mathcal{R}_{\mathsf{com}}$*, and a witness* $w^{(j)}$ *(which can be either an actual witness, or a trapdoor* $\mathbf{trap}^{(j)}$*). The output is a single proof* $\pi \in \mathbb{G}^* \times \{0,1\}^*$*.*

[18] Note that this is a simplified form of what is typically called "tight" in the literature: we do not compare the running times of A and R^{A} with each other, but simply "estimate" their ratio from the number of times $k(\lambda)$ that A is invoked. Whenever the running time of A is fixed for every λ (i.e., it always is equal to its worst-case running time), then these two definitions are equivalent.

- $\textsc{Verify}^{\mathcal{O}_\mathsf{H}}((\mathbf{crs}^{(j)}, L^{(j)}, \mathbf{u}^{(j)}, (\mathrm{com}_i^{(j)})_{i\in[\ell_\mathsf{v}{}^{(j)}]})_{j\in[N]}, \pi)$ *verifies a proof for a list of* N *statements. Each of them comes with a* $\mathbf{crs}^{(j)}$*, (the description of) a language* $L \subseteq \mathbb{G}^{\ell_\mathsf{u}^{(j)}} \times \mathbb{G}^{\ell_\mathsf{v}^{(j)}}$*,* $L \in \mathcal{L}$*, a vector* $\mathbf{u} \in \mathbb{G}^{\ell_\mathsf{u}^{(j)}}$ *of group elements, as well as a list of commitments* $\mathrm{com}_i^{(j)}$*. All of these statements are supplemented by a single proof* $\pi \in \mathbb{G}^* \times \{0,1\}^*$*, and the output of* Verify *is a single bit that indicates acceptance.*

Definition 11 (Properties of a Commit-and-Prove System). *Every commit-and-prove system must fulfill the following* ***correctness*** *property: For every RO* $\mathcal{O}_\mathsf{H}$*, and for every list of* N *statements, each consisting of a* $\mathbf{crs}^{(j)}$ *(generated with either* $\textsc{BGen}^{\mathcal{O}_\mathsf{H}}(1^\lambda)$ *or* $(\mathbf{crs}^{(j)}, \mathbf{trap}^{(j)}) \leftarrow \textsc{HGen}^{\mathcal{O}_\mathsf{H}}(1^\lambda)$*; note that two* $\mathbf{crs}^{(j)}$ *may have been generated using different methods), a language* $L^{(j)} \in \mathcal{L}$*, a vector* $\mathbf{u}^{(j)} \in \mathbb{G}^{\ell_\mathsf{u}^{(j)}}$ *of group elements, group elements* $\mathsf{v}_i^{(j)} \in \mathbb{G}$ *with randomnesses* $R_i^{(j)} \in \mathcal{R}_\mathsf{com}$ *and corresponding commitments* $\mathrm{com}_i^{(j)} \leftarrow \textsc{Com}^{\mathcal{O}_\mathsf{H}}(\mathbf{crs}^{(j)}, \mathsf{v}_i^{(j)}; R_i^{(j)})$*, and witnesses* $w^{(j)} \in W_{L^{(j)}}(\mathbf{u}^{(j)}, \mathbf{v}^{(j)})$ *(or* $w^{(j)} \in W_{L^{(j)}}(\mathbf{u}^{(j)}, \mathbf{v}^{(j)}) \cup \{\mathbf{trap}^{(j)}\}$ *if the* $\mathbf{crs}^{(j)}$ *was generated in hiding mode), we have* $\textsc{Verify}^{\mathcal{O}_\mathsf{H}}((\mathbf{crs}^{(j)}, L^{(j)}, \mathbf{u}^{(j)}, (\mathrm{com}_i^{(j)}))_{j\in[N]}, \pi) = 1$ *given the proof* π *generated as* $\pi \leftarrow \textsc{Prove}^{\mathcal{O}_\mathsf{H}}((\mathbf{crs}^{(j)}, L^{(j)}, \mathbf{u}^{(j)}, (\mathrm{com}_i^{(j)}, R_i^{(j)}), w^{(j)})_{j\in[N]})$*.*

It may further satisfy the following security properties:

Computational CRS Indistinguishability. *It is hard to determine whether hiding or binding mode is used. Formally, for every ppt adversary* $\mathcal{A}$*, the following must be negligible in* λ*:*

$$\left|\Pr\left[\mathcal{A}(\mathbf{crs}) = 1 \mid \mathbf{crs} \leftarrow \textsc{BGen}(1^\lambda)\right] - \Pr\left[\mathcal{A}(\mathbf{crs}) = 1 \mid (\mathbf{crs}, \mathbf{trap}) \leftarrow \textsc{HGen}(1^\lambda)\right]\right|$$

Dual-Mode Commitment. *Using* BGen *results in perfectly binding commitments. Formally, for any* $\mathbf{crs} \leftarrow \textsc{BGen}(1^\lambda)$*, there must not exist any elements* $\mathsf{v}_1, \mathsf{v}_2 \in \mathbb{G}$ *and randomness* $R_1, R_2 \in \mathcal{R}_\mathsf{com}$ *with* $\mathsf{v}_1 \neq \mathsf{v}_2$ *and* $\textsc{Com}(\mathbf{crs}, \mathsf{v}_1; R_1) = \textsc{Com}(\mathbf{crs}, \mathsf{v}_2; R_2)$*.*
Using HGen *results in perfectly hiding commitments. Formally, for any fixed* $(\mathbf{crs}, \mathbf{trap}) \leftarrow \textsc{HGen}(1^\lambda)$ *and every two elements* $\mathsf{v}_1, \mathsf{v}_2 \in \mathbb{G}$*, the two distributions* $\mathcal{D}_1$ *and* $\mathcal{D}_2$ *must be identical, where* $\mathcal{D}_i := \{\mathrm{com} \leftarrow \textsc{Com}(\mathbf{crs}, \mathsf{v}_i; R) \mid R \leftarrow\!\!\$\, \mathcal{R}_\mathsf{com}\}$*.*

Statistical Soundness. *In binding mode it is hard to prove an invalid statement. Formally, for every adversary* $\mathcal{A}$ *that makes at most* $\mathrm{poly}(\lambda)$ *calls to the random oracle* $\mathcal{O}_\mathsf{H}$ *(but otherwise is allowed to be inefficient in all other metrics) there exists a negligible function* $\mathsf{negl}(\lambda)$*, s.t. for any* $N \in \mathbb{N}$ *and list of* $(\mathbf{crs}^{(j)})_{j\in[N]}$*, and for any index* $j^* \in [N]$ *for which* $\mathbf{crs}^{(j^*)}$ *is an output of* $\textsc{BGen}(1^\lambda)$ *(i.e., the* j^**th crs is in binding mode), it is (almost) impossible to cheat by making the* j^**th statement invalid:*

$$\Pr\left[\begin{array}{l} L^{(1)}, \ldots, L^{(N)} \in \mathcal{L} \wedge \\ \textsc{Verify}^{\mathcal{O}_\mathsf{H}}((\mathbf{crs}^{(j)}, L^{(j)}, \mathbf{u}^{(j)}, (\mathrm{com}_i^{(j)}))_{j\in[N]}, \pi) = 1 \wedge \\ \nexists \mathsf{v}_i^{(j^*)}, R_i^{(j^*)} \text{ with } \mathrm{com}_i^{(j^*)} = \textsc{Com}(\mathbf{crs}^{(j^*)}, \mathsf{v}_i^{(j^*)}; R_i^{(j^*)}) \text{ and } (\mathbf{u}^{(j^*)}, \mathbf{v}^{(j^*)}) \in L^{(j^*)} \\ \qquad \Big| \ (L^{(j)}, \mathbf{u}^{(j)}, (\mathrm{com}_i^{(j)}))_{j\in[N]}, \pi \leftarrow \mathcal{A}^{\mathcal{O}_\mathsf{H}}((\mathbf{crs}^{(j)})_{j\in[N]}) \end{array}\right] < \mathsf{negl}(\lambda)\ ,$$

where the probability is taken over the choice of the RO $\mathcal{O}_\mathsf{H}$ *and* $\mathcal{A}$*'s internal randomness.*

Perfect Zero-Knowledge. *Consider any fixed random oracle* $\mathcal{O}_\mathsf{H}$*, and a list of* $\mathbf{crs}^{(j)}$*, each generated using either* BGen *or* HGen*. Further consider any list of statements, each consisting of a language* $L^{(j)} \in \mathcal{L}$*, group elements* $\mathbf{u}^{(j)} \in \mathbb{G}^{\ell_\mathsf{u}^{(j)}}$*, group elements elements* $\mathsf{v}_i^{(j)} \in \mathbb{G}$*, randomnesses* $R_i^{(j)} \in \mathcal{R}_\mathsf{com}$ *(for* $i \in [\ell_\mathsf{v}^{(j)}]$*) and corresponding commitments* $\mathrm{com}_i^{(j)} \leftarrow \textsc{Com}(\mathbf{crs}, \mathsf{v}_i^{(j)}; R_i^{(j)})$*.*
Then, the distribution of π *generated as follows is identical for any choice of witness* $w^{(j)} \in W_{L^{(j)}}(\mathbf{u}^{(j)}, \mathbf{v}^{(j)})$ *(if* $\mathbf{crs}^{(j)}$ *was generated using* $\textsc{BGen}(1^\lambda)$*) or* $w^{(j)} \in W_{L^{(j)}}(\mathbf{u}^{(j)}, \mathbf{v}^{(j)}) \cup \{\mathbf{trap}^{(j)}\}$ *(if* $\mathbf{crs}^{(j)}$ *was generated using* $(\mathbf{crs}^{(j)}, \mathbf{trap}^{(j)}) \leftarrow \textsc{HGen}(1^\lambda)$*):*

$$\pi \leftarrow \textsc{Prove}^{\mathcal{O}_\mathsf{H}}((\mathbf{crs}^{(j)}, L^{(j)}, \mathbf{u}^{(j)}, (\mathrm{com}_i^{(j)}, R_i^{(j)})_i, w^{(j)})_{j\in[N]})\ .$$

Definition 12 (Linear Languages with Conjunctions and Disjunctions over Groups). *The class of linear languages with conjuctions and disjunctions (LCDs) $\mathcal{L}$ contains exactly those languages that can be written in one of the following three forms:*

- $L_{(M_i,M_i')_{i\in[n]}}$, *with matrices* $M_1,\ldots,M_n \in \mathbb{Z}_p^{(1+\ell_\mathsf{w})\times\ell_\mathsf{u}}$ *and* $M_1',\ldots,M_n' \in \mathbb{Z}_p^{(1+\ell_\mathsf{w})\times\ell_\mathsf{v}}$. *This language is defined as*

$$L_{(M_i,M_i')_{i\in[n]}} := \left\{ (\mathbf{u},\mathbf{v}) \in \mathbb{G}^{\ell_\mathsf{u}} \times \mathbb{G}^{\ell_\mathsf{v}} \;\middle|\; \begin{array}{l} \exists \mathbf{w} \in \mathbb{Z}_p^{\ell_\mathsf{w}} \text{ s.t. for all } i \in [n]: \\ \left(1 \| \mathbf{w}^T\right) M_i \mathbf{u} + \left(1 \| \mathbf{w}^T\right) M_i' \mathbf{v} = 0 \end{array} \right\},$$

  The set $W_{L_{(M_i,M_i')_{i\in[n]}}}(\mathbf{u},\mathbf{v})$ *consists of all* $\mathbf{w}$ *that fulfill the condition above.*
- $L_1 \wedge L_2$ *for two languages* $L_1, L_2 \in \mathcal{L}$, *defined as*

$$L_1 \wedge L_2 := \{((\mathbf{u}_1\|\mathbf{u}_2),(\mathbf{v}_1\|\mathbf{v}_2)) \in \mathbb{G}^{\ell_{\mathsf{u}1}+\ell_{\mathsf{u}2}} \times \mathbb{G}^{\ell_{\mathsf{v}1}+\ell_{\mathsf{v}2}} \mid (\mathbf{u}_1,\mathbf{v}_1) \in L_1 \wedge (\mathbf{u}_2,\mathbf{v}_2) \in L_2\}$$

- $L_1 \vee L_2$ *for two languages* $L_1, L_2 \in \mathcal{L}$, *defined as*

$$L_1 \vee L_2 := \{((\mathbf{u}_1\|\mathbf{u}_2),(\mathbf{v}_1\|\mathbf{v}_2)) \in \mathbb{G}^{\ell_{\mathsf{u}1}+\ell_{\mathsf{u}2}} \times \mathbb{G}^{\ell_{\mathsf{v}1}+\ell_{\mathsf{v}2}} \mid (\mathbf{u}_1,\mathbf{v}_1) \in L_1 \vee (\mathbf{u}_2,\mathbf{v}_2) \in L_2\}$$

For the latter two, the witness sets are defined as follows:

$$W_{L_1\wedge L_2}((\mathbf{u}_1\|\mathbf{u}_2),(\mathbf{v}_1\|\mathbf{v}_2)) := \{(w_1,w_2) \mid w_1 \in W_{L_1}(\mathbf{u}_1,\mathbf{v}_1) \wedge w_2 \in W_{L_2}(\mathbf{u}_2,\mathbf{v}_2)\}$$
$$W_{L_1\vee L_2}((\mathbf{u}_1\|\mathbf{u}_2),(\mathbf{v}_1\|\mathbf{v}_2)) := \{(1,w) \mid w \in W_{L_1}(\mathbf{u}_1,\mathbf{v}_1)\} \cup \{(2,w) \mid w \in W_{L_2}(\mathbf{u}_2,\mathbf{v}_2)\}$$

We use $\mathrm{desc} : \mathcal{L} \to \{0,1\}^*$ *to denote some injective function that canonically describes each language* $L \in \mathcal{L}$ *using a short bitstring (i.e., polynomial in* $\log p$, n, ℓ_u, ℓ_v, ℓ_w, *and the number of operators* $\vee$ *and* $\wedge$*).*

In Appendix A, we provide Construction 3, an instantiation of commit-and-prove systems for LCDs based on Groth-Sahai commitments [GS08] from DDH, transformed into the (non-programmable and non-observable) ROM to replace the use of pairings. In Lemma 11 we prove that this instantiation fulfills all properties from Definition 11.

4 Constructing Signature Schemes without Observability

In this section, we present our constructive results.

4.1 A non-tight Schnorr-like signature scheme

The following signature scheme is *Schnorr-like* in the sense of Definition 5, but does not have a tight security reduction.

Construction 1 *We construct the following signature scheme* Σ_1 *with message space* $\mathcal{M} := \{0,1\}^\lambda$ *from a commit-and-prove system* Π *for LCDs. Throughout, given a message* $m \in \mathcal{M}$, *we use* m_i *to denote its* i*-th bit.*

$\textsc{KeyGen}^{\mathcal{O}_\mathsf{H}}(1^\lambda)$. *Generate* $(\mathbf{crs},\mathbf{trap}) \leftarrow \Pi.\textsc{HGen}(1^\lambda)$ *and commitments*

$$\mathrm{com}_i \leftarrow \Pi.\textsc{Com}(\mathbf{crs}, a_i\mathsf{g}_0; R) \quad \text{with} \quad R \leftarrow\!\!\$\, \mathcal{R}_\mathsf{com}\,,$$

for all $i = 0,\ldots,i_\lambda$, *with the scalars being chosen as* $a_0 := 1$ *and* $a_i := 0$ *for* $i \in [\lambda]$. *The keypair is defined as:*

$$\mathrm{pk} := (\mathbf{crs}, \mathrm{com}_0, \ldots, \mathrm{com}_\lambda)$$
$$\mathrm{sk} := (R_0, \ldots, R_\lambda, a_0, \ldots, a_\lambda)$$

$\textsc{Sign}^{\mathcal{O}_{\mathsf{H}}}(\mathrm{sk}, m)$. *We sign by proving that the scalars a_i that are committed to in com_i fulfill $a_0 + \sum_{i\in\lambda} m_i a_i \neq 0$. That is, given language*

$$L_m := \left\{ (\mathsf{g}_0, (\mathsf{a}_0, \ldots, \mathsf{a}_\lambda)) \mid \exists w \text{ s.t. } w \left(\mathsf{a}_0 + \sum_{\substack{i\in\lambda \\ m_i=1}} \mathsf{a}_i \right) = \mathsf{g}_0 \right\} ,$$

the signature is computed as

$$\sigma := \Pi.\textsc{Prove}^{\mathcal{O}_{\mathsf{H}}}(\mathbf{crs}, L_m, (\mathsf{g}_0), (\mathrm{com}_i, a_i\mathsf{g}_0, R_i)_{i\in\{0,\ldots,\lambda\}}, w) ,$$

using witness $w := (a_0 + \sum_{i\in[\lambda]} m_i a_i)^{-1}$ (over $\mathbb{Z}_p$).

$\textsc{Verify}^{\mathcal{O}_{\mathsf{H}}}(\mathrm{pk}, m, \sigma)$. *We verify the proof σ, i.e., we return*

$$\Pi.\textsc{Verify}^{\mathcal{O}_{\mathsf{H}}}(\mathbf{crs}, L_m, (\mathsf{g}_0), (\mathrm{com}_i)_{i\in\{0,\ldots,\lambda\}}, \sigma) ,$$

for the same language L_m as in Sign.

Theorem 1. *Construction 1 (when instantiated with our commit-and-prove system given in Construction 3) is a (perfectly correct) Schnorr-like signature scheme, for which there exists a (non-tight) generic black-box reduction in the non-programmable and non-observable ROM that proves its EUF-CMA security under the* $(\mathsf{DDH}, \frac{1}{2})$*-assumption.*

Proof. Construction 1 clearly fulfills **perfect correctness**, which follows from correctness of the underlying prove-and-commit system due to w always being a valid witness.

Construction 1 is also **Schnorr-like**, which can be seen from our instantiation of a commit-and-prove system (see Construction 3), and the fact that *all group elements and commitments that the proof system's verification function is called on are publicly computed as a linear function from the group elements contained in the public key* pk. The **technical restriction** (at the end of Definition 5) is fulfilled, because the bitstring $\mathrm{desc}(L_m)$ is always part of the RO-call that Verify makes. Since L_m uniquely determines the message m, the message is implicitly encoded in the RO-input.

We now prove **EUF-CMA security** of Construction 1 using a hybrid argument. This hybrid argument can be turned (using common techniques) into a single generic reduction R to the $(\mathsf{DDH}, \frac{1}{2})$-assumption.

In the following, we use a_m to denote $a_0 + \sum_{\substack{i\in[\lambda] \\ m_i=1}} a_i$ for a message $m \in \{0,1\}^\lambda$.

We point out that in [HK08, Theorem 3.4] (which in turn, is an explicit statement of something that was already proven in [Wat05]), it has been shown that for any polynomial $q_{\textsc{Sign}}(\lambda)$, there exist distributions $\mathcal{D}_i$ (for $i \in \{0,\ldots,\lambda\}$), s.t. for all $m^{(1)}, \ldots, m^{(q_{\textsc{Sign}}(\lambda))}, m^*$ (with $m^{(j)} \neq m^*$ for all j), the following holds:

$$\Pr\left[\begin{array}{l|l} \forall j \in [q_{\textsc{Sign}}(\lambda)] : a_{m^{(j)}} \neq 0 \wedge & \forall i \in \{0,\ldots,\lambda\} : \\ a_{m^*} = 0 & a_i \leftarrow \mathcal{D}_i \end{array} \right] \geq \frac{1}{8(\lambda+1)q_{\textsc{Sign}}(\lambda)}$$

We first recall $\mathsf{euf\text{-}cma}_{\Sigma_1,\mathcal{A}}(\lambda)$ (in the following denoted by Hyb_0):

1. Initialize $a_0 := 1$ and $a_1, \ldots, a_\lambda := 0$.
2. Run $(\mathbf{crs}, \mathbf{trap}) \leftarrow \Pi.\textsc{HGen}(1^\lambda)$ and $\mathrm{com}_i \leftarrow \Pi.\textsc{Com}(\mathbf{crs}, a_i\mathsf{g}_0; R)$ with $R \leftarrow\!\!\$\ \mathcal{R}_{\mathsf{com}}$.
3. Define the keypair $(\mathrm{pk}, \mathrm{sk})$ as $\mathrm{pk} := (\mathbf{crs}, \mathrm{com}_0, \ldots, \mathrm{com}_\lambda)$ and $\mathrm{sk} := (R_0, \ldots, R_\lambda, a_0, \ldots, a_\lambda)$. pk is sent to $\mathcal{A}$.
4. For each signature query $m^{(j)}$ (with $j \in [q_{\textsc{Sign}}]$), return the signature

$$\sigma := \Pi.\textsc{Prove}^{\mathcal{O}_{\mathsf{H}}}(\mathbf{crs}, L_{m^{(j)}}, (\mathsf{g}_0), (\mathrm{com}_i, R_i)_{i\in\{0,\ldots,\lambda\}}, w)$$

to $\mathcal{A}$, computed using witness $w := (a_0 + \sum_{m_i^{(j)}=1} a_i)^{-1}$.

5. $\mathcal{A}$ outputs (m^*, σ^*).
6. We return 1 iff $\Pi.\textsc{Verify}^{\mathcal{O}_{\mathsf{H}}}(\mathbf{crs}, L_{m^*}, (\mathsf{g}_0), (\mathrm{com}_i)_{i\in\{0,\dots,\lambda\}}, \sigma) = 1$ and $m^* \neq m_j$ (for all $j \in [q_{\textsc{Sign}}]$).

Hyb_1. In addition to Hyb_0, we add the following step 0:
- Sample $a'_i \leftarrow \mathcal{D}_i$ (for $i \in \{0,\dots,\lambda\}$).

Furthermore, we add the following between steps 5 and 6:
- If $a'_{m^{(j)}} = 0$ for some m_j or if $a'_{m^*} \neq 0$, immediately return 0.

Given the result above, we have

$$\Pr[\mathsf{Hyb}_1 \text{ outputs } 1] \geq \frac{1}{8(\lambda+1)q_{\textsc{Sign}}(\lambda)} \cdot \Pr[\mathsf{Hyb}_0 \text{ outputs } 1] \,.$$

Hyb_2. In this hybrid, we abort (with output 0) immediately whenever the adversary submits a message $m^{(j)}$ with $a'_{m^{(j)}} = 0$, or if $a'_{m^*} \neq 0$ holds for the forgery message m^*.
Clearly, we have $\mathsf{Hyb}_1 \equiv \mathsf{Hyb}_2$.

Hyb_3. In contrast to Hyb_2, step 1 now selects $a_i := a'_i$. Due to the *perfect dual-mode commitment* and *perfect zero-knowledge* properties, we have $\mathsf{Hyb}_2 \equiv \mathsf{Hyb}_3$.

Hyb_4. In contrast to Hyb_3, we now generate **crs** using $\mathbf{crs} \leftarrow \Pi.\textsc{BGen}(1^\lambda)$. (We can do so, because we do not use the trapdoor **trap** anywhere and all used witnesses are valid.)
By the *CRS Indistinguishability* property (for our concrete DDH-based commit-and-prove scheme), we have

$$\Pr[\mathsf{Hyb}_4 \text{ outputs } 1] \geq \Pr[\mathsf{Hyb}_3 \text{ outputs } 1] - 2\mathrm{Adv}^{\mathsf{DDH}}_{\mathcal{A}}(\lambda) - 2/p \,.$$

Note that in Hyb_4, we have

$$\Pr[\mathsf{Hyb}_4 \text{ outputs } 1] < \mathsf{negl}(\lambda) \,,$$

which follows from the *statistical soundness* property, because of $a_{m^*} = 0$ and therefore $(\mathsf{g}_0, (\mathsf{a}_0, \dots, \mathsf{a}_\lambda)) \notin L_{m^*}$.

In conclusion, we get

$$\mathrm{Adv}^{\mathsf{euf\text{-}cma}}_{\Sigma_1,\mathcal{A}} = \Pr[\mathsf{Hyb}_0] \leq 16(\lambda+1)q_{\textsc{Sign}}(\lambda) \cdot \mathrm{Adv}^{\mathsf{DDH}}_{\mathcal{A}}(\lambda) + \mathsf{negl}(\lambda).$$

Note that this constitutes a *non-tight* reduction to DDH. □

4.2 A tight *extended* Schnorr-like signature scheme

Now we present the construction of a tightly secure signature scheme, which is however only of the *extended* Schnorr-like form (see Definition 5). The construction is quite involved and is essentially a variant of the tightly secure signature scheme from [AHN+23]. For this reason we merely state the construction using the formalism of our ROM-based commit-and-prove system, and we refer the reader to [AHN+23] for further intuition.

For the construction, we will need a public-key encryption scheme PKE, in which the group $\mathbb{G}$ forms the message space, and ciphertexts also consist purely of group elements. Using ElGamal encryption [ElG84], we get a PKE that has the following form:

$\mathrm{PKE.KeyGen}(1^\lambda)$**:** Returns a secret key $\mathrm{sk} \in \mathbb{Z}_p$ and a public key $\mathrm{pk} \in \mathbb{G}$.
$\mathrm{PKE.Enc}(\mathrm{pk}, \mathsf{m})$**:** Given public key $\mathrm{pk} \in \mathbb{G}$ and a message $\mathsf{m} \in \mathbb{G}$, returns a ciphertext $\mathsf{ct} \in \mathbb{G}^2$.
$\mathrm{PKE.Dec}(\mathrm{sk}, \mathsf{ct})$**:** Given a secret key $\mathrm{sk} \in \mathbb{Z}_p$ and a ciphertext $\mathsf{ct} \in \mathbb{G}^2$, returns a message $\mathsf{m} \in \mathbb{G}$. Decryption can be written be written as a linear function of the elements in ct and coefficients that depend on sk.

IND-CPA security (based on $(\mathsf{DDH}, \frac{1}{2})$) is as defined by [AHN+23].

We can implement the languages required by the construction stated in Figure 1 from [AHN+23] in our framework as follows, where $\mathsf{x}_0, \mathsf{x}_1, \mathsf{sk}_0, \mathsf{sk}_1, \mathsf{sk}_2$ are the group elements that $c_0, c_1, c_2, k_0, k_1, k_2$ in that paper's construction commit to.

$$
\begin{aligned}
L_0 &:= \left\{ ((\mathsf{g}_0, \mathsf{ct}_0, \mathsf{m}), (\mathsf{x}_0, \mathsf{x}_1, \mathsf{sk}_0)) \;\middle|\; \begin{array}{r} \exists x_1, \mathrm{sk}_0 \text{ s.t. } \mathsf{x}_1 = x_1 \mathsf{g}_0 \wedge \mathsf{sk}_0 = \mathrm{sk}_0 \mathsf{g}_0 \wedge \\ \text{PKE.DEC}(\mathrm{sk}_0, \mathsf{ct}_0) = \mathsf{x}_0 + x_1 \mathsf{m} \end{array} \right\} \\
L_1' &:= \left\{ ((\mathsf{ct}_0, \mathsf{ct}_1), (\mathsf{sk}_0, \mathsf{sk}_1)) \;\middle|\; \begin{array}{r} \exists \mathrm{sk}_0, \mathrm{sk}_1 \text{ s.t. } \mathsf{sk}_0 = \mathrm{sk}_0 \mathsf{g}_0 \wedge \mathsf{sk}_1 = \mathrm{sk}_1 \mathsf{g}_0 \wedge \\ \text{PKE.DEC}(\mathrm{sk}_0, \mathsf{ct}_0) = \text{PKE.DEC}(\mathrm{sk}_1, \mathsf{ct}_1) \end{array} \right\} \\
L_1'' &:= \{((\mathsf{ct}_2), (\mathsf{x}_2, \mathsf{sk}_2)) \mid \exists \mathrm{sk}_2 \text{ s.t. } \mathsf{sk}_2 = \mathrm{sk}_2 \mathsf{g}_0 \wedge \text{PKE.DEC}(\mathrm{sk}_2, \mathsf{ct}_2) = \mathsf{x}_2\} \\
L_1 &:= L_1' \vee L_1''
\end{aligned}
$$

We can now re-state the signature scheme from [AHN+23] as follows.

Construction 2 *We construct the following signature scheme Σ_2 from commit-and-prove systems for the languages L_0 and L_1 above:*

$\text{KEYGEN}^{\mathcal{O}_\mathsf{H}}(1^\lambda)$. *Setup the proof system and ElGamal keypairs:*

$$
\begin{aligned}
&\mathbf{crs}_0, \mathbf{crs}_1 \leftarrow \Pi.\text{BGEN}(1^\lambda) \\
&(\mathrm{pk}_0, \mathrm{sk}_0), (\mathrm{pk}_1, \mathrm{sk}_1), (\mathrm{pk}_2, \mathrm{sk}_2) \leftarrow\$ \text{PKE.KEYGEN}(1^\lambda)
\end{aligned}
$$

Define values $x_0 \leftarrow\$ \mathbb{Z}_p$ and $x_1 := x_2 := 0 \in \mathbb{Z}_p$, and commit to them using randomness $R_0, R_1, R_2 \leftarrow\$ \mathcal{R}_\mathsf{com}$:

$$
\begin{aligned}
&\mathrm{com}_{x,0} \leftarrow \Pi.\text{COM}(\mathbf{crs}_0, x_0; R_0), \mathrm{com}_{x,1} \leftarrow \Pi.\text{COM}(\mathbf{crs}_0, x_1; R_1), \\
&\mathrm{com}_{x,2} \leftarrow \Pi.\text{COM}(\mathbf{crs}_1, x_2; R_2)
\end{aligned}
$$

Commit to secret keys using randomness $T_0, T_1, T_2, T_3 \leftarrow\$ \mathcal{R}_\mathsf{com}$:

$$
\begin{aligned}
&\mathrm{com}_{\mathrm{sk},0} \leftarrow \Pi.\text{COM}(\mathbf{crs}_1, \mathrm{sk}_0; T_0), \mathrm{com}_{\mathrm{sk},1} \leftarrow \Pi.\text{COM}(\mathbf{crs}_1, \mathrm{sk}_1; T_1), \\
&\mathrm{com}_{\mathrm{sk},2} \leftarrow \Pi.\text{COM}(\mathbf{crs}_1, \mathrm{sk}_2; T_2), \mathrm{com}_{\mathrm{sk},3} \leftarrow \Pi.\text{COM}(\mathbf{crs}_0, \mathrm{sk}_0; T_3)
\end{aligned}
$$

Output the following keys:

$$
\begin{aligned}
\mathrm{pk} &:= (\mathbf{crs}_0, \mathbf{crs}_1, (\mathrm{com}_{x,i})_{0 \le i \le 2}, (\mathrm{com}_{sk,i})_{0 \le i \le 3}, (\mathrm{pk}_i)_{0 \le i \le 2}) \\
\mathrm{sk} &:= ((\mathrm{sk}_i, x_i, R_i)_{0 \le i \le 2}, (T_i)_{0 \le i \le 3})
\end{aligned}
$$

$\text{SIGN}^{\mathcal{O}_\mathsf{H}}(\mathrm{sk}, m)$. *In contrast to the signature scheme from [AHN+23], our messages are scalars $m \in \mathbb{Z}_p$, but this is only a syntactic change.* SIGN *takes $m \in \mathbb{Z}_p$ as input and actually proceeds with $\mathsf{m} = m\mathsf{g}_0$ for the generator g_0 of $\mathbb{G}$. Then, encrypt values $z_0 := z_1 := x_0$ and $z_2 := 0$ as follows:*

$$
\mathsf{ct}_0 \leftarrow \text{PKE.ENC}(\mathrm{pk}_0, z_0 \mathsf{g}_0); \mathsf{ct}_1 \leftarrow \text{PKE.ENC}(\mathrm{pk}_1, z_1 \mathsf{g}_0); \mathsf{ct}_2 \leftarrow \text{PKE.ENC}(\mathrm{pk}_2, z_2 \mathsf{g}_0) \; .
$$

We will prove two things:

- *Under $\mathbf{crs}_0$, we have*

$$
(\underbrace{(\mathsf{g}_0, \mathsf{ct}_0, \mathsf{m})}_{=:\mathbf{u}}, \underbrace{(x_0, x_1, \mathrm{sk}_0)\mathsf{g}_0}_{=:\mathbf{v}}) \in L_0
$$

using witness $w := (x_1, \mathrm{sk}_0)$.
The commitments to $\mathbf{v}$ are given as $\mathrm{com} := ((\mathrm{com}_{x,0}, R_0), (\mathrm{com}_{x,1}, R_1), (\mathrm{com}_{\mathrm{sk},3}, T_3))$.

- *Under* $\mathbf{crs}_1$*, we have*

$$\underbrace{((\mathsf{ct}_0, \mathsf{ct}_1)}_{=:\mathbf{u}'}, \underbrace{(\mathrm{sk}_0, \mathrm{sk}_1)\mathsf{g}_0)}_{=:\mathbf{v}'} \in L_1' \quad \text{or} \quad (\underbrace{\mathsf{ct}_2}_{=:\mathbf{u}''}, \underbrace{(x_2, \mathrm{sk}_2)\mathsf{g}_0}_{=:\mathbf{v}''}) \in L_1''$$

  Note that both statements are true, so we can for example select witness $w' := (1, (\mathrm{sk}_0, \mathrm{sk}_1))$*. The commitments to* $\mathbf{v}'$ *resp.* $\mathbf{v}''$ *are given as* $\mathrm{com}' := ((\mathrm{com}_{\mathrm{sk},0}, T_0), (\mathrm{com}_{\mathrm{sk},1}, T_1))$ *and* $\mathrm{com}'' := ((\mathrm{com}_{x,2}, R_2), (\mathrm{com}_{\mathrm{sk},2}, T_2))$*.*

We output the signature $\sigma := (\mathsf{ct}_0, \mathsf{ct}_1, \mathsf{ct}_2, \pi)$*, using the proof*

$$\pi := \Pi.\textsc{Prove}^{\mathcal{O}_\mathsf{H}} \left(\begin{array}{l} (\mathbf{crs}_0, L_0, \mathbf{u}, \mathrm{com}, w)\,, \\ (\mathbf{crs}_1, L_1' \vee L_1'', (\mathbf{u}' \| \mathbf{u}''), \mathrm{com}' \| \mathrm{com}'', w') \end{array} \right) .$$

$\textsc{Verify}^{\mathcal{O}_\mathsf{H}}(\mathrm{pk}, \mathsf{m}, \sigma)$**.** *Parse* $\sigma = (\mathsf{ct}_0, \mathsf{ct}_1, \mathsf{ct}_2, \pi)$*. Then output the following, which denotes whether the proof* π *is valid:*

$$\Pi.\textsc{Verify}^{\mathcal{O}_\mathsf{H}} \left(\left(\begin{array}{l} (\mathbf{crs}_0, L_0, \mathbf{u}, (\mathrm{com}_{x,0}, \mathrm{com}_{x,1}, \mathrm{com}_{\mathrm{sk},3}))\,, \\ (\mathbf{crs}_1, L_1' \vee L_1'', (\mathbf{u}' \| \mathbf{u}''), (\mathrm{com}_{\mathrm{sk},0}, \mathrm{com}_{\mathrm{sk},1}, \mathrm{com}_{x,2}, \mathrm{com}_{\mathrm{sk},2})) \end{array} \right), \pi \right) ,$$

where statements $\mathbf{u}, \mathbf{u}', \mathbf{u}''$ *are as defined in* $\textsc{Sign}$*.*

As stated, this signature scheme is actually the tightly secure signature scheme from [AHN+23], cast in our formalism (and with our ROM-based commit-and-prove system). Hence, the security proof of [AHN+23] applies almost verbatim. Specifically, we claim:

Theorem 2. *For every adversary* $\mathcal{A}$ *on the EUF-CMA security on Construction 2 (implemented with our RO-based prove-and-commit system from Construction 3), there exists an adversary* $\mathcal{B}$ *on the DDH problem, such that*

$$\mathrm{Adv}^{\mathsf{euf\text{-}cma}}_{\Sigma_2, \mathcal{A}} \;\leq\; \mathsf{O}(\log q_{\textsc{Sign}}) \cdot \mathrm{Adv}^{\mathsf{DDH}}_{\mathcal{B}} + \frac{\mathrm{poly}(q_{\textsc{Sign}} + q_\mathsf{H})}{p},$$

where $q_{\textsc{Sign}}$ *and* q_H *is an upper bound on the number of signing and random oracle queries that* $\mathcal{A}$ *makes, and* p *denotes the group order.*

We do not give a full proof of Theorem 2, but only detail the (minor) differences to the setting and proof of [AHN+23, Theorem 3.6]:

- The scheme in [AHN+23] supports arbitrary group elements m as messages (to be structure-preserving), while our messages m are exponents. That is, we implicitly use the scheme from [AHN+23] with $\mathsf{m} = m\mathsf{g}_0$. This does not affect the security proof (since all messages to be signed are chosen by the adversary), but in fact enables the proof to know all discrete logarithms of messages to be signed.
- The proof of [AHN+23, Theorem 3.6] only proves UF-XCMA security, which is the same as our EUF-CMA security, except that a UF-XCMA adversary is required to give a discrete logarithm m of a (group element) message m along with a signing query for m. By our preceding remark, any signing query already contains (and in fact consists only of) m.
- Our definition of prove-and-commit schemes only guarantees statistical soundness (in the sense defined in Definition 11 and provided by our RO-based Construction 3) when its CRS has been generated as binding. In contrast, Groth-Sahai proofs as used in [AHN+23] enjoy perfect soundness under binding CRSs, and the proof of [AHN+23, Theorem 3.6] explicitly refers to this perfect soundness property (e.g., in the intermediate proof step captured by [AHN+23, Lemma 3.11]). Since our soundness guarantees are still statistical, they can be formalized as "perfect, unless an event $\mathsf{BAD}_{\mathrm{sound}}$ occurs", with $\mathsf{BAD}_{\mathrm{sound}}$ having probability at most q_H/p. Hence, the original proof applies, but with an additional $\mathrm{poly}(q_\mathsf{H}/p)$ term (reflected in Theorem 2).
- The remaining proof of [AHN+23, Theorem 3.6] applies verbatim (adapted to our notation).

We stress that Construction 2 fulfills the technical restriction (at the end of Definition 5), because the encoded message $\mathsf{m} = m\mathsf{g}_0$ is always injected into the $\mathcal{O}_\mathsf{H}$ as part of the statement $\mathbf{u}$.

5 Lower Bound

The following theorem is our main result:

Theorem 3. *Suppose that Σ is a Schnorr-like signature scheme with super-polynomial message space. Then, there cannot exist a* tight, generic reduction in non-programmable, non-observable ROM mode*, which proves EUF-CMA security of Σ under any (potentially even interactive)* generically true assumption $(\mathcal{P}, \epsilon)$.

We start with a simplification, and then move on to proving Theorem 3 in Section 5.2.

5.1 Simplifying the RO call

As a first step, we assume that Σ's verification algorithm has a slightly simplified form $\mathcal{O}_\mathsf{H}(\mathbf{h}, f(s,m,t))$ instead of $\mathcal{O}_\mathsf{H}(C(s,m,t)\cdot\mathbf{h}, f(s,m,t))$ (i.e., the matrix $C(s,m,t)$ is always the identity). In the following lemma, we prove that this simplification is w.l.o.g.

Lemma 1. *Suppose that Σ is Schnorr-like signature scheme. Then, we can transform Σ into another Schnorr-like signature scheme Σ', where* VERIFY *has the following more restricted form:*

$\textsc{Verify}^{\mathcal{O}_\mathsf{H}}(\mathrm{pk}, m, \sigma)$

parse $\mathrm{pk} = (\mathbf{g}, s)$ and $\sigma = (\mathbf{h}, t)$

$r \leftarrow \mathcal{O}_\mathsf{H}(\mathbf{h}, f(s,m,t))$

return $A(s,m,r,t)\cdot\mathbf{g} \stackrel{?}{=} \mathbf{h}$

Furthermore, for any tight, generic reduction in the non-programmable, non-observable ROM that proves EUF-CMA security of Σ under some generically true assumption $(\mathcal{P}, \epsilon)$, there also exists such a reduction that proves EUF-CMA security of Σ'.

Proof. The main idea is that we write $\mathbf{h}' := C(s,m,t)\cdot\mathbf{h}$ directly into the signature instead of $\mathbf{h}$. Because of this change, a verification matrix in the resulting scheme will need to have the form $C(s,m,t)\cdot A(s,m,r,t)$ instead of $A(s,m,r,t)$.

Formally, given $\Sigma = (\textsc{KeyGen}, \textsc{Sign}, \textsc{Verify})$, we construct $\Sigma' = (\textsc{KeyGen}, \textsc{Sign}', \textsc{Verify}')$ as follows:

- $\textsc{Sign}'(\mathrm{sk}, m)$ runs $(\mathbf{h}, t) \leftarrow \textsc{Sign}(\mathrm{sk}, m)$ and outputs $(C(s,m,t)\cdot\mathbf{h}, t)$.
- $\textsc{Verify}'(\mathrm{pk}, m, \sigma)$ for $\sigma = (\mathbf{h}', t)$ computes $r \leftarrow \mathcal{O}_\mathsf{H}(\mathbf{h}', f(s,m,t))$ and checks

$$C(s,m,t)\cdot A(s,m,r,t)\cdot\mathbf{g} \stackrel{?}{=} \mathbf{h}' \ .$$

  (Note that this verification equation still satisfies the Schnorr-like form. Furthermore, the technical restriction imposed by Definition 5 on Σ' can be easily derived from that of Σ.)

It may seem like this verification check could be easier to fulfill than verification in the original scheme Σ (e.g. due to low rank of matrix $C(s,m,t)$). However, given a forgery $\sigma' = (\mathbf{h}', t)$ for message m we can still use $\mathbf{h} := A(s,m,r,t)\cdot\mathbf{g}$ to get a forgery of the original scheme (i.e., we rely on the restricted verification form to avoid having to reverse-compute $\mathbf{h}$ from $\mathbf{h}'$).

More precisely, we can construct a reduction R' from EUF-CMA security of Σ' to EUF-CMA security of Σ as follows: in the beginning, R' receives a public key pk, and passes it through to its Σ'-adversary A'. Whenever A' queries a message m, R' forwards m to the EUF-CMA game of Σ, which returns a signature σ. R' transforms this into a signature σ' as follows, and returns σ' to A':

$$\sigma = (\mathbf{h}, t) \quad \rightsquigarrow \quad \sigma' := (C(s, m, t) \cdot \mathbf{h}, t)$$

Once A' produces a forgery $\sigma' = (\mathbf{h}', t)$ of m under scheme Σ', R' outputs a forgery σ for m under scheme Σ as follows:

$$\sigma' := (\mathbf{h}', t) \quad \rightsquigarrow \quad \sigma := (A(s, m, r, t) \cdot \mathbf{g}, t), \text{ where } r := \mathcal{O}_{\mathsf{H}}(\mathbf{h}', f(s, m, t))$$

Note that R' is a valid reduction that proves EUF-CMA security of Σ' under EUF-CMA security of Σ: it perfectly emulates the EUF-CMA game for Σ', and in addition, whenever the final forgery σ' is valid for m, then it will be successfully transformed into a valid forgery σ in scheme Σ.

Now let R be some generic reduction proving EUF-CMA security of Σ under the assumption $(\mathcal{P}, \epsilon)$, and let us define the generic reduction R^* proving EUF-CMA security of Σ' under the assumption $(\mathcal{P}, \epsilon)$ as the joint execution of R with R'. Then, by definition, for any Σ'-adversary A' against Σ', we have $\mathrm{Adv}^{\mathcal{P}}_{(\mathsf{R}^*)^{\mathsf{A}'}} = \mathrm{Adv}^{\mathcal{P}}_{\mathsf{R}^{\mathsf{A}}}$, where A is the Σ-adversary constructed as the joint execution of R' with A'. As shown before, we also have $\mathrm{Adv}^{\mathsf{euf\text{-}cma}}_{\Sigma',\mathsf{A}'}(\lambda) = \mathrm{Adv}^{\mathsf{euf\text{-}cma}}_{\Sigma,\mathsf{A}}(\lambda)$, and therefore R^*'s security loss is bounded by that of R. □

5.2 Attack outline

We now prove Theorem 3 under the assumption that Σ has the simplified verification form described in Lemma 1. For concreteness, we will further assume that the RO output space is $\mathcal{R}_{\mathsf{H}} := \{0,1\}^{\ell_{\mathsf{r}}}$ for some ℓ_{r}. The proof can easily be adapted to arbitrary $\mathcal{R}_{\mathsf{H}}$ (with polynomial-length output).

In Figure 5, we present two adversaries $\mathsf{A}_{\mathsf{unbounded}}$ and $\mathsf{A}_{\mathsf{prog}}$. The first step for both of them is to run a FILTER(pk) procedure, which produces a set

$$\mathcal{T} \subseteq \mathcal{M} \times \{0,1\}^{\ell_{\mathsf{r}}} \times \{0,1\}^{\ell_{\mathsf{t}}}$$

of triples for which our two adversaries will be able to sample signatures in a statistically close way. Given $\mathcal{T}$, we define the following shorthand notation:

$$\mathcal{T}_m := \{(r,t) \mid (m,r,t) \in \mathcal{T}\} \quad \text{and} \quad \mathcal{T}_{m,r} := \{t \mid (r,t) \in \mathcal{T}_m\} \,.$$

While sampling is easy in $\mathsf{A}_{\mathsf{prog}}$, we will have to construct a separate procedure SAMPLE(pk, $\mathbf{x}, \mathcal{T}$) for $\mathsf{A}_{\mathsf{unbounded}}$, where $\mathbf{x}$ is the dlog of $\mathbf{g}$.

The desired filtering outcome. Fix some message m and $\mathcal{O}_{\mathsf{H}}$-output r. For any potential dlog $\mathbf{x} \in \mathbb{Z}_p^{\ell_{\mathsf{g}}}$ of $\mathbf{g}$, for any $\mathbf{y} \in \mathbb{Z}_p^{\ell_{\mathsf{h}}}$, and for any bitstring $f \in \{0,1\}^*$, we let $S^{\mathcal{T}}_{m,r}(\mathbf{x}, \mathbf{y}, f)$ be the set of remaining tags t that would result in bitstring $f = f(s, m, t)$ while the matrix $A(s, m, r, t)$, seen as a function of the potential dlog $\mathbf{x}$, would simultaneously cross the point $(\mathbf{x}, \mathbf{y})$:

$$S^{\mathcal{T}}_{m,r}(\mathbf{x}, \mathbf{y}, f) := \{t \in \mathcal{T}_{m,r} \mid A(s, m, r, t) \cdot \mathbf{x} = \mathbf{y} \wedge f = f(s, m, t)\}$$

For convenience, we define $i^{\mathcal{T}}_{m,r}(\mathbf{x}, \mathbf{y}, f)$ as the unique $i \in [\ell_{\mathsf{t}}] \cup \{\bot\}$ with

$$2^{i-1} \leq |S^{\mathcal{T}}_{m,r}(\mathbf{x}, \mathbf{y}, f)| < 2^i \,,$$

or as $i = \bot$ if $|S^{\mathcal{T}}_{m,r}(\mathbf{x}, \mathbf{y}, f)| = 0$. Intuitively, $i^{\mathcal{T}}_{m,r}(\mathbf{x}, \mathbf{y}, f)$ describes a rough interval that covers the number of tags t resulting in bitstring f and for which the matrix $A(s, m, r, t)$ crosses the point $(\mathbf{x}, \mathbf{y})$.

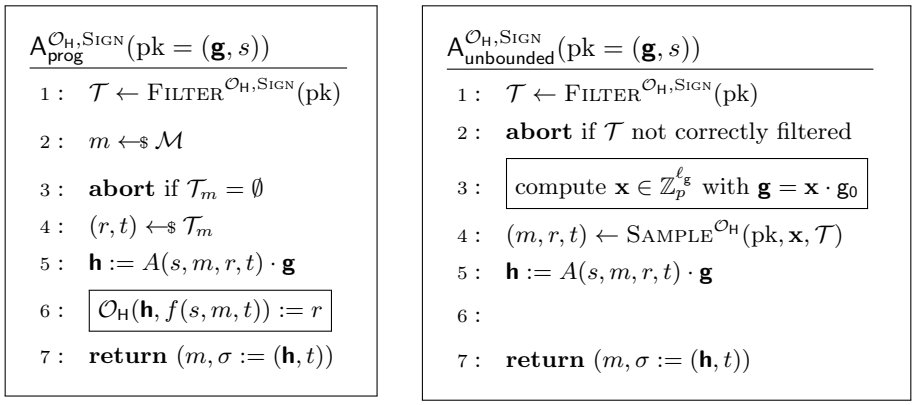

Fig. 5: Our two adversaries, with their respective "superpower" emphasized. Whenever either FILTER or SAMPLE aborts (or returns $\perp$), the adversary exits unsuccessfully without forgery. $\mathsf{A}_{\mathsf{prog}}$ (who controls the RO $\mathcal{O}_\mathsf{H}$ entirely) responds to all $\mathcal{O}_\mathsf{H}$-queries consistently and uniformly at random (unless that entry has been programmed before). Recall that g_0 is a generator available to any generic procedure.

Given a fixed message m and some $i \in [\ell_\mathsf{t}]$, as well as a point $(\mathbf{x},\mathbf{y})$ and a bitstring $f \in \{0,1\}^*$, we define

$$R^{\mathcal{T}}_{m,i}(\mathbf{x},\mathbf{y},f) := \{r \in \{0,1\}^{\ell_\mathsf{r}} \mid i^{\mathcal{T}}_{m,r}(\mathbf{x},\mathbf{y},f) = i\}\ .$$

Intuitively, in the terms used in the technical overview, a triple (m,r,t) would be considered "suitable" for $\mathbf{x}$, if $R^{\mathcal{T}}_{m,i}(\mathbf{x},\mathbf{y},f)$ (on $f = f(s,m,t)$, $\mathbf{y} = A(s,m,r,t)\cdot\mathbf{x}$, and $i = i^{\mathcal{T}}_{m,r,}(\mathbf{x},\mathbf{y})$) contains a polynomially-sized fraction of all RO-outputs. Put differently, in order for $\mathsf{A}_{\mathsf{unbounded}}$ to be able to sample signatures in a distribution that roughly matches $\mathcal{T}$, we require that for all remaining triples $(m,r,t) \in \mathcal{T}$, there are many RO outputs r' for which the number of bitstrings t' that result in the same RO-input made during VERIFY is similar to that of r. This intuition about the property we want from the filtering process is formalized in the following definition.

Definition 13. *Given polynomials* $\mathrm{poly}_{\mathsf{bound}}(\lambda)$ *and* $\mathrm{poly}_{\mathsf{filter}}(\lambda)$*, we say that a set* $\mathcal{T} \subseteq \mathcal{M}\times\{0,1\}^{\ell_\mathsf{r}}\times\{0,1\}^{\ell_\mathsf{t}}$ ***is correctly filtered*** *w.r.t. public key* $\mathrm{pk} = (\mathbf{g},s)$*, if for the dlog* $\mathbf{x}$ *of* $\mathbf{g}$ *(i.e.,* $\mathbf{g} = \mathbf{x}\cdot\mathsf{g}_0$*), the following holds:*

$$\Pr\left[\begin{array}{l|l} \mathcal{T}_m \neq \emptyset \ \wedge & m \leftarrow\!\$\ \mathcal{M} \\ |R^{\mathcal{T}}_{m,i}(\mathbf{x},\mathbf{y},f)| \geq \frac{2^{\ell_\mathsf{r}}}{\mathrm{poly}_{\mathsf{bound}}(\lambda)} & (r,t) \leftarrow\!\$\ \mathcal{T}_m \end{array}\right] \geq 1 - \frac{1}{\mathrm{poly}_{\mathsf{filter}}(\lambda)}\ , \tag{6}$$

where the values f*,* $\mathbf{y}$*, and* i *used inside the event are defined as* $f := f(s,m,t)$*,* $\mathbf{y} := A(s,m,r,t)\cdot\mathbf{x}$*, and* $i := i^{\mathcal{T}}_{m,r}(\mathbf{x},\mathbf{y},f)$*.*

In Section 5.3, we define the procedure FILTER and state the properties that it satisfies. In Section 5.4, we show how to implement SAMPLE, and then combine everything in Section 5.5. Lemmas stated along the way are proven from Section 5.6 onwards.

5.3 Implementing the filter process

See Figure 6 for a full description of $\textsc{Filter}^{\mathcal{O}_\mathsf{H},\textsc{Sign}}$ and some intuition.

There are two main properties that we will need from FILTER. First, when interacting with an *honest* signer, then with overwhelming probability, FILTER outputs some set $\mathcal{T}$ (instead of aborting). The following lemma captures this:

$\textsc{Filter}^{\mathcal{O}_{\mathsf{H}},\textsc{Sign}}(\mathrm{pk} = (\mathbf{g}, s))$

$\mathcal{T} := \mathcal{M} \times \{0,1\}^{\ell_{\mathsf{r}}} \times \{0,1\}^{\ell_{\mathsf{t}}}$
repeat $2\mathrm{poly}_{\mathsf{filter}}(\lambda) \cdot (\ell_{\mathsf{t}} + \ell_{\mathsf{r}})$ **times**
  $\mathcal{T}' \leftarrow \textsc{FilterStep}^{\mathcal{O}_{\mathsf{H}},\textsc{Sign}}(\mathrm{pk}, \mathcal{T})$
  if $\mathcal{T}' = \mathcal{T}$ **then return** $\mathcal{T}$
  else $\mathcal{T} := \mathcal{T}'$
end repeat
return $\bot$

$\textsc{FilterStep}^{\mathcal{O}_{\mathsf{H}},\textsc{Sign}}(\mathrm{pk} = (\mathbf{g}, s), \mathcal{T})$

Phase 1: Filter potential dlog's of g

$\mathcal{G} := \mathbb{Z}_p^{\ell_{\mathsf{g}}}$
repeat $4(\mathrm{poly}_{\mathsf{filter}}(\lambda))^2 \cdot (\ell_{\mathsf{t}} + \ell_{\mathsf{r}}) \cdot (\lambda + \ell_{\mathsf{g}} \cdot \lceil \log p \rceil)$ **times**
  $m \leftarrow\!\!\$\ \mathcal{M}$ and $(\mathbf{h}, t) := \sigma \leftarrow \textsc{Sign}(m)$
  $f := f(s, m, t)$; $r \leftarrow \mathcal{O}_{\mathsf{H}}(\mathbf{h}, f)$; $A := A(s, m, r, t)$
  abort if $A \cdot \mathbf{g} \neq \mathbf{h}$ // i.e., if the signature is not valid
  $\mathcal{G} := \mathcal{G} \setminus \left\{ \mathbf{x} \in \mathcal{G} \;\middle|\; \begin{array}{l} |R^{\mathcal{T}}_{m,i}(\mathbf{x}, A\mathbf{x}, f)| < \frac{2^{\ell_{\mathsf{r}}}}{\mathrm{poly}_{\mathsf{bound}}(\lambda)} \\ \quad \text{with } i := i^{\mathcal{T}}_{m,r}(\mathbf{x}, A\mathbf{x}, f) \neq \bot \end{array} \right\}$
end repeat

Phase 2: Filter triples (m, r, t) **in** $\mathcal{T}$

if there is *at least one* $\mathbf{x} \in \mathcal{G}$ for which Equation 6 is not fulfilled **then**
  pick any such $\mathbf{x}$
  return $\mathcal{T} \setminus \left\{ (m, r, t) \;\middle|\; \begin{array}{l} |R^{\mathcal{T}}_{m,i}(\mathbf{x}, \mathbf{y}, f)| < \frac{2^{\ell_{\mathsf{r}}}}{\mathrm{poly}_{\mathsf{bound}}(\lambda)}\ , \\ \text{with } \mathbf{y} := A(s, m, r, t) \cdot \mathbf{x}; f := f(s, m, t); i := i^{\mathcal{T}}_{m,r}(\mathbf{x}, \mathbf{y}, f) \end{array} \right\}$
else return $\mathcal{T}$

Fig. 6: The Filter procedure, which is parameterized by two polynomials $\mathrm{poly}_{\mathsf{filter}}(\lambda)$ and $\mathrm{poly}_{\mathsf{bound}}(\lambda)$. Note that it makes at most $8(\mathrm{poly}_{\mathsf{filter}}(\lambda))^3 \cdot (\ell_{\mathsf{t}} + \ell_{\mathsf{r}}) \cdot (\lambda + \ell_{\mathsf{g}} \cdot \lceil \log p \rceil)$ queries to the signing oracle Sign, and the same number of queries to $\mathcal{O}_{\mathsf{H}}$. Whenever FilterStep aborts, Filter returns $\bot$. Each filtering iteration consists of two parts: (1) finding a set $\mathcal{G}$ of dlog's of $\mathbf{g}$ that are *plausible* (because a signer with a bounded number of RO-calls would be able to produce the received triples), and (2) deleting all triples that are bad for one of the remaining dlog-candidates.

Lemma 2. *Taken over the choice of an RO $\mathcal{O}_{\mathsf{H}}$, a keypair* $(\mathrm{sk}, \mathrm{pk}) \leftarrow \textsc{KeyGen}(1^\lambda)$*, and an execution of* $\mathcal{T} \leftarrow \textsc{Filter}^{\mathcal{O}_{\mathsf{H}}, \textsc{Sign}^{\mathcal{O}_{\mathsf{H}}}(\mathrm{sk},\cdot)}(\mathrm{pk})$*, we have* $\Pr[\mathcal{T} = \bot] \leq \mathsf{negl}(\lambda)$.

We prove this lemma in Section 5.7.

The second property is that, regardless of whether Filter is interacting with an honest signer or not, the probability of returning some $\mathcal{T}$ that is *incorrectly filtered* (not including the case where $\mathcal{T} = \bot$), is bounded by roughly $\frac{(q_{\mathsf{H}}+q_{\textsc{Sign}})\cdot \ell_{\mathsf{t}}}{\mathrm{poly}_{\mathsf{bound}}(\lambda)}$, where q_{H} is an upper bound on the number of $\mathcal{O}_{\mathsf{H}}$-queries made by the signer. This is captured by the following lemma.

Lemma 3. *Let $\mathcal{R}$ be some* stateful *procedure, which outputs a public key* pk *and then responds to signature queries from* Filter*, and has access to an RO $\mathcal{O}_{\mathsf{H}}$. It is computationally unbounded, and may query $\mathcal{O}_{\mathsf{H}}$ (but not program it) up to a total of q_{H} times. Then, the following is a bound on the correctness of the filtering step:*

$$\Pr_{\mathcal{O}_{\mathsf{H}}}\left[\begin{array}{l|l} \mathcal{T} = \bot \vee & \mathrm{pk} \leftarrow \mathcal{R}^{\mathcal{O}_{\mathsf{H}}}(1^\lambda) \\ \mathcal{T} \textit{ is correctly filtered} & \mathcal{T} \leftarrow \textsc{Filter}^{\mathcal{O}_{\mathsf{H}}, \mathcal{R}^{\mathcal{O}_{\mathsf{H}}}}(\mathrm{pk}) \end{array}\right] \geq 1 - (q_{\mathsf{H}} + q_{\textsc{Sign}}) \cdot \frac{\ell_{\mathsf{t}}}{\mathrm{poly}_{\mathsf{bound}}(\lambda)} - \mathsf{negl}(\lambda)\ .$$

We prove this in Section 5.8. Intuitively, it follows by analyzing the probability that $\mathcal{R}$ (with bounded number of RO queries) is capable of producing a triple that is unsuitable for the *real* dlog of $\mathbf{g}$.

Combining both properties, note that when interacting with an honest signer, there is non-negligible probability of finding a *correctly* filtered $\mathcal{T}$ (see Lemma 7).

5.4 The Sample Procedure

Figure 7 shows procedure $\textsc{Sample}^{\mathcal{O}_{\mathsf{H}}}(\mathbf{x}, \mathcal{T})$ for any fixed polynomial $\mathrm{poly}_{\mathsf{bound}}(\lambda)$.

Intuitively, the goal of $\textsc{SampleStep}^{\mathcal{O}_{\mathsf{H}}}(\mathbf{x}, \mathcal{T})$ is to output each triple (m, r, t) according to the following distribution:

$$\alpha^{\mathcal{T}}_{m,r,t} := \begin{cases} \frac{1}{|\mathcal{M}|\cdot|\mathcal{T}_m|\cdot \mathrm{poly}_{\mathsf{bound}}(\lambda)\cdot 2} & \text{if } |R^{\mathcal{T}}_{m,i}(\mathbf{x}, \mathbf{y}, f)| \geq \frac{2^{\ell_{\mathsf{r}}}}{\mathrm{poly}_{\mathsf{bound}}(\lambda)}\ , \\ 0 & \text{otherwise}\ , \end{cases}$$

with $\mathbf{y} := A(s, m, r, t) \cdot \mathbf{x}$, $f := f(s, m, t)$, $i := i^{\mathcal{T}}_{m,r}(\mathbf{x}, \mathbf{y}, f)$. For convenience, let

$$\beta^{\mathcal{T}} := 1 - \sum_{(m,r,t)\in\mathcal{T}} \alpha^{\mathcal{T}}_{m,r,t}$$

denote the remaining probability (with which the procedure is expected to output $\bot$). First of all, note that for a correctly filtered set $\mathcal{T}$, this "abort-probability" $\beta^{\mathcal{T}}$ is bounded by $1 - \frac{1}{\mathrm{poly}_{\mathsf{bound}}(\lambda)\cdot 4}$:

Lemma 4. *For all polynomials* $\mathrm{poly}_{\mathsf{bound}}(\lambda)$ *and* $\mathrm{poly}_{\mathsf{filter}}(\lambda) \geq 2$*, and for any $\mathcal{T}$ that is correctly filtered w.r.t.* $\mathrm{pk} = (\mathbf{g}, s)$*, we have*

$$\beta^{\mathcal{T}} \leq 1 - \frac{1}{\mathrm{poly}_{\mathsf{bound}}(\lambda) \cdot 4}\ .$$

$\textsc{Sample}^{\mathcal{O}_{\mathsf{H}}}(\mathrm{pk}, \mathbf{x}, \mathcal{T})$

repeat $\mathrm{poly}_{\mathsf{bound}}(\lambda) \cdot 4 \cdot \lambda$ **times**
  if result $\leftarrow \textsc{SampleStep}^{\mathcal{O}_{\mathsf{H}}}(\mathrm{pk}, \mathbf{x}, \mathcal{T})$ did not abort **then return** result
end repeat
abort

$\textsc{SampleStep}^{\mathcal{O}_{\mathsf{H}}}(\mathrm{pk} = (\mathbf{g}, s), \mathbf{x}, \mathcal{T})$

$m \leftarrow\!\!\$\ \mathcal{M}$
abort if $\mathcal{T}_m = \emptyset$
$(r^*, t^*) \leftarrow\!\!\$\ \mathcal{T}_m$
$f := f(s, m, t^*);\ \mathbf{y} := A(s, m, r^*, t^*) \cdot \mathbf{x};\ i := i^{\mathcal{T}}_{m,r^*}(\mathbf{x}, \mathbf{y}, f)$
abort if $|R^{\mathcal{T}}_{m,i}(\mathbf{x}, \mathbf{y}, f)| < \dfrac{2^{\ell_{\mathsf{r}}}}{\mathrm{poly}_{\mathsf{bound}}(\lambda)}$
abort with prob. $1 - \dfrac{2^{\ell_{\mathsf{r}}}}{\mathrm{poly}_{\mathsf{bound}}(\lambda) \cdot |R^{\mathcal{T}}_{m,i}(\mathbf{x}, \mathbf{y}, f)|} \cdot \dfrac{2^{i-1} \cdot |R^{\mathcal{T}}_{m,i}(\mathbf{x}, \mathbf{y}, f)|}{\sum_{r \in R^{\mathcal{T}}_{m,i}(\mathbf{x},\mathbf{y},f)} |S^{\mathcal{T}}_{m,r}(\mathbf{x}, \mathbf{y}, f)|}$
$r \leftarrow \mathcal{O}_{\mathsf{H}}(\mathbf{y} \cdot \mathbf{g}_0, f)$
abort if $r \notin R^{\mathcal{T}}_{m,i}(\mathbf{x}, \mathbf{y}, f)$
abort with prob. $1 - \dfrac{|S^{\mathcal{T}}_{m,r}(\mathbf{x}, \mathbf{y}, f)|}{2^i}$
$t \leftarrow\!\!\$\ S^{\mathcal{T}}_{m,r}(\mathbf{x}, \mathbf{y}, f)$
return (m, r, t)

Fig. 7: The procedure Sample used within $\mathsf{A}_{\mathsf{unbounded}}$.

Proof. We call a triple (m, r, t) "good", if $|R^{\mathcal{T}}_{m,i}(\mathbf{x}, \mathbf{y}, f)| > \frac{2^{\ell_{\mathsf{r}}}}{\mathrm{poly}_{\mathsf{bound}}(\lambda)}$ (for corresponding $\mathbf{y}$, f, and i as above). Then:

$$\begin{aligned}
\beta^{\mathcal{T}} &= 1 - \sum_{(m,r,t) \in \mathcal{T}} \alpha^{\mathcal{T}}_{m,r,t} = 1 - \sum_{\text{good } (m,r,t) \in \mathcal{T}} \frac{1}{|\mathcal{M}| \cdot |\mathcal{T}_m| \cdot \mathrm{poly}_{\mathsf{bound}}(\lambda) \cdot 2} \\
&= 1 - \frac{1}{\mathrm{poly}_{\mathsf{bound}}(\lambda) \cdot 2} \cdot \sum_{\text{good } (m,r,t) \in \mathcal{T}} \Pr_{\substack{\tilde{m} \leftarrow\!\!\$ \mathcal{M} \\ (\tilde{r}, \tilde{t}) \leftarrow\!\!\$ \mathcal{T}_{\tilde{m}}}} [(m, r, t) = (\tilde{m}, \tilde{r}, \tilde{t})] \\
&\leq 1 - \frac{1}{\mathrm{poly}_{\mathsf{bound}}(\lambda) \cdot 2} \cdot \underbrace{\left(1 - \frac{1}{\mathrm{poly}_{\mathsf{filter}}(\lambda)}\right)}_{\geq \frac{1}{2}} \leq 1 - \frac{1}{\mathrm{poly}_{\mathsf{bound}}(\lambda) \cdot 4} \,.
\end{aligned}$$

Note that the second-to-last inequality follows by definition of the filtering property. □

Unfortunately, $\textsc{SampleStep}^{\mathcal{O}_{\mathsf{H}}}$ cannot perfectly achieve the distribution $\alpha^{\mathcal{T}}_{m,r,t}$, because the RO $\mathcal{O}_{\mathsf{H}}$ may not be entirely fresh for our adversaries when interacting with the reduction. Still, we obtain the following lemma, proven in Section 5.9. On a high level, the proof analyzes the output distribution of SampleStep assuming a *fresh* RO. The latter is justified through the technical property (see end of Definition 5), from which we may conclude that likely no RO query relevant to a fresh message $m \leftarrow\!\!\$\ \mathcal{M}$ has been made before (see Section 5.6).

Lemma 5. *Let* $\mathrm{poly}_{\mathsf{bound}}(\lambda)$ *and* $\mathrm{poly}_{\mathsf{filter}}(\lambda)$ *be arbitrary polynomials that parameterize* Sample. *Let* $\mathcal{R}$ *be a* stateful *procedure that first chooses* pk *and a set* $\mathcal{T}$*, then receives a triple* (m, r, t) *(or* $\perp$*) and*

finally outputs a bit b. $\mathcal{R}$ may both query and program the RO $\mathcal{O}_{\mathsf{H}}$*, up to a total of $q_{\mathsf{H}} \leq \mathrm{poly}(\lambda)$ times. Then, the distribution*

$$\left\{ \begin{array}{l} b \ \wedge \ \mathcal{T} \neq \bot \ \wedge \\ \mathcal{T} \textit{ fulfills filter cond.} \end{array} \middle| \begin{array}{l} (\mathrm{pk} = (\mathbf{g}, s), \mathcal{T}) \leftarrow \mathcal{R}^{\mathcal{O}_{\mathsf{H}}}() \\ \textit{choose } \mathbf{x} \in \mathbb{Z}_p^{\ell_{\mathsf{g}}} \textit{ s.t. } \mathbf{g} = \mathbf{x} \cdot \mathsf{g}_0 \\ (m, r, t) \leftarrow \textsc{Sample}^{\mathcal{O}_{\mathsf{H}}}(\mathrm{pk}, \mathbf{x}, \mathcal{T}) \\ b \leftarrow \mathcal{R}^{\mathcal{O}_{\mathsf{H}}}(m, r, t) \end{array} \right\}$$

differs statistically by at most $\mathsf{negl}(\lambda)$ *from the same distribution where* Sample *is replaced by* IdealSample*, defined as follows:*

$\textsc{IdealSample}^{\mathcal{O}_{\mathsf{H}}}(\mathrm{pk}, \mathbf{x}, \mathcal{T})$

sample (m, r, t), each with probability $\alpha^{\mathcal{T}}_{m,r,t}/(1 - \beta^{\mathcal{T}})$
$\mathcal{O}_{\mathsf{H}}(A(s, m, r, t) \cdot \mathbf{g}, f(s, m, t)) := r$
return (m, r, t)

5.5 Putting things together

Using Lemma 5, we are able to show that the two adversaries are close:

Lemma 6. *Let $\mathcal{R}$ be a (computationally unbounded) procedure that may engage in $\leq k$* euf-cma*-sessions with another procedure $\mathcal{A}$, i.e., for each session, it sends a* pk *to $\mathcal{A}$, then needs to answer $q_{\textsc{Sign}} \leq \mathrm{poly}(\lambda)$ signing queries, and finally accepts a forgery from $\mathcal{A}$. Upon exiting, $\mathcal{R}$ outputs a single bit. Further suppose that $\mathcal{R}$ and $\mathcal{A}$ have access to an RO $\mathcal{O}_{\mathsf{H}}$ (and $\mathcal{A}$ may even* program *it), which $\mathcal{R}$ queries at most $q_{\mathsf{H}} \leq \mathrm{poly}(\lambda)$ times. We write the execution of $\mathcal{R}$ with access to $\mathcal{A}$ as $\mathcal{R}^{\mathcal{A}}$.*

Let $\mathrm{poly}_{\mathsf{bound}}(\lambda)$ *and* $\mathrm{poly}_{\mathsf{filter}}(\lambda)$ *be arbitrary polynomials, which serve as parameters to* $\mathsf{A}_{\mathsf{unbounded}}$ *and* $\mathsf{A}_{\mathsf{prog}}$*. Then:*

$$\begin{aligned} &\Pr[\mathcal{R}^{\mathsf{A}_{\mathsf{prog}}} \textit{ outputs 1}] \geq \\ &\Pr[\mathcal{R}^{\mathsf{A}_{\mathsf{unbounded}}} \textit{ outputs 1}] - k \cdot \left(\frac{(q_{\mathsf{H}} + q_{\textsc{Sign}}) \cdot \ell_{\mathsf{t}}}{\mathrm{poly}_{\mathsf{bound}}(\lambda)} + \frac{1}{\mathrm{poly}_{\mathsf{filter}}(\lambda)} \right) - \mathsf{negl}(\lambda) \ . \end{aligned}$$

Proof. Let Hyb_0 be identical to $\mathcal{R}^{\mathsf{A}_{\mathsf{unbounded}}}$, except that Hyb_0 immediately exits with output 0 whenever the first step of $\mathsf{A}_{\mathsf{unbounded}}$, which is

$$\mathcal{T} \leftarrow \textsc{Filter}^{\mathcal{O}_{\mathsf{H}}, \textsc{Sign}}(\mathrm{pk})$$

(where the oracle Sign is provided by $\mathcal{R}$), produces a set $\mathcal{T}$ for which (1) $\mathcal{T} \neq \bot$, *and* (2) the filtering condition on $\mathcal{T}$ is not fulfilled. (Note that we make this change *for all* of the $\leq k$ invocations of $\mathsf{A}_{\mathsf{unbounded}}$.)

By Lemma 3, note that the statistical distance between $\mathcal{R}^{\mathsf{A}_{\mathsf{unbounded}}}$ and Hyb_0 is bounded by

$$k \cdot \left(\frac{(q_{\mathsf{H}} + q_{\textsc{Sign}}) \cdot \ell_{\mathsf{t}}}{\mathrm{poly}_{\mathsf{bound}}(\lambda)} \right) + \mathsf{negl}(\lambda) \ .$$

Next, let Hyb_1 be identical to Hyb_0, except that $\mathsf{A}_{\mathsf{unbounded}}$'s invocation of Sample is replaced by IdealSample, as defined by Lemma 5. Applying Lemma 5 for k times (each time for some procedure $\mathcal{R}'$ that runs all of Hyb_0 itself except the i-th run of Sample), a straightforward hybrid argument shows $\mathsf{Hyb}_1 \approx_{\mathsf{negl}(\lambda)} \mathsf{Hyb}_0$.

Finally, let Hyb_2 be identical to Hyb_1, except that IdealSample is replaced by sampling a triple (m, r, t) in the same way as $\mathsf{A}_{\mathsf{prog}}$ does:

$$m \leftarrow\!\$\ \mathcal{M}$$
$$(r, t) \leftarrow\!\$\ \mathcal{T}_m\ ,$$

(or aborting whenever $\mathcal{T}_m = \emptyset$).

Clearly, $\alpha^{\mathcal{T}}_{m,r,t}/(1-\beta^{\mathcal{T}})$ is equal to the probability of obtaining (m, r, t) through the sampling method above when conditioning on $|R^{\mathcal{T}}_{m,i}(\mathbf{x}, \mathbf{y}, f)| \geq \frac{2^{\ell_{\mathsf{r}}}}{\mathrm{poly}_{\mathsf{bound}}(\lambda)}$ (with $\mathbf{y} := A(s, m, r, t) \cdot \mathbf{x}$, $f := f(s, m, t)$, and $i := i^{\mathcal{T}}_{m,r}(\mathbf{x}, \mathbf{y}, f)$). Therefore, the statistical distance between running IdealSample and the sampling method above is exactly

$$\Pr\left[\begin{array}{l|l} \mathcal{T}_m \neq \emptyset \vee & m \leftarrow\!\$\ \mathcal{M} \\ |R^{\mathcal{T}}_{m,i}(\mathbf{x}, \mathbf{y}, f)| < \frac{2^{\ell_{\mathsf{r}}}}{\mathrm{poly}_{\mathsf{bound}}(\lambda)} & (r, t) \leftarrow\!\$\ \mathcal{T}_m \end{array}\right] \leq \frac{1}{\mathrm{poly}_{\mathsf{filter}}(\lambda)}$$

whenever the filtering condition is fulfilled. Since Hyb_2 aborts with output 0 whenever for some $\mathcal{T} \neq \bot$ (as computed by Filter) the filtering condition is not fulfilled, we get $\mathsf{Hyb}_2 \approx_{1/\mathrm{poly}_{\mathsf{filter}}(\lambda)} \mathsf{Hyb}_1$.

To conclude, note that Hyb_2 is identical to $\mathcal{R}^{\mathsf{A}_{\mathsf{prog}}}$, except that it has some additional aborts (whenever $\mathcal{T} \neq \bot$ does not fulfill the filtering condition). Therefore, $\Pr[\mathcal{R}^{\mathsf{A}_{\mathsf{prog}}} \text{ outputs } 1] \geq \Pr[\mathsf{Hyb}_2 \text{ outputs } 1]$. □

In addition, the unbounded adversary has sufficient success probability:

Lemma 7. *For all polynomials* $\mathrm{poly}_{\mathsf{bound}}(\lambda)$ *and* $\mathrm{poly}_{\mathsf{filter}}(\lambda)$ *that parameterize* $\mathsf{A}_{\mathsf{unbounded}}$*, the adversary* $\mathsf{A}_{\mathsf{unbounded}}$ *breaks EUF-CMA security:*

$$\mathrm{Adv}^{\mathsf{euf\text{-}cma}}_{\Sigma, \mathsf{A}_{\mathsf{unbounded}}}(\lambda) \geq 1 - (q_{\mathsf{H}} + q_{\mathrm{Sign}}) \cdot \frac{\ell_{\mathsf{t}}}{\mathrm{poly}_{\mathsf{bound}}(\lambda)} - \mathsf{negl}(\lambda)\ .$$

Proof. Combining Lemmas 2 and 3, the probability that $\mathsf{A}_{\mathsf{unbounded}}$ aborts due to $\mathcal{T} = \bot$ or the filtering condition not being fulfilled, is

$$\leq (q_{\mathsf{H}} + q_{\mathrm{Sign}}) \cdot \frac{\ell_{\mathsf{t}}}{\mathrm{poly}_{\mathsf{bound}}(\lambda)} + \mathsf{negl}(\lambda)\ .$$

Furthermore, Lemma 5 (applied to the procedure $\mathcal{R}$ that is exactly the euf-cma challenger) shows that conditioned on the above not happening, $\mathsf{A}_{\mathsf{unbounded}}$ always returns a valid forgery with probability $1 - \mathsf{negl}(\lambda)$. □

Combining these two results, we we get the main result stated in Theorem 3:

Proof (of Theorem 3). Suppose that there exists a tight, generic reduction R in non-observable ROM mode that proves EUF-CMA security of Σ under some generically true assumption $(\mathcal{P}, \epsilon)$. By definition, there exists some polynomial $\mathrm{loss}(\lambda)$, s.t. for every polynomial bound $q_{\mathrm{Sign}}(\lambda)$ on the number of signature queries and for every generic adversary A (making $\leq q_{\mathrm{Sign}}(\lambda)$ signature queries and $\leq \mathrm{poly}(\lambda)$ queries to $\mathcal{O}_{\mathsf{H}}$):

$$\frac{\mathrm{Adv}^{\mathsf{euf\text{-}cma}}_{\Sigma, \mathsf{A}}(\lambda)}{\mathrm{Adv}^{\mathcal{P}}_{\mathsf{R}^{\mathsf{A}}_{q_{\mathrm{Sign}}}}(\lambda)} \cdot k(\lambda) \leq \mathrm{loss}(\lambda) + \mathsf{negl}(\lambda)\ ,$$

where $1 \leq k(\lambda) \leq \mathrm{poly}(\lambda)$ is a bound on the number of euf-cma-sessions initiated by R. We choose

$$\begin{aligned} \mathrm{poly}_{\mathsf{filter}}(\lambda) &:= 3 \cdot \mathrm{loss}(\lambda)\ , \\ q_{\mathrm{Sign}}(\lambda) &:= 8(\mathrm{poly}_{\mathsf{filter}}(\lambda))^3 \cdot (\ell_{\mathsf{t}} + \ell_{\mathsf{r}}) \cdot (\lambda + \ell_{\mathsf{g}} \cdot \lceil \log p \rceil)\ , \text{ and} \\ \mathrm{poly}_{\mathsf{bound}}(\lambda) &:= 6 \cdot (q_{\mathsf{H}} + q_{\mathrm{Sign}}(\lambda)) \cdot \ell_{\mathsf{t}} \cdot \mathrm{loss}(\lambda)\ . \end{aligned}$$

where q_{H} is an upper bound on the number of $\mathcal{O}_{\mathsf{H}}$-queries made by R (which may depend on q_{SIGN} but always remains polynomial by assumption).

The resulting adversary $\mathsf{A}_{\mathsf{unbounded}}$ makes at most $q_{\text{SIGN}}(\lambda)$ signing queries by our construction of FILTER, and also clearly $\leq \text{poly}(\lambda)$ queries to $\mathcal{O}_{\mathsf{H}}$ – one per signature query during FILTER, and one per iteration of SAMPLE. Furthermore, by Lemma 7, it has advantage $\text{Adv}^{\mathsf{euf\text{-}cma}}_{\Sigma,\mathsf{A}_{\mathsf{unbounded}}}(\lambda) \geq 1 - (q_{\mathsf{H}} + q_{\text{SIGN}}) \cdot \frac{\ell_{\mathsf{t}}}{\text{poly}_{\mathsf{bound}}(\lambda)} - \mathsf{negl}(\lambda)$. Therefore, plugging $\mathsf{A}_{\mathsf{unbounded}}$ into the tightness bound, we get

$$\begin{aligned}\text{Adv}^{\mathcal{P}}_{\mathsf{R}^{\mathsf{A}_{\mathsf{unbounded}}}_{q_{\text{SIGN}}}}(\lambda) &\geq \frac{k(\lambda)}{\text{loss}(\lambda)} - \frac{k(\lambda) \cdot (q_{\mathsf{H}} + q_{\text{SIGN}}) \cdot \ell_{\mathsf{t}}}{\text{loss}(\lambda) \cdot \text{poly}_{\mathsf{bound}}(\lambda)} - \mathsf{negl}(\lambda) \\ &\geq \frac{k(\lambda)}{\text{loss}(\lambda)} - \frac{k(\lambda)}{6 \cdot \text{loss}(\lambda)} - \mathsf{negl}(\lambda) = \frac{5 \cdot k(\lambda)}{6 \cdot \text{loss}(\lambda)} - \mathsf{negl}(\lambda) \ .\end{aligned}$$

We now apply Lemma 6 (on the procedure $\mathcal{R}$ that is defined as the combination of assumption $\mathcal{P}$ and reduction $\mathsf{R}_{q_{\text{SIGN}}}$), which bounds the statistical distance between $\text{Adv}^{\mathcal{P}}_{\mathsf{R}^{\mathsf{A}_{\mathsf{unbounded}}}_{q_{\text{SIGN}}}}(\lambda)$ and $\text{Adv}^{\mathcal{P}}_{\mathsf{R}^{\mathsf{A}_{\mathsf{prog}}}_{q_{\text{SIGN}}}}(\lambda)$, we get

$$\begin{aligned}\text{Adv}^{\mathcal{P}}_{\mathsf{R}^{\mathsf{A}_{\mathsf{prog}}}_{q_{\text{SIGN}}}}(\lambda) &\geq \frac{5 \cdot k(\lambda)}{6 \cdot \text{loss}(\lambda)} - \frac{k(\lambda) \cdot (q_{\mathsf{H}} + q_{\text{SIGN}}) \cdot \ell_{\mathsf{t}}}{\text{poly}_{\mathsf{bound}}(\lambda)} - \frac{k(\lambda)}{\text{poly}_{\mathsf{filter}}(\lambda)} - \mathsf{negl}(\lambda) \\ &\geq \frac{5 \cdot k(\lambda)}{6 \cdot \text{loss}(\lambda)} - \frac{k(\lambda)}{6 \cdot \text{loss}(\lambda)} - \frac{k(\lambda)}{3 \cdot \text{loss}(\lambda)} - \mathsf{negl}(\lambda) \\ &= \frac{k(\lambda)}{3 \cdot \text{loss}(\lambda)} - \mathsf{negl}(\lambda) \geq \frac{1}{3 \cdot \text{loss}(\lambda)} - \mathsf{negl}(\lambda) \ .\end{aligned}$$

Note that $\mathsf{R}^{\mathsf{A}_{\mathsf{prog}}}_{q_{\text{SIGN}}}$ only performs polynomially many group-operations. Therefore, due to $\text{loss}(\lambda)$ being a polynomial in λ, $\mathsf{R}^{\mathsf{A}_{\mathsf{prog}}}_{q_{\text{SIGN}}}$ breaks the generic assumption $(\mathcal{P}, \epsilon)$. □

5.6 Helpful Lemma: unlikelihood of useful preexisting queries

The following helpful lemma will be used multiple times throughout the remaining proofs.

Lemma 8. *Fix some* $\text{pk} = (\mathbf{g}, s)$ *and an arbitrary set* $\mathcal{Q} \subseteq \mathbb{G}^* \times \{0,1\}^*$. *Then, when generating* $m \leftarrow\!\!\$\ \mathcal{M}$ *uniformly, the probability of there being an RO input in* $\mathcal{Q}$ *that could appear in a valid signature of* m *is bounded by* $\frac{|\mathcal{Q}|}{|\mathcal{M}|}$, *i.e.,*

$$\Pr[\exists r, t \text{ with } (A(s,m,r,t) \cdot \mathbf{g}, f(s,m,t)) \in \mathcal{Q} \mid m \leftarrow\!\!\$\ \mathcal{M}] \leq \frac{|\mathcal{Q}|}{|\mathcal{M}|}.$$

Proof. It suffices to show that for every $(\mathbf{h}, f) \in \mathcal{Q}$,

$$\Pr[\exists r, t \text{ with } \mathbf{h} = A(s,m,r,t) \cdot \mathbf{g} \wedge f = f(s,m,t) \mid m \leftarrow\!\!\$\ \mathcal{M}] \leq \frac{1}{|\mathcal{M}|} \ ,$$

since then the full lemma follows by union-bound.

Note that for each input $(\mathbf{h}, f) \in \mathcal{Q}$, there cannot exist more than one message $m^* \in \mathcal{M}$ for which there are values r and t s.t. $A(s, m^*, r, t) \cdot \mathbf{g} = \mathbf{h}$ and $f(s,m,t) = f$. This is an immediate consequence of the technical restriction that we have put on algebraic signature schemes that ensures that each RO input needs to uniquely determine the message m whose signature it is used in (see Definition 5).

Therefore, the probability in the equation above is bounded by

$$\leq \Pr[m = m^* \mid m \leftarrow\!\!\$\ \mathcal{M}] \leq \frac{1}{|\mathcal{M}|} \ .$$

□

5.7 Proof of Lemma 2

In the following proof, we will make use the following random variables: $\mathcal{G}^{(j)}$ denotes the set $\mathcal{G}$ at the end of phase 1 during the j-th iteration, and $\mathbf{x}^{(j)}$ denotes the element that is picked in line 9 during the j-th iteration (both may be $\perp$ if the j-th iteration, or an earlier iteration, aborted). Furthermore, $\mathcal{T}^{(j)}$ denotes the state of $\mathcal{T}$ *after* j iterations (or $\perp$ if the j-th iteration aborted). For convenience, if Filter ran a total of k iterations (potentially having aborted in the k-th of them), then we define $\mathcal{T}^{(j)} := \mathcal{T}^{(k)}$ for every $j > k$, and we define $\mathcal{T}^* := \mathcal{T}^{(k)}$ to be the final set of triples (note that we can have $\mathcal{T}^* \neq \perp$ despite $\mathcal{T} = \perp$, which happens whenever Filter does not conclude within $2\mathrm{poly}_{\mathsf{filter}}(\lambda) \cdot (\ell_{\mathsf{t}} + \ell_{\mathsf{r}})$ iterations).

Whenever there was no abort in line 5 (i.e., $\mathcal{T}^* \neq \perp$), we use $\mathcal{M}_{\mathsf{bad}} := \{m \in \mathcal{M} \mid \mathcal{T}^*_m = \emptyset\}$ to denote the set of messages m for which no corresponding triples $(m, r, t) \in \mathcal{T}^*$ exists in the end. Before proving Lemma 2, we will need the following, showing that w.h.p., there is only a small number of "bad" messages without triples.

Lemma 9. *For every fixed RO $\mathcal{O}_{\mathsf{H}}$, and for every fixed keypair* $(\mathrm{sk}, \mathrm{pk})$*, we have*

$$\Pr_{\textsc{Filter}}\left[\mathcal{T}^* \neq \perp \wedge \frac{|\mathcal{M}_{\mathsf{bad}}|}{|\mathcal{M}|} > \frac{1}{2\mathrm{poly}_{\mathsf{filter}}(\lambda)}\right] \leq \mathsf{negl}(\lambda) \ ,$$

where this probability is taken over the execution of $\textsc{Filter}^{\mathcal{O}_{\mathsf{H}},\textsc{Sign}^{\mathcal{O}_{\mathsf{H}}}(\mathrm{sk},\cdot)}(\mathrm{pk})$.

Proof. Let $\mathrm{pk} = (\mathbf{g}, s)$ be fixed. In the following, whenever m and a corresponding signature $\sigma = (\mathbf{h}, t)$ are defined,[19] writing r implicitly refers to the value $\mathcal{O}_{\mathsf{H}}(\mathbf{h}, f(s, m, t))$, A refers to $A(s, m, r, t)$, and f refers to $f(s, m, t)$. If, in addition, a dlog-candidate $\mathbf{x}$ is defined, then we implicitly use $\mathbf{y} := A \cdot \mathbf{x}$, and i to implicitly mean $i := i_{m,r}(\mathbf{x}, \mathbf{y}, f)$.

Let us consider a fixed a fixed iteration j, a fixed dlog-candidate $\mathbf{x} \in \mathbb{Z}_p^{\ell_{\mathsf{g}}}$, and a fixed set of triples $\mathcal{T}^{(j-1)}$ that remain before the j-th iteration. Given this, we may ask whether the following holds:

$$\Pr_{\substack{m \leftarrow\!\$ \mathcal{M} \\ \sigma \leftarrow \textsc{Sign}^{\mathcal{O}_{\mathsf{H}}}(\mathrm{sk},m)}}\left[|R^{\mathcal{T}^{(j-1)}}_{m,i}(\mathbf{x}, \mathbf{y}, f)| < \frac{2^{\ell_{\mathsf{r}}}}{\mathrm{poly}_{\mathsf{bound}}(\lambda)}\right] \geq \frac{1}{4(\mathrm{poly}_{\mathsf{filter}}(\lambda))^2 \cdot (\ell_{\mathsf{t}} + \ell_{\mathsf{r}})} \ . \tag{7}$$

(We stress that this probability is *not* taken over keypair generation or an execution of Filter. Instead, the inequality above denotes, for some given $\mathbf{x}$ and $\mathcal{T}^{(j-1)}$, an *event* that may or may not be fulfilled in an execution of the procedure Filter.) Intuitively, if Equation 7 is true for some j, $\mathbf{x}$, and $\mathcal{T}^{(j-1)}$, then it is highly likely (over the randomness in the j-th iteration) that $\mathbf{x} \notin \mathcal{G}^{(j)}$, because the probability of deleting $\mathbf{x}$ from $\mathcal{G}^{(j)}$ is $\geq \frac{1}{4(\mathrm{poly}_{\mathsf{filter}}(\lambda))^2 \cdot (\ell_{\mathsf{t}} + \ell_{\mathsf{r}})}$ for every single repetition of the loop in phase 1. More precisely, for every iteration j, $\mathbf{x} \in \mathbb{Z}_p^{\ell_{\mathsf{g}}}$, and $\mathcal{T}^{(j-1)}$ that fulfills Equation 7, we have

$$\begin{aligned}\Pr_{\textsc{FilterStep}}[\mathbf{x} \in \mathcal{G}^{(j)}] &\leq \left(1 - \frac{1}{4(\mathrm{poly}_{\mathsf{filter}}(\lambda))^2 \cdot (\ell_{\mathsf{t}} + \ell_{\mathsf{r}})}\right)^{4(\mathrm{poly}_{\mathsf{filter}}(\lambda))^2 \cdot (\ell_{\mathsf{t}} + \ell_{\mathsf{r}}) \cdot (\lambda + \ell_{\mathsf{g}} \cdot \lceil \log p \rceil)} \\ &\leq e^{-(\lambda + \ell_{\mathsf{g}} \cdot \log p)} \ ,\end{aligned}$$

where this probability is taken over the execution $\textsc{FilterStep}^{\mathcal{O}_{\mathsf{H}},\textsc{Sign}^{\mathcal{O}_{\mathsf{H}}}(\mathrm{sk},\cdot)}(\mathrm{pk}, \mathcal{T}^{(j-1)})$ (including all randomness used by Sign; but recall that $\mathcal{O}_{\mathsf{H}}$ is fixed).

Denote by $\mathsf{GoodChoice}^{(j)}$ the event that either $\mathbf{x}^{(j)} = \perp$ or Equation 7 is false for $\mathbf{x}^{(j)}$. Because $\mathbf{x}^{(j)}$ is chosen from the set $\mathcal{G}^{(j)}$ (unless $\mathbf{x}^{(j)} = \perp$), $\mathsf{GoodChoice}^{(j)}$ must be true whenever there is no $\mathbf{x} \in \mathcal{G}^{(j)}$ that fulfills Equation 7. Hence, by union bound over all $\mathbf{x} \in \mathbb{Z}_p^{\ell_{\mathsf{g}}}$ that fulfill Equation 7 of the

[19] Here, we do not impose validity on the signature σ. W.l.o.g., $\textsc{Sign}(\mathrm{sk}, m)$ always returns *some* signature, but it is not required to always be valid.

calculation above:

$$\Pr_{\text{FILTER}}[\mathsf{GoodChoice}^{(j)}] \geq 1 - e^{-(\lambda+\ell_{\mathsf{g}}\cdot\lceil\log p\rceil)} \cdot |\mathbb{Z}_p^{\ell_{\mathsf{g}}}| = 1 - e^{-\lambda} .$$

Let $\mathsf{GoodChoices}$ denote the event that $\mathsf{GoodChoice}^{(j)}$ holds for all iterations j. Then, by union-bound over all j,

$$\Pr_{\text{FILTER}}[\mathsf{GoodChoices}] \geq 2\text{poly}_{\mathsf{filter}}(\lambda) \cdot (\ell_{\mathsf{t}} + \ell_{\mathsf{r}}) \cdot e^{-\lambda} > 1 - \mathsf{negl}(\lambda) .$$

Whenever $\mathsf{GoodChoice}^{(j)}$ is true and $\mathcal{T}^{(j)} \neq \bot$, then, by the negation of Equation 7 and by the definition of FILTERSTEP, we have

$$\Pr_{\substack{m\leftarrow\$\mathcal{M}\\ \sigma\leftarrow\text{SIGN}^{\mathcal{O}_{\mathsf{H}}}(\text{sk},m)}}\left[(m,r,t)\in\mathcal{T}^{(j-1)} \Rightarrow (m,r,t)\in\mathcal{T}^{(j)}\right] \geq 1 - \frac{1}{4(\text{poly}_{\mathsf{filter}}(\lambda))^2\cdot(\ell_{\mathsf{t}}+\ell_{\mathsf{r}})} .$$

Note that $\mathcal{T}^{(0)} = \mathcal{M}\times\{0,1\}^{\ell_{\mathsf{r}}}\times\{0,1\}^{\ell_{\mathsf{t}}}$. Therefore, using a union bound, for an entire filtering process with $\leq 2\text{poly}_{\mathsf{filter}}(\lambda)\cdot(\ell_{\mathsf{t}}+\ell_{\mathsf{r}})$ iterations in which $\mathsf{GoodChoices}$ holds and $\mathcal{T}^* \neq \bot$, the following will be satisfied at the end:

$$\Pr_{m\leftarrow\$\mathcal{M}}[\mathcal{T}_m^* = \emptyset] \leq \Pr_{\substack{m\leftarrow\$\mathcal{M}\\ \sigma\leftarrow\text{SIGN}^{\mathcal{O}_{\mathsf{H}}}(\text{sk},m)}}[(r,t)\notin\mathcal{T}_m^*] \leq \frac{1}{2\text{poly}_{\mathsf{filter}}(\lambda)} .$$

Recall that $m\in\mathcal{M}_{\mathsf{bad}}$ means $\mathcal{T}_m^* = \emptyset$, and therefore (if $\mathsf{GoodChoices}$ holds and $\mathcal{T}^*\neq\bot$), $\frac{|\mathcal{M}_{\mathsf{bad}}|}{|\mathcal{M}|} \leq \frac{1}{2\text{poly}_{\mathsf{filter}}(\lambda)}$. Thus, $\Pr[\mathsf{GoodChoices}] \geq 1-\mathsf{negl}(\lambda)$ implies the lemma. □

Proof (of Lemma 2). Note that FILTER only outputs $\mathcal{T}=\bot$ whenever some iteration aborted due to an invalid signature in line 5, or it has exceeded $2\text{poly}_{\mathsf{filter}}(\lambda)\cdot(\ell_{\mathsf{t}}+\ell_{\mathsf{r}})$ iterations without converging. The first case (equivalent to $\mathcal{T}^*=\bot$) has negligible probability (by correctness of the signature scheme), and therefore we only prove that the second case has negligible probability. To do so, fix any RO $\mathcal{O}_{\mathsf{H}}$ and keypair (sk,pk). We will show that $\Pr[\mathcal{T}=\bot\wedge\mathcal{T}^*\neq\bot]\leq\mathsf{negl}(\lambda)$, taken over the execution of $\mathcal{T}\leftarrow\text{FILTER}^{\mathcal{O}_{\mathsf{H}},\text{SIGN}^{\mathcal{O}_{\mathsf{H}}}(\text{sk},\cdot)}(\text{pk})$.

Whenever $\mathcal{T}^*\neq\bot$, then for every iteration j and message m, we denote by

$$p_m^{(j)} := 1 - \frac{|\mathcal{T}_m^{(j)}|}{|\mathcal{T}_m^{(j-1)}|} \text{ if } \mathcal{T}_m^{(j-1)}\neq\emptyset \quad \text{or} \quad p_m^{(j)} := 1 \text{ if } \mathcal{T}_m^{(j-1)} = \emptyset$$

the fraction of $\mathcal{T}_m^{(j)}$ that has been deleted from $\mathcal{T}_m$ in the j-th iteration. For any given message m, we also define $p_m := \sum_{j\in[2\text{poly}_{\mathsf{filter}}(\lambda)\cdot(\ell_{\mathsf{t}}+\ell_{\mathsf{r}})]} p_m^{(j)}$ as the sum of fractions deleted from $\mathcal{T}_m$ over time.

We will now show that the following two things hold whenever $\mathcal{T}^*\neq\bot$:

1. For every $m\in\mathcal{M}$ with $p_m\geq\ell_{\mathsf{t}}+\ell_{\mathsf{r}}$, we also have $m\in\mathcal{M}_{\mathsf{bad}}$.
2. If $\mathcal{T}=\bot$, there must be strictly more than $\frac{|\mathcal{M}|}{2\text{poly}_{\mathsf{filter}}(\lambda)}$ messages $m\in\mathcal{M}$ for which $p_m\geq\ell_{\mathsf{t}}+\ell_{\mathsf{r}}$.

These claims together imply that it is impossible for $\mathcal{T}=\bot$, $\mathcal{T}^*\neq\bot$, and $|\mathcal{M}_{\mathsf{bad}}|\leq\frac{|\mathcal{M}|}{2\text{poly}_{\mathsf{filter}}(\lambda)}$ to occur at the same time (i.e., that FILTER aborts due to exceeding the iteration limit while $\mathcal{M}_{\mathsf{bad}}$ is small). Together with Lemma 9, this already shows $\Pr[\mathcal{T}=\bot\wedge\mathcal{T}^*\neq\bot]\leq\mathsf{negl}(\lambda)$, and therefore concludes the proof of Lemma 2.

We start with proving the first claim. Recall that $m\in\mathcal{M}_{\mathsf{bad}}$ means that at the end of FILTER, the set $\mathcal{T}_m=\emptyset$ is empty. Also note that in the very beginning, we have $|\mathcal{T}_m^{(0)}|\leq 2^{\ell_{\mathsf{t}}+\ell_{\mathsf{r}}}$ due to

$\mathcal{T}_m^{(0)} = \{0,1\}^{\ell_r} \times \{0,1\}^{\ell_t}$. Therefore:

$$\begin{aligned}
|\mathcal{T}_m| &= |\mathcal{T}_m^{(2\mathrm{poly}_{\mathsf{filter}}(\lambda)\cdot(\ell_t+\ell_r))}| = |\mathcal{T}_m^{(0)}| \cdot \prod_{j\in[2\mathrm{poly}_{\mathsf{filter}}(\lambda)\cdot(\ell_t+\ell_r)]} (1-p_m^{(j)}) \\
&\leq |\mathcal{T}_m^{(0)}| \cdot \Big(1 - \frac{1}{2\mathrm{poly}_{\mathsf{filter}}(\lambda)\cdot(\ell_t+\ell_r)} \cdot \underbrace{p_m}_{\geq \ell_t+\ell_r}\Big)^{2\mathrm{poly}_{\mathsf{filter}}(\lambda)\cdot(\ell_t+\ell_r)} \\
&\leq |\mathcal{T}_m^{(0)}| \cdot e^{-\ell_t-\ell_r} \leq (2/e)^{\ell_t+\ell_r} < 1
\end{aligned}$$

(The first inequality follows from $\prod_{i\in[n]}(1-z_i) \leq (1-\frac{1}{n}\sum_{i\in[n]} z_i)^n$ for non-negative z_i's, which is exactly the AM/GM inequality when applied to the terms $1-z_i$ for $i \in [n]$.) This shows $|\mathcal{T}_m| = 0$, i.e., $m \in \mathcal{M}_{\mathsf{bad}}$.

Now we prove the second claim. Note that for each iteration $j \in [2\mathrm{poly}_{\mathsf{filter}}(\lambda)\cdot(\ell_t+\ell_r)]$, unless filtering has already terminated, deletion in phase 2 is performed using a dlog-candidate $\mathbf{x} \in \mathbb{Z}_p^{\ell_g}$ for which the filtering condition is not fulfilled, i.e.,

$$\Pr\left[\begin{array}{l|l} \mathcal{T}_m^{(j-1)} = \emptyset \vee & m \leftarrow\!\$\, \mathcal{M} \\ |R_{m,i}^{\mathcal{T}^{(j-1)}}(\mathbf{x},\mathbf{y},f)| < \frac{2^{\ell_r}}{\mathrm{poly}_{\mathsf{bound}}(\lambda)} & (r,t) \leftarrow\!\$\, \mathcal{T}_m^{(j-1)} \end{array}\right] > \frac{1}{\mathrm{poly}_{\mathsf{filter}}(\lambda)} ,$$

where the values f, $\mathbf{y}$, and i are defined as $f := f(s,m,t)$, $\mathbf{y} := A(s,m,r,t)\cdot\mathbf{x}$, and $i := i_{m,r}^{\mathcal{T}}(\mathbf{x},\mathbf{y},f)$.

Since for each such m and $(r,t) \in \mathcal{T}_m^{(j-1)}$ with $|R_{m,i}^{\mathcal{T}^{(j-1)}}(\mathbf{x},\mathbf{y},f)| < \frac{2^{\ell_r}}{\mathrm{poly}_{\mathsf{bound}}(\lambda)}$, we have $(r,t) \notin \mathcal{T}_m^{(j)}$, the "average fraction" of deleted triples (average over all messages) is larger than $1/\mathrm{poly}_{\mathsf{filter}}(\lambda)$ in each iteration j:

$$\frac{1}{|\mathcal{M}|}\sum_{m\in\mathcal{M}} p_m^{(j)} > \frac{1}{\mathrm{poly}_{\mathsf{filter}}(\lambda)} .$$

Furthermore, summing up over all iterations (of which there are exactly $2\mathrm{poly}_{\mathsf{filter}}(\lambda)\cdot(\ell_t+\ell_r)$ whenever $\mathcal{T} = \perp$ and $\mathcal{T}^* \neq \perp$, because FILTER must exceed its iteration limit):

$$\sum_{j\in[2\mathrm{poly}_{\mathsf{filter}}(\lambda)\cdot(\ell_t+\ell_r)]} \sum_{m\in\mathcal{M}} p_m^{(j)} > 2(\ell_t+\ell_r)\cdot|\mathcal{M}| .$$

Now suppose that claim 2 does not hold, i.e., there are at most $\frac{|\mathcal{M}|}{2\mathrm{poly}_{\mathsf{filter}}(\lambda)}$ messages $m \in \mathcal{M}$ for which $p_m \geq \ell_t+\ell_r$. Then, the double-sum from the previous equation would have been $\leq \frac{|\mathcal{M}|}{2\mathrm{poly}_{\mathsf{filter}}(\lambda)} \cdot 2\mathrm{poly}_{\mathsf{filter}}(\lambda)\cdot(\ell_t+\ell_r) + |\mathcal{M}|\cdot(\ell_t+\ell_r) = 2(\ell_t+\ell_r)|\mathcal{M}|$ (because each p_m is bounded by the number of iterations for each $m \in \mathcal{M}$, i.e., $p_m \leq 2\mathrm{poly}_{\mathsf{filter}}(\lambda)\cdot(\ell_t+\ell_r)$), a contradiction. □

5.8 Proof of Lemma 3

We continue to use the same notation as in Section 5.7.

Proof (of Lemma 3). W.l.o.g., assume that $\mathcal{R}$ never makes the same query twice. Furthermore, assume that $\mathcal{R}$, for each signature $\sigma = (\mathbf{h},t)$ that it generates for a message m, has at some point before returning σ made the corresponding $\mathcal{O}_{\mathsf{H}}$-query $\mathcal{O}_{\mathsf{H}}(\mathbf{h}, f(s,m,t))$ that is needed to verify the signature. Note that this means that we can only assume $q_{\mathsf{H}} + q_{\mathrm{SIGN}}$ (instead of q_{H}) as an upper bound on the number of $\mathcal{O}_{\mathsf{H}}$-queries made by $\mathcal{R}$.

Denote by $\mathbf{x}$ the correct dlog of $\mathbf{g}$, i.e., $\mathbf{g} = \mathbf{x}\cdot\mathsf{g}_0$. Note that whenever $\mathbf{x} \in \mathcal{G}^{(j)}$ for every iteration j *and* FILTER outputs $\neq \perp$, then $\mathcal{T}$ must be correctly filtered (because once FILTER converges and outputs $\mathcal{T}$, $\mathbf{x}$ must fulfill Equation 6 for $\mathcal{T}$ due to the check on line 8). Therefore, we will only prove that with at least the desired bound of this lemma, $\mathbf{x} \in \mathcal{G}^{(j)}$ holds for every iteration j.

Fix any set $\mathcal{T}^*$, message $m \in \mathcal{M}$, $i^* \in [\ell_t]$, $\mathbf{y} \in \mathbb{Z}_p^{\ell_h}$, and bitstring f with $|R^{\mathcal{T}^*}_{m,i^*}(\mathbf{x},\mathbf{y},f)| < \frac{2^{\ell_r}}{\mathrm{poly}_{\mathsf{bound}}(\lambda)}$. Then it is easy to see that, if we would sample r uniformly at random (i.e., without consulting the RO $\mathcal{O}_\mathsf{H}$), it is unlikely that $i^{\mathcal{T}^*}_{m,r}(\mathbf{x},\mathbf{y},f)$ hits i^*:

$$\Pr_{r\leftarrow\$\{0,1\}^{\ell_r}}\left[i^{\mathcal{T}^*}_{m,r}(\mathbf{x},\mathbf{y},f) = i^*\right] < \frac{1}{\mathrm{poly}_{\mathsf{bound}}(\lambda)}\ .$$

(Note that this probability is not taken over the experiment described in this lemma, but over a uniformly random r). Furthermore, by union bound over those $i^* \in [\ell_t]$ with $|R^{\mathcal{T}^*}_{m,i^*}(\mathbf{x},\mathbf{y},f)| < \frac{2^{\ell_r}}{\mathrm{poly}_{\mathsf{bound}}(\lambda)}$:

$$\Pr_{r\leftarrow\$\{0,1\}^{\ell_r}}\left[i^{\mathcal{T}^*}_{m,r}(\mathbf{x},\mathbf{y},f) \neq \bot \wedge |R^{\mathcal{T}^*}_{m,i^{\mathcal{T}^*}_{m,r}(\mathbf{x},\mathbf{y},f)}(\mathbf{x},\mathbf{y},f)| < \frac{2^{\ell_r}}{\mathrm{poly}_{\mathsf{bound}}(\lambda)}\right] < \frac{\ell_t}{\mathrm{poly}_{\mathsf{bound}}(\lambda)}\ . \tag{8}$$

Now consider the experiment described in this lemma again, consisting of runinng $\mathcal{R}^{\mathcal{O}_\mathsf{H}}(1^\lambda)$ and $\textsc{Filter}^{\mathcal{O}_\mathsf{H},\mathcal{R}^{\mathcal{O}_\mathsf{H}}(\mathsf{sk},\cdot)}(\mathsf{pk})$. Fix some $k \in \mathbb{N}$, and let m_k be the random variable denoting $\textsc{Filter}$'s k-th signature query to $\mathcal{R}$. Let $\mathcal{Q}_k$ be the set of RO queries that has been made by anyone (either $\mathcal{R}$ or $\textsc{Filter}$) up to the moment of $\textsc{Filter}$'s k-th signature query. We define the event Bad_k as follows (intuitively, it means that there was already some RO query beforehand that can be used to construct a signature for message m_k):

$$\begin{aligned}\mathsf{Bad}_k := \exists(\mathbf{h},f) \in \mathcal{Q}_k, r \in \{0,1\}^{\ell_r}, t \in \{0,1\}^{\leq \ell_t} \text{ with}\\ f = f(s,m_k,t) \text{ and } \mathbf{h} = A(s,m_k,r,t)\cdot\mathbf{g}\end{aligned}$$

Lemma 8 immediately shows $\Pr[\mathsf{Bad}_k] \leq \frac{|\mathcal{Q}_k|}{|\mathcal{M}|} \leq \mathsf{negl}(\lambda)$ (due to $\mathcal{Q}_k$ having polynomial size and $\mathcal{M}$ being super-polynomially large).

Now fix some $q \in \mathbb{N}$ and let $(\mathbf{h}_q, f_q)$ be the q-th RO query made by $\mathcal{R}$ throughout the execution of $\textsc{Filter}^{\mathcal{O}_\mathsf{H},\mathcal{R}^{\mathcal{O}_\mathsf{H}}(\mathsf{sk},\cdot)}(\mathsf{pk})$. Let k_q be the random variable denoting that at the time of this q-th RO query, $\textsc{Filter}$ is currently waiting for the response to its k_q-th signature query to $\mathcal{R}$, and let j_q be the iteration in which $\textsc{Filter}$ made this k_q-th signature query. We further use $\mathbf{y}_q$ to denote the dlog of $\mathbf{h}_q$, i.e., $\mathbf{h}_q = \mathbf{y}_q \cdot \mathsf{g}_0$.

We say that the event Useful_q holds if the query $(\mathbf{h}_q, f_q)$ can potentially be used for constructing a signature σ for message m_{k_q}, i.e., if $r \in \{0,1\}^{\ell_r}$ and $t \in \{0,1\}^{\leq\ell_t}$ exist with $f_q = f(s,m_{k_q},t)$ and $\mathbf{h}_q = A(s,m_{k_q},r,t)\cdot\mathbf{g}$. Conditioned on $\overline{\mathsf{Bad}_{k_q}}$, we have either $\overline{\mathsf{Useful}_q}$, or else the query $\mathcal{O}_\mathsf{H}(\mathbf{h}_q,f_q)$ hasn't been made before. In the latter case, the output $r_q = \mathcal{O}_\mathsf{H}(\mathbf{h}_q,f_q)$ can be viewed as being uniformly sampled from $\{0,1\}^{\ell_r}$ *after* $\mathcal{T}^{(j_q-1)}$ has been chosen. Hence, Equation 8 yields

$$\begin{aligned}\Pr\left[\mathsf{Useful}_q \wedge i^{\mathcal{T}^{(j_q-1)}}_{m_{k_q},r_q}(\mathbf{x},\mathbf{y}_q,f_q) \neq \bot \wedge |R^{\mathcal{T}^{(j_q-1)}}_{m_{k_q},i^{\mathcal{T}^{(j_q-1)}}_{m_{k_q},r_q}(\mathbf{x},\mathbf{y}_q,f_q)}(\mathbf{x},\mathbf{y}_q,f_q)| < \frac{2^{\ell_r}}{\mathrm{poly}_{\mathsf{bound}}(\lambda)} \;\middle|\; \overline{\mathsf{Bad}_{k_q}}\right]\\ < \frac{\ell_t}{\mathrm{poly}_{\mathsf{bound}}(\lambda)}\ ,\end{aligned}$$

taken over the full experiment from this lemma. Together with a union-bound of Bad_k over all $k \in [q_{\textsc{Sign}}]$, we obtain

$$\begin{aligned}\Pr\left[\mathsf{Useful}_q \wedge i^{\mathcal{T}^{(j_q-1)}}_{m_{k_q},r_q}(\mathbf{x},\mathbf{y}_q,f_q) \neq \bot \wedge |R^{\mathcal{T}^{(j_q-1)}}_{m_{k_q},i^{\mathcal{T}^{(j_q-1)}}_{m_{k_q},r_q}(\mathbf{x},\mathbf{y}_q,f_q)}(\mathbf{x},\mathbf{y}_q,f_q)| < \frac{2^{\ell_r}}{\mathrm{poly}_{\mathsf{bound}}(\lambda)}\right]\\ < \frac{\ell_t}{\mathrm{poly}_{\mathsf{bound}}(\lambda)} + \mathsf{negl}(\lambda)\end{aligned} \tag{9}$$

for every $q \in [q_{\mathsf{H}} + q_{\text{SIGN}}]$. Note that if the event inside this probability does not hold, then there *exists no* bitstring t with $\mathbf{y}_q = A(s, m_q, r_q, t) \cdot \mathbf{x}$ and $f_q = f(s, m_q, t)$ for which $\sigma = (\mathbf{h}_q, t)$ is a valid signature that would led FILTER to delete $\mathbf{x}$ from $\mathcal{G}$ (compare the event in Equation 9 with the deletion condition in line 6). Intuitively, this means that the q-th RO query made by $\mathcal{R}$ is not helpful for making FILTER delete $\mathbf{x}$ from $\mathcal{G}$ in iteration j_q.

Therefore, by our assumption that for any signature σ sent to FILTER, the corresponding RO query required for verification has been made by $\mathcal{R}$ beforehand, we get

$$\Pr\left[\forall j : \mathbf{x} \in \mathcal{G}^{(j)}\right] \geq 1 - (q_{\mathsf{H}} + q_{\text{SIGN}}) \cdot \frac{\ell_{\mathsf{t}}}{\text{poly}_{\mathsf{bound}}(\lambda)} - \mathsf{negl}(\lambda) \ ,$$

which implies the full lemma as discussed before. □

5.9 Proof of Lemma 5

Before we can show Lemma 5, we need the following result about SAMPLESTEP.

Lemma 10. *Let* $\text{poly}_{\mathsf{bound}}(\lambda)$ *be an arbitrary polynomial that parameterizes* SAMPLESTEP. *Let* $\mathcal{R}$ *be a* stateful *procedure that first chooses* pk *and a set* $\mathcal{T}$*, then receives a triple* (m, r, t) *(or* $\perp$*) and finally outputs a bit* b*.* $\mathcal{R}$ *may* both query and program the RO $\mathcal{O}_{\mathsf{H}}$*, up to a total of* $q_{\mathsf{H}} \leq \text{poly}(\lambda)$ *times. Then, the distribution*

$$\left\{ b \;\middle|\; \begin{array}{l} (\text{pk} = (\mathbf{g}, s), \mathcal{T}) \leftarrow \mathcal{R}^{\mathcal{O}_{\mathsf{H}}}() \\ \textit{choose } \mathbf{x} \in \mathbb{Z}_p^{\ell_{\mathsf{g}}} \textit{ s.t. } \mathbf{g} = \mathbf{x} \cdot \mathsf{g}_0 \\ (m, r, t) \leftarrow \text{SAMPLESTEP}^{\mathcal{O}_{\mathsf{H}}}(\text{pk}, \mathbf{x}, \mathcal{T}) \\ b \leftarrow \mathcal{R}^{\mathcal{O}_{\mathsf{H}}}(m, r, t) \end{array} \right\}$$

differs statistically by at most $\mathsf{negl}(\lambda)$ *from the same distribution where* SAMPLESTEP *is replaced by* IDEALSAMPLESTEP*, defined as follows:*

IDEALSAMPLESTEP$(\text{pk}, \mathbf{x}, \mathcal{T})$

abort with probability $\beta^{\mathcal{T}}$

sample (m, r, t), each with probability $\alpha^{\mathcal{T}}_{m,r,t}/(1 - \beta^{\mathcal{T}})$

$\mathcal{O}_{\mathsf{H}}(A(s, m, r, t) \cdot \mathbf{g}, f(s, m, t)) := r$

return (m, r, t)

Proof. Let us denote by Hyb_0 the distribution that uses SAMPLESTEP, and by Hyb_2 the distribution that uses IDEALSAMPLESTEP.

We first introduce a hybrid Hyb_1 that is identical to the distribution that uses SAMPLESTEP, except that inside SAMPLESTEP, we replace

$$\begin{array}{l} r \leftarrow \mathcal{O}_{\mathsf{H}}(\mathbf{y} \cdot \mathsf{g}_0, f) \\ \texttt{// lines 8-9} \end{array} \quad \text{by} \quad \begin{array}{l} r \leftarrow \{0,1\}^{\ell_{\mathsf{r}}} \\ \texttt{// lines 8-9} \\ \mathcal{O}_{\mathsf{H}}(\mathbf{y} \cdot \mathsf{g}_0, f) := r \end{array} \tag{10}$$

i.e., we generate the RO output ourselves, and then program the RO on it whenever we didn't abort.

By laziness of the RO $\mathcal{O}_{\mathsf{H}}$, the distributions Hyb_0 and Hyb_1 are identical conditioned on the query $(\mathbf{y} \cdot \mathsf{g}_0, f)$ having never been made or programmed before. Furthermore, since $\mathcal{R}$ only makes/programs q_{H} many $\mathcal{O}_{\mathsf{H}}$-queries, and the query is chosen as $(\mathbf{y} \cdot \mathsf{g}_0, f) = (A(s, m, r, t) \cdot \mathbf{g}, f(s, m, t))$ for some randomly sampled $m \leftarrow\!\!\$\ \mathcal{M}$, Lemma 8 implies that the probability of $(\mathbf{y} \cdot \mathsf{g}_0, f)$ having been made/programmed before is at most $\frac{q_{\mathsf{H}}}{|\mathcal{M}|} < \mathsf{negl}(\lambda)$. Therefore, $\mathsf{Hyb}_0 \approx_{\mathsf{negl}(\lambda)} \mathsf{Hyb}_1$.

Now it remains to prove $\mathsf{Hyb}_1 \equiv \mathsf{Hyb}_2$, for which it suffices to show that Hyb_1 returns each triple $(m,r,t) \in \mathcal{T}$ with probability exactly $\alpha^{\mathcal{T}}_{m,r,t}$, and $\perp$ with remaining probability. (Note that Hyb_1 does not program the RO when it returns $\perp$, so its behavior matches that of Hyb_2 conditioned on aborting.)

Let us consider any $(m,r,t) \in \mathcal{T}$ with corresponding $f := f(s,m,t)$, $\mathbf{y} := A(s,m,r,t) \cdot \mathbf{x}$, and $i := i^{\mathcal{T}}_{m,r}(\mathbf{x},\mathbf{y},f)$. If $|R^{\mathcal{T}}_{m,i}(\mathbf{x},\mathbf{y},f)| < \frac{2^{\ell_r}}{\mathrm{poly}_{\mathsf{bound}}(\lambda)}$, then clearly the probability of returning (m,r,t) is 0 (due to the abort in line 5). Otherwise, the probability that, after line 5, the modified SampleStep didn't abort yet and it has selected these values for m, f, $\mathbf{y}$, and i, is exactly

$$\frac{1}{|\mathcal{M}|} \cdot \frac{\sum_{r \in R^{\mathcal{T}}_{m,i}(\mathbf{x},\mathbf{y},f)} |S^{\mathcal{T}}_{m,r}(\mathbf{x},\mathbf{y},f)|}{|\mathcal{T}_m|} \ ,$$

because the first factor denotes the probability of choosing $m \in \mathcal{M}$ correctly, and the second factor corresponds to sampling an element from $\mathcal{T}_m$ that produces the required values of f, $\mathbf{y}$, and i.

Then, the probability of surviving the coin flip in line 6 is

$$\frac{2^{\ell_r}}{\mathrm{poly}_{\mathsf{bound}}(\lambda) \cdot |R^{\mathcal{T}}_{m,i}(\mathbf{x},\mathbf{y},f)|} \cdot \frac{2^{i-1} \cdot |R^{\mathcal{T}}_{m,i}(\mathbf{x},\mathbf{y},f)|}{\sum_{r \in R^{\mathcal{T}}_{m,i}(\mathbf{x},\mathbf{y},f)} |S^{\mathcal{T}}_{m,r}(\mathbf{x},\mathbf{y},f)|} \ .$$

In addition, the probability of $\mathcal{O}_{\mathsf{H}}$ returning the correct r (in line 7) is $2^{-\ell_r}$, and the probability of sampling the correct t (in line 10) is 2^{-i}.

Hence, because all of these steps need to be fulfilled independently, the probability for SampleStep to produce tuple (m,r,t) is exactly the product of that of all individual steps, which is

$$\frac{1}{|\mathcal{M}| \cdot |\mathcal{T}_m| \cdot \mathrm{poly}_{\mathsf{bound}}(\lambda) \cdot 2} = \alpha^{\mathcal{T}}_{m,r,t} \ ,$$

as desired. □

Proof (of Lemma 5). Let us denote by Hyb_0 the distribution that uses Sample, and by Hyb_2 the distribution that uses IdealSample.

Let Hyb_1 be identical to Hyb_0, except that the loop inside Sample invokes $\mathrm{poly}_{\mathsf{bound}}(\lambda) \cdot 4 \cdot \lambda$ times $\textsc{IdealSampleStep}^{\mathcal{O}_{\mathsf{H}}}(\mathbf{x},\mathcal{T})$ instead of $\textsc{SampleStep}^{\mathcal{O}_{\mathsf{H}}}(\mathbf{x},\mathcal{T})$. By Lemma 10 (with a straightforward hybrid argument), $\mathsf{Hyb}_0 \approx_{\mathsf{negl}(\lambda)} \mathsf{Hyb}_1$. It remains to prove $\mathsf{Hyb}_1 \approx_{\mathsf{negl}(\lambda)} \mathsf{Hyb}_2$.

Note that Hyb_1 runs, inside a loop, $\mathrm{poly}_{\mathsf{bound}}(\lambda) \cdot 4 \cdot \lambda$ times $\textsc{IdealSampleStep}^{\mathcal{O}_{\mathsf{H}}}(\mathrm{pk},\mathbf{x},\mathcal{T})$ (and returns the first outcome that does not $\perp$). Hyb_2 simply runs $\textsc{IdealSample}^{\mathcal{O}_{\mathsf{H}}}(\mathrm{pk},\mathbf{x},\mathcal{T})$ instead (which never aborts). Hence, the distribution Hyb_1, conditioned on its modified Sample not aborting, is identical to the distribution Hyb_2 (to see this, note that IdealSampleStep, conditioned on not aborting, behaves identical to IdealSample; in addition, IdealSampleStep does not have any side-effects on the RO $\mathcal{O}_{\mathsf{H}}$ whenever it aborts).

We now prove that the probability of the modified Sample inside Hyb_1 to abort is negligible conditioned on $\mathcal{T} \neq \perp$ fulfilling the filtering condition. (Recall that whenever this condition is not fulfilled, then each of the hybrids would output 0 anyways.) Lemma 4 gives us the bound $\beta^{\mathcal{T}} \leq (1 - \frac{1}{\mathrm{poly}_{\mathsf{bound}}(\lambda) \cdot 4})$ on IdealSampleStep's abort probability. Therefore, all $\mathrm{poly}_{\mathsf{bound}}(\lambda) \cdot 4 \cdot \lambda$ independent iterations of Sample in Hyb_1 to fail has probability

$$(\beta^{\mathcal{T}})^{\mathrm{poly}_{\mathsf{bound}}(\lambda) \cdot 4 \cdot \lambda} \leq \left(1 - \frac{1}{\mathrm{poly}_{\mathsf{bound}}(\lambda) \cdot 4}\right)^{\mathrm{poly}_{\mathsf{bound}}(\lambda) \cdot 4 \cdot \lambda} \leq e^{-\lambda} \leq \mathsf{negl}(\lambda) \ ,$$

whenever the filtering condition is fulfilled, and hence $\mathsf{Hyb}_1 \approx_{\mathsf{negl}(\lambda)} \mathsf{Hyb}_2$. □

References

AB13. Prabhanjan Ananth and Raghav Bhaskar. Non observability in the random oracle model. In Willy Susilo and Reza Reyhanitabar, editors, *ProvSec 2013*, volume 8209 of *LNCS*, pages 86–103. Springer, Berlin, Heidelberg, October 2013.

AHN+23. Masayuki Abe, Dennis Hofheinz, Ryo Nishimaki, Miyako Ohkubo, and Jiaxin Pan. Compact structure-preserving signatures with almost tight security. *Journal of Cryptology*, 36(4):37, October 2023.

BD20. Mihir Bellare and Wei Dai. The multi-base discrete logarithm problem: Tight reductions and non-rewinding proofs for Schnorr identification and signatures. In Karthikeyan Bhargavan, Elisabeth Oswald, and Manoj Prabhakaran, editors, *INDOCRYPT 2020*, volume 12578 of *LNCS*, pages 529–552. Springer, Cham, December 2020.

BDF+11. Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. Random oracles in a quantum world. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 41–69. Springer, Berlin, Heidelberg, December 2011.

BG81. Charles H. Bennett and John Gill. Relative to a random oracle a, p^a != np^a != co-np^a with probability 1. *SIAM J. Comput.*, 10(1):96–113, 1981.

BHKR25. Nicholas Brandt, Dennis Hofheinz, Michael Klooß, and Michael Reichle. Tightly-secure blind signatures in pairing-free groups. In *ASIACRYPT 2025, Part VI*, LNCS, pages 337–369. Springer, Singapore, December 2025.

BL96. Dan Boneh and Richard J. Lipton. Algorithms for black-box fields and their application to cryptography (extended abstract). In Neal Koblitz, editor, *CRYPTO'96*, volume 1109 of *LNCS*, pages 283–297. Springer, Berlin, Heidelberg, August 1996.

BLS01. Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532. Springer, Berlin, Heidelberg, December 2001.

BR93. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93*, pages 62–73. ACM Press, November 1993.

CATZ24. Rutchathon Chairattana-Apirom, Stefano Tessaro, and Chenzhi Zhu. Pairing-free blind signatures from CDH assumptions. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part I*, volume 14920 of *LNCS*, pages 174–209. Springer, Cham, August 2024.

CDS94. Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Yvo Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 174–187. Springer, Berlin, Heidelberg, August 1994.

CFGG22. Dario Catalano, Dario Fiore, Rosario Gennaro, and Emanuele Giunta. On the impossibility of algebraic vector commitments in pairing-free groups. In Eike Kiltz and Vinod Vaikuntanathan, editors, *TCC 2022, Part II*, volume 13748 of *LNCS*, pages 274–299. Springer, Cham, November 2022.

CFO24. Gavin Cho, Georg Fuchsbauer, and Adam O'Neill. Schnorr signatures are tightly secure in the ROM under a non-interactive assumption. Cryptology ePrint Archive, Report 2024/1528, 2024.

Che05. Benoît Chevallier-Mames. An efficient CDH-based signature scheme with a tight security reduction. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 511–526. Springer, Berlin, Heidelberg, August 2005.

CJS14. Ran Canetti, Abhishek Jain, and Alessandra Scafuro. Practical UC security with a global random oracle. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 2014*, pages 597–608. ACM Press, November 2014.

Cor00. Jean-Sébastien Coron. On the exact security of full domain hash. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 229–235. Springer, Berlin, Heidelberg, August 2000.

CP93. David Chaum and Torben P. Pedersen. Wallet databases with observers. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 89–105. Springer, Berlin, Heidelberg, August 1993.

Cra96. Ronald Cramer. *Modular Design of Secure, yet Practical Cryptographic Protocols*. Phd thesis, University of Amsterdam, 1996.

DHH+21. Nico Döttling, Dominik Hartmann, Dennis Hofheinz, Eike Kiltz, Sven Schäge, and Bogdan Ursu. On the impossibility of purely algebraic signatures. In Kobbi Nissim and Brent Waters, editors, *TCC 2021, Part III*, volume 13044 of *LNCS*, pages 317–349. Springer, Cham, November 2021.

ElG84. Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 10–18. Springer, Berlin, Heidelberg, August 1984.

FH17. Masayuki Fukumitsu and Shingo Hasegawa. Impossibility of the provable security of the Schnorr signature from the one-more DL assumption in the non-programmable random oracle model. In Tatsuaki Okamoto, Yong Yu, Man Ho Au, and Yannan Li, editors, *ProvSec 2017*, volume 10592 of *LNCS*, pages 201–218. Springer, Cham, October 2017.

FHJ20. Marc Fischlin, Patrick Harasser, and Christian Janson. Signatures from sequential-OR proofs. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part III*, volume 12107 of *LNCS*, pages 212–244. Springer, Cham, May 2020.

FJS14. Nils Fleischhacker, Tibor Jager, and Dominique Schröder. On tight security proofs for Schnorr signatures. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 512–531. Springer, Berlin, Heidelberg, December 2014.

FKL18. Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Cham, August 2018.

FLR+10. Marc Fischlin, Anja Lehmann, Thomas Ristenpart, Thomas Shrimpton, Martijn Stam, and Stefano Tessaro. Random oracles with(out) programmability. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 303–320. Springer, Berlin, Heidelberg, December 2010.

FPS20. Georg Fuchsbauer, Antoine Plouviez, and Yannick Seurin. Blind Schnorr signatures and signed ElGamal encryption in the algebraic group model. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 63–95. Springer, Cham, May 2020.

FRS22. Marc Fischlin, Felix Rohrbach, and Tobias Schmalz. A random oracle for all of us. In Lejla Batina and Joan Daemen, editors, *AFRICACRYPT 22*, volume 2022 of *LNCS*, pages 469–489. Springer, Cham, July 2022.

FS87. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Berlin, Heidelberg, August 1987.

FS90. Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *22nd ACM STOC*, pages 416–426. ACM Press, May 1990.

GBL08. Sanjam Garg, Raghav Bhaskar, and Satyanarayana V. Lokam. Improved bounds on security reductions for discrete log based signatures. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 93–107. Springer, Berlin, Heidelberg, August 2008.

GJ03. Eu-Jin Goh and Stanislaw Jarecki. A signature scheme as secure as the Diffie-Hellman problem. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 401–415. Springer, Berlin, Heidelberg, May 2003.

Gro06. Jens Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In Xuejia Lai and Kefei Chen, editors, *ASIACRYPT 2006*, volume 4284 of *LNCS*, pages 444–459. Springer, Berlin, Heidelberg, December 2006.

GS08. Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, Berlin, Heidelberg, April 2008.

GTH26. Milan Gonzalez-Thauvin and Keitaro Hashimoto. Asymmetric message franking in the plain model: Generic and efficient constructions. In *CRYPTO 2026*, August 2026.

HJK12. Dennis Hofheinz, Tibor Jager, and Edward Knapp. Waters signatures with optimal security reduction. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 66–83. Springer, Berlin, Heidelberg, May 2012.

HK08. Dennis Hofheinz and Eike Kiltz. Programmable hash functions and their applications. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 21–38. Springer, Berlin, Heidelberg, August 2008.

JR13. Charanjit S. Jutla and Arnab Roy. Shorter quasi-adaptive NIZK proofs for linear subspaces. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 1–20. Springer, Berlin, Heidelberg, December 2013.

KMP16. Eike Kiltz, Daniel Masny, and Jiaxin Pan. Optimal security proofs for signatures from identification schemes. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 33–61. Springer, Berlin, Heidelberg, August 2016.

KRW24. Michael Klooß, Michael Reichle, and Benedikt Wagner. Practical blind signatures in pairing-free groups. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part I*, volume 15484 of *LNCS*, pages 363–395. Springer, Singapore, December 2024.

KW03. Jonathan Katz and Nan Wang. Efficiency improvements for signature schemes with tight security reductions. In Sushil Jajodia, Vijayalakshmi Atluri, and Trent Jaeger, editors, *ACM CCS 2003*, pages 155–164. ACM Press, October 2003.

Mau05. Ueli M. Maurer. Abstract models of computation in cryptography (invited paper). In Nigel P. Smart, editor, *10th IMA International Conference on Cryptography and Coding*, volume 3796 of *LNCS*, pages 1–12. Springer, Berlin, Heidelberg, December 2005.
Nec94. Vassiliy Ilyich Nechaev. Complexity of a determinate algorithm for the discrete logarithm. *Mathematical Notes*, 55(2):165–172, February 1994.
Nie02. Jesper Buus Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 111–126. Springer, Berlin, Heidelberg, August 2002.
Oka93. Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 31–53. Springer, Berlin, Heidelberg, August 1993.
PS96. David Pointcheval and Jacques Stern. Security proofs for signature schemes. In Ueli M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 387–398. Springer, Berlin, Heidelberg, May 1996.
PV05. Pascal Paillier and Damien Vergnaud. Discrete-log-based signatures may not be equivalent to discrete log. In Bimal K. Roy, editor, *ASIACRYPT 2005*, volume 3788 of *LNCS*, pages 1–20. Springer, Berlin, Heidelberg, December 2005.
RS21. Lior Rotem and Gil Segev. Tighter security for Schnorr identification and signatures: A high-moment forking lemma for Σ-protocols. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 222–250, Virtual Event, August 2021. Springer, Cham.
Sch90. Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 239–252. Springer, New York, August 1990.
Seu12. Yannick Seurin. On the exact security of Schnorr-type signatures in the random oracle model. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 554–571. Springer, Berlin, Heidelberg, April 2012.
Sho97. Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Berlin, Heidelberg, May 1997.
Unr07. Dominique Unruh. Random oracles and auxiliary input. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 205–223. Springer, Berlin, Heidelberg, August 2007.
Wat05. Brent R. Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 114–127. Springer, Berlin, Heidelberg, May 2005.
Zha22. Mark Zhandry. To label, or not to label (in generic groups). In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part III*, volume 13509 of *LNCS*, pages 66–96. Springer, Cham, August 2022.

A Instantiation of a Commit-and-Prove System in the Non-Observable, Non-Programmable ROM

In the following proof system, the $\mathbf{crs} = (\mathsf{g}_1, \mathsf{g}_2, \mathsf{g}_3, \mathsf{g}_4)$ first consists of a triple $(\mathsf{g}_1, \mathsf{g}_2, \mathsf{g}_3)$, which is either a DDH triple (in binding mode), or a uniformly random triple (in hiding mode). These group elements are used for constructing commitments. For the triple $(\mathsf{g}_1, \mathsf{g}_2, \mathsf{g}_4)$, the roles are switched: it forms a DDH triple in hiding mode but not in binding mode. This will allow proof simulations in hiding mode (and lets us prove invalid statements).

Construction 3 (Commit-and-Prove System using ROM) *We construct a commit-and-prove system using random oracles for the class of LCDs (Definition 12) as follows. The RO output space must be* $\mathcal{R}_{\mathsf{H}} = \mathbb{Z}_p$.

$\textsc{BGen}(1^\lambda)$**.** *Sample* $x, y \leftarrow\!\!\$\ \mathbb{Z}_p$ *and* $z \leftarrow\!\!\$\ \mathbb{Z}_p \setminus \{xy\}$*, and choose* $\mathbf{crs} := (x\mathsf{g}_0, y\mathsf{g}_0, (xy)\mathsf{g}_0, z\mathsf{g}_0)$.
$\textsc{HGen}(1^\lambda)$**.** *Sample* $x, y \leftarrow\!\!\$\ \mathbb{Z}_p$ *and* $z \leftarrow\!\!\$\ \mathbb{Z}_p \setminus \{xy\}$*, and choose* $\mathbf{crs} := (x\mathsf{g}_0, y\mathsf{g}_0, z\mathsf{g}_0, (xy)\mathsf{g}_0)$ *with* $\mathbf{trap} := x$.
$\textsc{Com}(\mathbf{crs}, \mathsf{v}; R)$**.** *Parse* $\mathbf{crs} = (\mathsf{g}_1, \mathsf{g}_2, \mathsf{g}_3, \mathsf{g}_4)$ *and randomness* $R = (s, t) \in \mathbb{Z}_p^2 =: \mathcal{R}_{\mathsf{com}}$*. Then return the commitment* $\mathrm{com} = (\mathsf{c}, \mathsf{c}')$*, consisting of the two group elements*

$$\mathsf{c} = s\mathsf{g}_0 + t\mathsf{g}_1 \ ,$$
$$\mathsf{c}' = s\mathsf{g}_2 + t\mathsf{g}_3 + \mathsf{v} \ .$$

$\text{PROVE}^{\mathcal{O}_\mathsf{H}}((\mathbf{crs}^{(j)}, L^{(j)}, \mathbf{u}^{(j)}, (\text{com}_i^{(j)}, R_i^{(j)})_{i\in[\ell_\mathsf{v}]}, w^{(j)})_{j\in[N]})$. *Parse* $\mathbf{crs}^{(j)} = (\mathsf{g}_1^{(j)}, \mathsf{g}_2^{(j)}, \mathsf{g}_3^{(j)}, \mathsf{g}_4^{(j)})$, $\text{com}_i^{(j)} = (\mathsf{c}_i^{(j)}, \mathsf{c}_i'^{(j)})$ *and* $R_i^{(j)} = (s_i^{(j)}, t_i^{(j)})$. *In the following, we use* $\mathbf{c} := (\mathsf{c}_1^{(1)}, \ldots, \mathsf{c}_{\ell_\mathsf{v}^{(1)}}^{(1)}, \ldots, \mathsf{c}_1^{(N)}, \ldots, \mathsf{c}_{\ell_\mathsf{v}^{(N)}}^{(N)})$ *to denote the vector of all commitment elements* $\mathsf{c}_i^{(j)}$, *and similarly for vectors* $\mathbf{c}'$, $\mathbf{s}$, *and* $\mathbf{t}$.
Let

$$L_{\text{ddh}} := \{((\mathsf{g}_0, \mathsf{g}_1, \mathsf{g}_2, \mathsf{g}_4), \epsilon) \mid \exists x \text{ s.t. } \mathsf{g}_1 = \mathsf{g}_0^x \wedge \mathsf{g}_4 = \mathsf{g}_2^x\}$$

be the language of DDH tuples (where ϵ *denotes an "empty" tuple, i.e.,* L_{ddh}*'s statements do not contain any committed group elements), and correspondingly*

$$L := (L^{(1)} \vee L_{\text{ddh}}) \wedge \cdots \wedge (L^{(N)} \vee L_{\text{ddh}})$$

the language that requires for each $j \in [N]$ *either a valid proof for language* $L^{(j)}$, *or* $\mathbf{crs}^{(j)}$ *being in hiding mode. Furthermore, let*

$$\mathbf{u} := \mathbf{u}^{(1)} \| (\mathsf{g}_0, \mathsf{g}_1^{(1)}, \mathsf{g}_2^{(1)}, \mathsf{g}_4^{(1)}) \| \ldots \| \mathbf{u}^{(N)} \| (\mathsf{g}_0, \mathsf{g}_1^{(N)}, \mathsf{g}_2^{(N)}, \mathsf{g}_4^{(N)})$$

be the corresponding non-committed part *of the statement (note that the committed part is given by* $\mathbf{c}$ *and* $\mathbf{c}'$*).*
For any $j \in [N]$, *we construct a witness* $\widetilde{w}^{(j)}$ *for*

$$(\mathbf{u}^{(j)} \| (\mathsf{g}_0, \mathsf{g}_1^{(j)}, \mathsf{g}_2^{(j)}, \mathsf{g}_4^{(j)}), \mathbf{v}^{(j)}) \in L^{(j)} \vee L_{\text{ddh}} \ ,$$

with $\mathbf{v}$ *being the group elements that the commitments* $\text{com}_i^{(j)}$ *refer to, as follows: If* $w^{(j)}$ *is a valid trapdoor for* $\mathbf{crs}^{(j)}$ *(i.e.,* $\mathsf{g}_2^{(j)} = w^{(j)}\mathsf{g}_0$ *and* $\mathsf{g}_4^{(j)} = w^{(j)}\mathsf{g}_1^{(j)}$*), then choose* $\widetilde{w}^{(j)} := (2, w^{(j)})$ *(because* $w^{(j)}$ *is a valid witness for* L_{ddh}*). Otherwise, choose* $\widetilde{w}^{(j)} := (1, w^{(j)})$ *(because* $w^{(j)}$ *must be a valid witness for* $L^{(j)}$*; or else the input to the prover was not valid). The overall witness for the language* L *is given by* $w := (\widetilde{w}^{(1)}, \ldots, \widetilde{w}^{(N)})$.
We now use the Fiat-Shamir transform in order to produce a proof that $(\mathbf{u}, \mathbf{v}) \in L$. *See Figures 8-10 for definitions of* P_1 *and* P_2.
1. *Run* $P_1(L, \mathbf{u}, (\mathbf{c}, \mathbf{c}'), w)$ *to obtain* st *and* π'.
2. *Invoke* $r \leftarrow \mathcal{O}_\mathsf{H}(\pi' \| \mathbf{u} \| \mathbf{c} \| \mathbf{c}', \text{desc}(L)) \in \mathbb{Z}_p$.
3. *Run* $P_2(L, (\mathbf{s}, \mathbf{t}), w, \text{st}, r)$ *to obtain* π''.

Return $\pi := (\pi', \pi'')$.

$\text{VERIFY}^{\mathcal{O}_\mathsf{H}}((\mathbf{crs}^{(j)}, L^{(j)}, \mathbf{u}^{(j)}, (\text{com}_i^{(j)})_{i\in[\ell_\mathsf{v}]})_{j\in[N]}, \pi)$. *Let language* L *and the statement* $\mathbf{u}$ *as well as commitment element vectors* $\mathbf{c}$ *and* $\mathbf{c}'$ *be defined as in* PROVE. *Parse* $\pi = (\pi', \pi'')$ *and run* $r \leftarrow \mathcal{O}_\mathsf{H}(\pi' \| \mathbf{u} \| \mathbf{c} \| \mathbf{c}', \text{desc}(L))$. *Return the output of*

$$V(L, \mathbf{u}, (\mathbf{c}, \mathbf{c}'), \pi', \pi'', r) \ .$$

Lemma 11. *The Groth-Sahai proof system using RO (Construction 3) fulfills all security properties stated in Definition 11 under the* $(\mathsf{DDH}, \frac{1}{2})$*-assumption.*

Proof. We prove each of the four properties:

Computational CRS Indistinguishability. The following can be shown (through a straightforward reduction to $(\mathsf{DDH}, \frac{1}{2})$ that takes as input $(\mathsf{g}_1, \mathsf{g}_2, \mathsf{g}_3)$ and forwards $(\mathsf{g}_1, \mathsf{g}_2, \mathsf{g}_3, z'\mathsf{g}_0)$ for some uniformly random $z' \leftarrow\!\!\$\ \mathbb{Z}_p$) for a suitable DDH adversary $\mathcal{B}$:

$$\left|\Pr\left[\mathcal{A}(\mathbf{crs}) \mid \mathbf{crs} \leftarrow \text{BGEN}(1^\lambda)\right] - \Pr\left[\mathcal{A}(x\mathsf{g}_0, y\mathsf{g}_0, z\mathsf{g}_0, z'\mathsf{g}_0) \mid x, y, z, z' \leftarrow\!\!\$\ \mathbb{Z}_p\right]\right| \leq \text{Adv}_\mathcal{B}^{\mathsf{DDH}}(\lambda) + 1/p \ .$$

Note that the above uses the fact that $z \leftarrow\!\!\$\ \mathbb{Z}_p \setminus \{xy\}$ (as sampled in BGEN) is statistically indistinguishable from $z \leftarrow\!\!\$\ \mathbb{Z}_p$ (as sampled by the DDH challenger).

$P_1(L_{(M_i,M_i')_{i\in[n]}}, \mathbf{u}, (\mathbf{c},\mathbf{c}'), \mathbf{w})$

sample $\mathbf{x} \leftarrow\!\$\ \mathbb{Z}_p^{\ell_\mathsf{w}}$

sample $y_i, y_i' \leftarrow\!\$\ \mathbb{Z}_p\ \forall i \in [n]$

$\mathsf{h}_i := y_i\mathbf{g}_0 + y_i'\mathbf{g}_1 + \left(0 \| \mathbf{x}^T\right) M_i'\mathbf{c}$

$\mathsf{h}_i' := y_i\mathbf{g}_2 + y_i'\mathbf{g}_3 + \left(0 \| \mathbf{x}^T\right) (M_i\mathbf{u} + M_i'\mathbf{c}')$

return $\mathrm{st} := (\mathbf{x}, (y_i, y_i')), \pi' := (\mathsf{h}_i, \mathsf{h}_i')_{i\in[n]}$

$P_2(L_{(M_i,M_i')_{i\in[n]}}, (\mathbf{s},\mathbf{t}), \mathbf{w}, \mathrm{st}, r)$

$\mathbf{a} := r \cdot \mathbf{w} + \mathbf{x} \quad \in \mathbb{Z}_p^{\ell_\mathsf{w}}$

$b_i := r \cdot \left(1 \| \mathbf{w}^T\right) M_i'\mathbf{s} - y_i \quad \in \mathbb{Z}_p$

$b_i' := r \cdot \left(1 \| \mathbf{w}^T\right) M_i'\mathbf{t} - y_i' \quad \in \mathbb{Z}_p$

return $\pi'' := (\mathbf{a}, (b_i, b_i')_{i\in[n]})$

$P_{\mathrm{sim}}(L_{(M_i,M_i')_{i\in[n]}}, \mathbf{u}, (\mathbf{c},\mathbf{c}'), r)$

sample $\mathbf{a} \leftarrow\!\$\ \mathbb{Z}_p^{\ell_\mathsf{w}}, b_i, b_i' \leftarrow\!\$\ \mathbb{Z}_p$

$\mathsf{h}_i := \left(r \| \mathbf{a}^T\right) M_i'\mathbf{c} - b_i\mathbf{g}_0 - b_i'\mathbf{g}_1$

$\mathsf{h}_i' := \left(r \| \mathbf{a}^T\right) (M_i\mathbf{u} + M_i'\mathbf{c}') - b_i\mathbf{g}_2 - b_i'\mathbf{g}_3$

return $\pi' := (\mathsf{h}_i, \mathsf{h}_i')_{i\in[n]}, \pi'' := (\mathbf{a}, (b_i, b_i')_{i\in[n]})$

$V(L_{(M_i,M_i')_{i\in[n]}}, \mathbf{u}, (\mathbf{c},\mathbf{c}'), \pi', \pi'', r)$

parse $\pi' = (\mathsf{h}_i, \mathsf{h}_i')_{i\in[n]}, \pi'' = (\mathbf{a}, (b_i, b_i')_{i\in[n]})$

$\mathsf{h}_i \stackrel{?}{=} \left(r \| \mathbf{a}^T\right) M_i'\mathbf{c} - b_i\mathbf{g}_0 - b_i'\mathbf{g}_1$

$\mathsf{h}_i' \stackrel{?}{=} \left(r \| \mathbf{a}^T\right) (M_i\mathbf{u} + M_i'\mathbf{c}') - b_i\mathbf{g}_2 - b_i'\mathbf{g}_3$

return 1 iff both true

Fig. 8: Subroutines used in proving and verifying statements in the language $L_{(M_i,M_i')_{i\in[n]}}$. This can be viewed as a Σ-protocol: P_1 computes a commitment π', and then P_2 computes a response to a given challenge r. In Construction 3, the Fiat-Shamir transform is applied to P_1 and P_2, with the challenge being generated as $\mathcal{O}_\mathsf{H}(\pi')$. The procedure P_{sim} allows to falsely prove that a statement is in $L_{(M_i,M_i')_{i\in[n]}}$, requiring the challenge to be known before computing commitment π'. This is necessarily inside OR-proofs. In all four procedures, the group elements $\mathbf{g}_0, \mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3$ are those from the **crs**.

$P_1(L_1 \wedge L_2, \mathbf{u}, (\mathbf{c},\mathbf{c}'), (w_1, w_2))$

$\mathrm{st}_1, \pi_1' \leftarrow P_1(L_1, \mathbf{u}_1, (\mathbf{c}_1, \mathbf{c}_1'), w_1)$

$\mathrm{st}_2, \pi_2' \leftarrow P_1(L_2, \mathbf{u}_2, (\mathbf{c}_2, \mathbf{c}_2'), w_2)$

return $\mathrm{st} := (\mathrm{st}_1, \mathrm{st}_2), \pi' := (\pi_1', \pi_2')$

$P_2(L_1 \wedge L_2, (\mathbf{s},\mathbf{t}), (w_1, w_2), (\mathrm{st}_1, \mathrm{st}_2), r)$

$\pi_1'' \leftarrow P_2(L_1, (\mathbf{s}_1, \mathbf{t}_1), w_1, \mathrm{st}_1, r)$

$\pi_2'' \leftarrow P_2(L_2, (\mathbf{s}_2, \mathbf{t}_2), w_2, \mathrm{st}_2, r)$

return $\pi'' := (\pi_1'', \pi_2'')$

$P_{\mathrm{sim}}(L_1 \wedge L_2, \mathbf{u}, (\mathbf{c},\mathbf{c}'), r)$

$(\pi_1', \pi_1'') \leftarrow P_{\mathrm{sim}}(L_1, \mathbf{u}_1, (\mathbf{c}_1, \mathbf{c}_1'), r)$

$(\pi_2', \pi_2'') \leftarrow P_{\mathrm{sim}}(L_2, \mathbf{u}_2, (\mathbf{c}_2, \mathbf{c}_2'), r)$

return $\pi' := (\pi_1', \pi_2'), \pi'' := (\pi_1'', \pi_2'')$

$V(L_1 \wedge L_2, \mathbf{u}, (\mathbf{c},\mathbf{c}'), \pi', (\pi_1'', \pi_2''), r)$

$V(L_1, \mathbf{u}_1, (\mathbf{c}_1, \mathbf{c}_1'), \pi_1', \pi_1'', r) \stackrel{?}{=} 1$

$V(L_2, \mathbf{u}_2, (\mathbf{c}_2, \mathbf{c}_2'), \pi_2', \pi_2'', r) \stackrel{?}{=} 1$

return 1 iff both true

Fig. 9: Subroutines used in proving and verifying statements in the AND-language $L_1 \wedge L_2$. See Figure 8 for details. In the above, we use $\mathbf{u}_1 \| \mathbf{u}_2 = \mathbf{u} \in \mathbb{G}^{\ell_{\mathsf{u}1}+\ell_{\mathsf{u}2}}$, $\mathbf{c}_1 \| \mathbf{c}_2 = \mathbf{c} \in \mathbb{G}^{\ell_{\mathsf{v}1}+\ell_{\mathsf{v}2}}$, $\mathbf{c}_1' \| \mathbf{c}_2' = \mathbf{c}' \in \mathbb{G}^{\ell_{\mathsf{v}1}+\ell_{\mathsf{v}2}}$, $\mathbf{s}_1 \| \mathbf{s}_2 = \mathbf{s} \in \mathbb{Z}_p^{\ell_{\mathsf{v}1}+\ell_{\mathsf{v}2}}$, $\mathbf{t}_1 \| \mathbf{t}_2 = \mathbf{t} \in \mathbb{Z}_p^{\ell_{\mathsf{v}1}+\ell_{\mathsf{v}2}}$, $\pi_1' \| \pi_2' = \pi'$.

$P_1(L_1 \vee L_2, \mathbf{u}, (\mathbf{c}, \mathbf{c}'), (b, w))$

$\mathrm{st}_b, \pi'_b \leftarrow P_1(L_b, \mathbf{u}_b, (\mathbf{c}_b, \mathbf{c}'_b), w)$
sample $r_{\overline{b}} \leftarrow\!\!\$\ \mathbb{Z}_p$
$(\pi'_{\overline{b}}, \pi''_{\overline{b}}) \leftarrow P_{\mathrm{sim}}(L_{\overline{b}}, \mathbf{u}_{\overline{b}}, (\mathbf{c}_{\overline{b}}, \mathbf{c}'_{\overline{b}}), r_{\overline{b}})$
return $\mathrm{st} := (\mathrm{st}_b, r_{\overline{b}}, \pi''_{\overline{b}}), \pi' := (\pi'_1, \pi'_2)$

$P_2(L_1 \vee L_2, (\mathbf{s}, \mathbf{t}), (b, w), (\mathrm{st}_b, r_{\overline{b}}, \pi''_{\overline{b}}), r)$

$r_b := r - r_{\overline{b}} \quad \in \mathbb{Z}_p$
$\pi''_b \leftarrow P_2(L_b, (\mathbf{s}_b, \mathbf{t}_b), w, \mathrm{st}_b, r_b)$
return $\pi'' := (r_1, \pi''_1, \pi''_2)$

$P_{\mathrm{sim}}(L_1 \vee L_2, \mathbf{u}, (\mathbf{c}, \mathbf{c}'), r)$

sample $r_1 \leftarrow\!\!\$\ \mathbb{Z}_p$
$r_2 := r - r_1 \quad \in \mathbb{Z}_p$
$(\pi'_1, \pi''_1) \leftarrow P_{\mathrm{sim}}(L_1, \mathbf{u}_1, (\mathbf{c}_1, \mathbf{c}'_1), r_1)$
$(\pi'_2, \pi''_2) \leftarrow P_{\mathrm{sim}}(L_2, \mathbf{u}_2, (\mathbf{c}_2, \mathbf{c}'_2), r_2)$
return $\pi' := (\pi'_1, \pi'_2), \pi'' := (r_1, \pi''_1, \pi''_2)$

$V(L_1 \vee L_2, \mathbf{u}, (\mathbf{c}, \mathbf{c}'), \pi', (r_1, \pi''_1, \pi''_2), r)$

$r_2 := r - r_1 \quad \in \mathbb{Z}_p$
$V(L_1, \mathbf{u}_1, (\mathbf{c}_1, \mathbf{c}'_1), \pi'_1, \pi''_1, r_1) \stackrel{?}{=} 1$
$V(L_2, \mathbf{u}_2, (\mathbf{c}_2, \mathbf{c}'_2), \pi'_2, \pi''_2, r_2) \stackrel{?}{=} 1$
return 1 iff both true

Fig. 10: Subroutines used in proving and verifying statements in the OR-language $L_1 \vee L_2$. We use the CDS trick [CDS94]. See Figures 8 and 9 for details. We use $\overline{b}$ to denote $3 - b$ (i.e., the index of the statement that is *not* necessarily valid.)

Analogously to the above, we also get

$$\left|\Pr\left[\mathcal{A}(\mathbf{crs}) \mid \mathbf{crs} \leftarrow \mathrm{HGen}(1^\lambda)\right] - \Pr\left[\mathcal{A}(x\mathsf{g}_0, y\mathsf{g}_0, z\mathsf{g}_0, z'\mathsf{g}_0) \mid x, y, z, z' \leftarrow\!\!\$\ \mathbb{Z}_p\right]\right| \leq \mathrm{Adv}^{\mathsf{DDH}}_{\mathcal{B}}(\lambda) + 1/p\ .$$

for a suitable DDH adversary $\mathcal{B}$. Therefore CRS indistinguishability follows by a hybrid argument.

Dual-Mode Commitment. If the **crs** is generated in binding mode, then any commitment $\mathrm{com} = (\mathsf{c}, \mathsf{c}')$ (regardless of the randomness used) for v fulfills $\mathsf{c}' - y\mathsf{c} = \mathsf{v}$, where y is the value sampled during BGen. Hence, the group element v is uniquely determined its commitment com, and two different elements cannot result in the same commitment.
Let $a \in \mathbb{Z}_p$ be the dlog of v, i.e., $\mathsf{v} = a\mathsf{g}_0$. If the **crs** is generated in binding mode, sampling $(s, t) \leftarrow\!\!\$\ \mathbb{Z}_p^2$ is identical to sampling $(s', t') \leftarrow\!\!\$\ \mathbb{Z}_p^2$ and then choosing $t = (t' - ys' - a)/(z - xy)$ and $s = s' - xt$ (due to $z \neq xy$). Plugging this into how the commitment is generated, we get $\mathsf{c} = s'\mathsf{g}_0$ and $\mathsf{c}' = t'\mathsf{g}_0$. Hence, for any fixed v and randomly sampled $R \leftarrow\!\!\$\ \mathcal{R}_{\mathsf{com}}$, the commitment has the same distribution as two uniformly random group elements, i.e., it is independent of the element v that is being committed to.

Statistical Soundness. We are going to show the following fact about the underlying proof system defined by P_1, P_2, and V: For any $\mathbf{crs} \leftarrow \mathrm{BGen}(1^\lambda)$, for any language $L \in \mathcal{L}$, for any statements $(\mathbf{u}, \mathbf{v}) \in \mathbb{G}^{\ell_\mathsf{u}} \times \mathbb{G}^{\ell_\mathsf{v}}$ that *do not* satisfy the language (i.e., $(\mathbf{u}, \mathbf{v}) \notin L$) and corresponding commitments $\mathrm{com}_i = (\mathsf{c}_i, \mathsf{c}'_i) \leftarrow \mathrm{Com}(\mathbf{crs}, \mathsf{v}_i; R_i)$ for arbitrary $R_i \in \mathcal{R}_{\mathsf{com}}$, and for any π', there exists at most 1 choice of $r \in \mathcal{R}_\mathsf{H}$ for which

$$\exists \pi'' \text{ with } V(L, \mathbf{u}, (\mathbf{c}, \mathbf{c}'), \pi', \pi'', r) = 1\ .$$

Once we have established the above, note that statistical soundness follows using a standard Fiat-Shamir transform argument in the following way: w.l.o.g., we may assume that for any output $(L^{(j)}, \mathbf{u}^{(j)}, (\mathrm{com}_i^{(j)}))_{j \in [N]}, (\pi', \pi'')$ of the adversary, it has previously made the query $\mathcal{O}_\mathsf{H}(\pi' \| \mathbf{u} \| \mathbf{c} \| \mathbf{c}', \mathrm{desc}(L))$ to the random oracle (with L, $\mathbf{u}$, $\mathbf{c}$, and $\mathbf{c}'$ defined as in Prove given the languages $L^{(j)}$, statements $\mathbf{u}^{(j)}$ and the commitments $\mathrm{com}_i^{(j)}$). Now, under the assumption that $\mathbf{crs}^{(j^*)}$ is an output of $\mathrm{BGen}(1^\lambda)$, we have $(\mathsf{g}_0, \mathsf{g}_1^{(j^*)}, \mathsf{g}_2^{(j^*)}, \mathsf{g}_4^{(j^*)}) \notin L_{\mathrm{ddh}}$. Furthermore, under the assumption that the adversary $\mathcal{A}$ succeeded, we have $(\mathbf{u}^{(j^*)}, \mathbf{v}^{(j^*)}) \notin L^{(j^*)}$ for the vector $\mathbf{v}^{(j^*)}$ of elements $\mathsf{v}_i^{(j^*)} := \mathsf{c}_i'^{(j^*)} - y^{(j^*)}\mathsf{c}_i^{(j^*)}$ (because $(\mathsf{c}_i^{(j^*)}, \mathsf{c}_i'^{(j^*)})$ is always a valid commitment of $\mathsf{v}_i^{(j^*)}$ given the

value $y^{(j^*)}$ generated by BGen and randomness $R_i^{(j^*)} := (s^{(j^*)}, 0)$ s.t. $\mathsf{c}_i^{(j^*)} = s_i^{(j^*)}\mathsf{g}_0$). Therefore,

$$(\mathbf{u}^{(j^*)} \| (\mathsf{g}_0, \mathsf{g}_1^{(j^*)}, \mathsf{g}_2^{(j^*)}, \mathsf{g}_4^{(j^*)}), \mathbf{v}^{(j^*)}) \notin L^{(j^*)} \vee L_{\mathrm{ddh}} \ .$$

By definition of language L (which is a conjunction of $L^{(j^*)} \vee L_{\mathrm{ddh}}$ and others) and by the previous fact, there exists at most 1 choice of $r \in \mathcal{R}_{\mathsf{H}}$ for which there is a π'' that makes Verify output 1. Hence, by randomness of $\mathcal{O}_{\mathsf{H}}$, the probability in the soundness definition is bounded by $T \cdot \frac{1}{|\mathcal{R}_{\mathsf{H}}|} \leq \mathsf{negl}(\lambda)$, with $T \leq \mathrm{poly}(\lambda)$ being a bound on the number of RO-calls by $\mathcal{A}$.

It remains to prove the fact stating uniqueness of $r \in \mathcal{R}_{\mathsf{H}}$. We will show that assuming the existence of two different $r, \tilde{r} \in \mathcal{R}_{\mathsf{H}}$ and corresponding proofs $\pi'', \tilde{\pi}''$, the statement $(\mathbf{u}, \mathbf{v})$ is contained in the language L (as this would immediately contradict the assumption that the statement $(\mathbf{u}, \mathbf{v})$ does not fulfill the definition of L).

We do so inductively for all languages according to their definition:

- For $L = L_{(M_i, M_i')_{i\in[n]}}$, suppose there exist two different challenges $r, \tilde{r} \in \mathbb{Z}_p$ and corresponding proofs $\pi'' = (\mathbf{a}, (b_i, b_i')_{i\in[n]})$ and $\tilde{\pi}'' = (\tilde{\mathbf{a}}, (\tilde{b}_i, \tilde{b}_i')_{i\in[n]})$ s.t. the verifier V would output 1. Consider now the values that $V(L_{(M_i,M_i')_{i\in[n]}}, \mathbf{u}, (\mathbf{c}, \mathbf{c}'), \pi', \cdot, \cdot)$ forces upon h_i and h_i' in the two cases of r, π'' and $\tilde{r}, \tilde{\pi}''$. Setting equal the value $\mathsf{h}_i' - y\mathsf{h}_i$ (with y as defined by BGen) that we receive in the two respective cases, we get the identity

$$\left(r \| \mathbf{a}^T\right)\left(M_i\mathbf{u} + M_i'\mathbf{c}' - yM_i'\mathbf{c}\right) = \left(\tilde{r} \| \tilde{\mathbf{a}}^T\right)\left(M_i\mathbf{u} + M_i'\mathbf{c}' - yM_i'\mathbf{c}\right) \ ,$$

  and due to $M'\mathbf{c}' - yM_i'\mathbf{c} = M'\mathbf{v}$, the vector $\frac{1}{r-\tilde{r}} \cdot (\mathbf{a} - \tilde{\mathbf{a}})$ is a witness proving $(\mathbf{u}, \mathbf{v}) \in L$.
- For $L = L_1 \wedge L_2$, suppose there exist two different challenges $r, \tilde{r} \in \mathbb{Z}_p$ and corresponding proofs (π_1'', π_2'') and $(\tilde{\pi}_1'', \tilde{\pi}_2'')$. Then, by induction, for both $b \in \{1, 2\}$ we get $(\mathbf{u}_b, \mathbf{v}_b) \in L_b$, and therefore also $(\mathbf{u}, \mathbf{v}) \in L$.
- For $L = L_1 \vee L_2$, suppose there exist two different challenges $r, \tilde{r} \in \mathbb{Z}_p$ and corresponding proofs (r_1, π_1'', π_2'') and $(\tilde{r}_1, \tilde{\pi}_1'', \tilde{\pi}_2'')$. Define $r_2 := r - r_1$ and $\tilde{r}_2 := \tilde{r} - \tilde{r}_1$. Due to $r \neq \tilde{r}$, at least one of $r_1 \neq \tilde{r}_1$ and $r_2 \neq \tilde{r}_2$ must hold, and therefore we get $(\mathbf{u}_b, \mathbf{v}_b) \in L_b$ for at least one $b \in \{1, 2\}$, implying $(\mathbf{u}, \mathbf{v}) \in L$.

Perfect Zero-Knowledge. We first need to show that for any $(\mathbf{crs}, \mathbf{trap}) \leftarrow \mathrm{HGen}(1^\lambda)$, for any language L, and for any statements $(\mathbf{u}, \mathbf{v}) \in L$ with corresponding commitments $(\mathsf{c}_i, \mathsf{c}_i') = \mathrm{Com}(\mathbf{crs}, \mathsf{v}_i; (s_i, t_i))$ and witnesses $w \in W_L(\mathbf{u}, \mathbf{v})$, the following two distributions are identical *for any fixed* $r \in \mathbb{Z}_p$:

$$\left\{ (\pi', \pi'') \;\middle|\; \begin{array}{c} \mathrm{st}, \pi' \leftarrow P_1(L, \mathbf{u}, (\mathbf{c}, \mathbf{c}'), w) \\ \pi'' \leftarrow P_2(L, (\mathbf{s}, \mathbf{t}), w, \mathrm{st}, r) \end{array} \right\} \equiv \{(\pi', \pi'') \leftarrow P_{\mathrm{sim}}(L, \mathbf{u}, (\mathbf{c}, \mathbf{c}'), r)\} \ ,$$

i.e., for any fixed r, simulated proofs and normally generated proofs are identically distributed. The above can be easily seen by our recursive definition of LCDs:

- For $L = L_{(M_i,M_i')_{i\in[n]}}$, the marginal distribution of $\pi'' = (\mathbf{a}, (b_i, b_i')_{i\in[n]})$ is clearly uniformly random in $\mathbb{Z}_p^{\ell_{\mathsf{w}}} \times (\mathbb{Z}_p^2)^n$. Furthermore, the distribution of π' conditioned on π'' is identical in both cases: This is because there exists only a single π' for which V returns 1 and in both distributions a π' is generated on which V returns 1 (due to the correctness property).
- For $L = L_1 \wedge L_2$, the proof $\pi' = (\pi_1', \pi_2'), \pi'' = (\pi_1'', \pi_2'')$ consists of two independently generated proofs π_1', π_1'' and π_2', π_2'', both of which are perfectly simulated by induction.
- For $L = L_1 \vee L_2$, note that r_1 is in both cases uniformly random in $\mathbb{Z}_p$. In addition, for any r_b, the proof π_b', π_b'' is perfectly simulated by induction, and the two proofs for $b = 1$ and $b = 2$ are independently generated of each other.

Now consider the distribution as defined in the definition of the perfect zero-knowledge property. That is, we are given a list of $\mathbf{crs}^{(j)}$, languages $L^{(j)}$, statements $\mathbf{u}^{(j)}$, commitments $\mathrm{com}_i^{(j)}$, and witnesses $w^{(j)}$. Let the combined language L, combined statement $\mathbf{u}$ as well as combined commitment element vectors $\mathbf{c}$ and $\mathbf{c}'$ and combined witness w be defined as in Prove. Since we have shown that proofs can be perfectly simulated for any fixed r, we immediately get witness-

indistinguishability, i.e., for any fixed RO $\mathcal{O}_{\mathsf{H}}$, the distribution

$$\left\{ (\pi', \pi'') \;\middle|\; \begin{array}{l} \mathrm{st}, \pi' \leftarrow P_1(L, \mathbf{u}, (\mathbf{c}, \mathbf{c}'), w) \\ \quad r \leftarrow \mathcal{O}_{\mathsf{H}}(\pi' \| \mathbf{u} \| \mathbf{c} \| \mathbf{c}', \mathrm{desc}(L)) \\ \quad \pi'' \leftarrow P_2(L, (\mathbf{s}, \mathbf{t}), w, \mathrm{st}, r) \end{array} \right\}$$

is independent of the witness $w \in W_L(\mathbf{u}, \mathbf{v})$, even when the entire function table of $\mathcal{O}_{\mathsf{H}}$ is public. Note that w has the form $w = (\widetilde{w}^{(1)}, \ldots, \widetilde{w}^{(N)})$, i.e., it contains one witness $\widetilde{w}^{(j)}$ for each of the languages $L^{(j)} \vee L_{\mathrm{ddh}}$. As required by perfect zero-knowledge, *any* witness $w^{(j)} \in W_{L^{(j)}}(\mathbf{u}^{(j)}, \mathbf{v}^{(j)})$ will yield a valid witness $w \in W_L(\mathbf{u}, \mathbf{v})$ (and therefore also result in the same proof distribution of (π', π'') for the combined language L as shown above), by selecting $\widetilde{w}^{(j)} := (1, w^{(j)})$ as a witness for $W_{L^{(j)} \vee L_{\mathrm{ddh}}}$. Furthermore, when $\mathbf{crs}^{(j)}$ is in hiding mode, $w^{(j)} = \mathbf{trap}^{(j)}$ also results in a valid witness w, because we can select $\widetilde{w}^{(j)} := (2, \mathbf{trap}^{(j)})$ as a witness for $W_{L^{(j)} \vee L_{\mathrm{ddh}}}$.

□

Remark 1. There are two potential optimizations to the proof system in Construction 3 that are applicable whenever some weaker primitive is sufficient.

- First, the proof (π', π'') produced by P_1 and P_2 for a language of the form $L_{(M_i, M_i')_{i \in [n]}}$ can be significantly simplified *whenever no commitments are present*, i.e., whenever $\ell_{\mathsf{v}} = 0$. The first part π' will need to contain only a single group element h_i' (instead of two) per equation:

  $$\mathsf{h}_i' := \left(0 \| \mathbf{x}^T\right) M_i \mathbf{u} \;,$$

  and the second part π'' only needs to contain the vector $\mathbf{a} = r \cdot \mathbf{w} + \mathbf{x}$.
  Note that this optimization even applies to the language L_{ddh} that is heavily used inside Prove and Verify.
- Second, if only *perfect witness-indistinguishability* instead of *perfect zero-knowledge* is required, then it is unnecessary to include a trapdoor. That is, the **crs** only requires 3 instead of 4 group elements, and the combined language L as defined in Prove simplifies to $L := L^{(1)} \wedge \cdots \wedge L^{(N)}$ (i.e., none of the individual languages $L^{(j)}$ needs to be masked with an additional "$\vee\, L_{\mathrm{ddh}}$").