# On the (In)security of SNARKs in the Presence of Oracles

Dario Fiore[1] and Anca Nitulescu[1,2]

[1] IMDEA Software Institute, Madrid, Spain
{dario.fiore}@imdea.org
[2] ENS, CNRS, INRIA, and PSL, Paris, France
Anca.Nitulescu@ens.fr

**Abstract.** In this work we study the feasibility of knowledge extraction for succinct non-interactive arguments of knowledge (SNARKs) in a scenario that, to the best of our knowledge, has not been analyzed before. While prior work focuses on the case of adversarial provers that may receive (statically generated) *auxiliary information*, here we consider the scenario where adversarial provers are given *access to an oracle*. For this setting we study if and under what assumptions such provers can admit an extractor. Our contribution is mainly threefold.

First, we formalize the question of extraction in the presence of oracles by proposing a suitable proof of knowledge definition for this setting. We call SNARKs satisfying this definition O-SNARKs.

Second, we show how to use O-SNARKs to obtain formal and intuitive security proofs for three applications (homomorphic signatures, succinct functional signatures, and SNARKs on authenticated data) where we recognize an issue while doing the proof under the standard proof of knowledge definition of SNARKs.

Third – and this is the most prominent part of our work – we study whether O-SNARKs exist, providing both negative and positive results. On the negative side, we show that, assuming collision-resistant hash functions, there do not exist O-SNARKs in the standard model for every oracle family. Next, we study the interesting case of *signing* oracles. We give a negative result showing that even O-SNARKs for every signing oracle family do not exist. On the positive side, instead, we show that, when considering signature schemes with appropriate restrictions on the message length, O-SNARKs for the corresponding signing oracles exist, based on classical SNARKs and assuming extraction with respect to specific distributions of auxiliary input.

# Table of Contents


On the (In)security of SNARKs in the Presence of Oracles . . . . . 1
*Dario Fiore and Anca Nitulescu*
1 Introduction . . . . . 3
1.1 Extraction in the Presence of Oracles . . . . . 4
1.2 An Overview of Our Results . . . . . 5
2 Preliminaries . . . . . 8
2.1 Succinct Non-Interactive Arguments . . . . . 8
3 SNARKs in the presence of oracles . . . . . 11
3.1 O-SNARKs: SNARKs in the presence of oracles. . . . . 11
3.2 Non-Adaptive O-SNARKs . . . . . 12
4 On the existence of O-SNARKs . . . . . 14
4.1 O-SNARKs in the random oracle model from Micali's CS proofs . . . . . 14
4.2 Impossibility of O-SNARKs for every family of oracles, in the standard model . . . . . 14
4.3 Impossibility of O-SNARKs for every family of signing oracles . . . . . 16
4.4 O-SNARKs for signing oracles from SNARKs in the random oracle model . . . . . 20
4.5 O-SNARKs for signing oracles from SNARKs . . . . . 22
4.6 O-SNARKs for (pseudo)random oracles from SNARKs . . . . . 24
5 Applications of O-SNARKs . . . . . 26
5.1 Homomorphic Signatures . . . . . 27
5.2 Succinct Functional Signatures . . . . . 34
5.3 SNARKs on authenticated data . . . . . 43
A Additional Preliminaries . . . . . 45
A.1 Digital Signatures . . . . . 45
B The SNARK definition used in [BGI14] . . . . . 46

## 1 Introduction

**Succinct Arguments.** Proof systems [GMR89] are fundamental in theoretical computer science and cryptography. Extensively studied aspects of proof systems are the expressivity of provable statements and the efficiency. Related to efficiency, it has been shown that statistically-sound proof systems are unlikely to allow for significant improvements in communication [BHZ87, GH98, GVW02, Wee05]. When considering proof systems for $\mathsf{NP}$ this means that, unless some complexity-theoretic collapses occur, in a statistically sound proof system any prover has to communicate, roughly, as much information as the size of the $\mathsf{NP}$ witness. The search of ways to beat this bound motivated the study of *computationally-sound* proof systems, also called *argument systems* [BCC88]. Assuming existence of collision-resistant hash functions, Kilian [Kil92] showed a four-message interactive argument for $\mathsf{NP}$. In this protocol, membership of an instance $x$ in an $\mathsf{NP}$ language with $\mathsf{NP}$ machine $M$ can be proven with communication and verifier's running time bounded by $p(\lambda, |M|, |x|, \log t)$, where $\lambda$ is a security parameter, $t$ is the $\mathsf{NP}$ verification time of machine $M$ for the instance $x$, and $p$ is a *universal* polynomial. Argument systems of this kind are called *succinct*.

**Succinct Non-Interactive Arguments.** Starting from Kilian's protocol, Micali [Mic94] constructed a *one-message* succinct argument for $\mathsf{NP}$ whose soundness is set in the random oracle model. The fact that one-message succinct arguments are unlikely to exist for hard-enough languages in the plain model motivated the consideration of *two-message* non-interactive arguments, in which the verifier generates its message (a common reference string, if this can be made publicly available) ahead of time and independently of the statement to be proved. Such systems are called *succinct non-interactive arguments* (SNARGs) [GW11]. The area of SNARGs became quite popular in the last years with the proposal of several constructions in the standard model, some of which gained significant improvements in efficiency [BCCT12, BCC$^+$14, GGPR13, PHGR13, BSCG$^+$13, BCTV14]. Noteworthy is that all such constructions are based on non-falsifiable assumptions [Nao03], a class of assumptions that is likely to be inherent in proving the security of SNARGs (without random oracles), as shown by Gentry and Wichs [GW11].

**Proof of Knowledge.** SNARGs have been also strengthened to become *succinct non-interactive arguments of knowledge* (SNARKs) [BCCT12, BCC$^+$14]. SNARKs are SNARGs where computational soundness is replaced by *proof of knowledge*. Intuitively speaking, this property says that every prover producing a convincing proof must "know" a witness. On the one hand, proof of knowledge turns out to be useful in many applications, such as delegation of computation where the untrusted worker contributes its own input to the computation, or recursive proof composition [Val08, BCCT13]. On the other hand, the formalization of proof of knowledge in SNARKs is a delicate point. Typically, the concept that the prover "must know" a witness is expressed by assuming that such knowledge can be efficiently extracted from the prover by means of a so-called *knowledge extractor*. In SNARKs extractors are inherently non-black-box, and proof of knowledge requires that for every adversarial prover $\mathcal{A}$ generating an accepting proof $\pi$ there must be an extractor $\mathcal{E}_{\mathcal{A}}$ that, given the same input of $\mathcal{A}$, outputs a valid witness.

**Extraction with Auxiliary Input.** Unfortunately, stated as above, proof of knowledge is insufficient for being used in many applications. The problem is that, when using SNARKs in larger cryptographic protocols, adversarial provers may get additional information which can contribute to the generation of adversarial proofs. To address this problem, a stronger, and more useful, definition of proof of knowledge requires that for any adversary $\mathcal{A}$ there is an extractor $\mathcal{E}_{\mathcal{A}}$ such that,

for any honestly generated $\mathsf{crs}$ and any polynomial-size auxiliary input $aux$, whenever $\mathcal{A}(\mathsf{crs}, aux)$ returns an accepting proof, $\mathcal{E}_{\mathcal{A}}(\mathsf{crs}, aux)$ outputs a valid witness.

This type of definition is certainly more useful for using SNARKs in larger cryptographic protocols, but it also introduces other subtleties. As first discussed in [HT98], extraction in the presence of arbitrary auxiliary input can be problematic, if not implausible. Formal evidence of this issue has been recently given in [BCPR14, BP15]. Bitansky et al. [BCPR14] show that, assuming indistinguishability obfuscation, there do not exist extractable one-way functions (and thus SNARKs) with respect to arbitrary auxiliary input of unbounded polynomial length. Boyle and Pass [BP15] generalize this result showing that assuming collision-resistant hash functions and differing-input obfuscation, there is a fixed auxiliary input distribution for which extractable one-way functions do not exist.

### 1.1 Extraction in the Presence of Oracles

In this work we continue the study on the feasibility of extraction by looking at a scenario that, to the best of our knowledge, has not been explicitly analyzed before. We consider the case in which adversarial provers run in interactive security experiments where they are given *access to an oracle*. For this setting we study if and under what assumptions such provers can admit an extractor.

Before giving more detail on our results, let us discuss a motivation for analyzing this scenario. To keep the presentation simple, here we give a motivation via a hypotetical example; more concrete applications are discussed later.

A CASE STUDY APPLICATION. Consider an application where Alice gets a collection of signatures generated by Bob, and she has to prove to a third party that she owns a valid signature of Bob on some message $m$ such that $P(m) = 1$. Let us say that this application is secure if Alice, after asking for signatures on several messages, cannot cheat letting the third party accept for a false statement (i.e., $P(m) = 0$, or $P(m) = 1$ but Alice did not receive a signature on $m$). If messages are large and one wants to optimize bandwidth, SNARKs can be a perfect candidate solution for doing such proofs,[3] i.e., Alice can generate a proof of knowledge of $(m, \sigma)$ such that "$(m, \sigma)$ verifies with Bob's public key and $P(m) = 1$".

AN ATTEMPT OF SECURITY PROOF. Intuitively, the security of this protocol should follow easily from the proof of knowledge of the SNARK and the unforgeability of the signature scheme. However, somewhat surprisingly, the proof becomes quite subtle. Let us consider a cheating Alice that always outputs a proof for a statement in the language.[4] If Alice is still cheating, then it must be that she is using a signature on a message that she did not query – in other words a forgery. Then one would like to reduce such a cheating Alice to a forger for the signature scheme. To do this, one would proceed as follows. For any Alice one defines a forger that, on input the verification key $\mathsf{vk}$, generates the SNARK $\mathsf{crs}$, gives $(\mathsf{crs}, \mathsf{vk})$ to Alice, and simulate's Alice's queries using its own signing oracle. When Alice comes with the cheating proof, the forger would need an extractor for Alice in order to obtain the forgery from her. However, even if we see Alice as a SNARK prover with auxiliary input $\mathsf{vk}$, Alice does not quite fit the proof of knowledge definition in which adversaries have no oracles. To handle similar cases, one typically shows that for every, interactive, Alice there is a non-interactive algorithm $\mathcal{B}$ that runs Alice simulating her oracles (i.e., $\mathcal{B}$ samples the signing key) and returns the same output. The good news is that for such $\mathcal{B}$ one can claim the existence

[3] Further motivation can also be to maintain the privacy of $m$ by relying on zero-knowledge SNARKs.
[4] The other case of statements not in the language can be easily reduced to the soundness of the SNARK.

of an extractor $\mathcal{E}_{\mathcal{B}}$ as it fits the proof of knowledge definition. The issue is though that $\mathcal{E}_{\mathcal{B}}$ expects the same input of $\mathcal{B}$, which includes the secret signing key. This means that our candidate forger mentioned above (which does not have the secret signing key) cannot run this extractor.

Applications that need extraction with oracles. Besides the above example, this issue can show up essentially in every application of SNARKs in which adversaries have access to oracles with a secret state, and one needs to run an extractor during an experiment (e.g., a reduction) where the secret state of the oracle is not available. For instance, we recognize this issue while trying to formally prove the security of a "folklore" construction of homomorphic signatures based on SNARKs and digital signatures that is mentioned in several papers (e.g., [BF11, GW13, CF13, GVW15]). The same issue appears in a generic construction of SNARKs on authenticated data in [BBFR15] (also informally discussed in [BCCT12]), where the security proof uses the existence of an extractor for the oracle-aided prover, but without giving particular justification. A similar issue also appears in the construction of succinct functional signatures of [BGI14]. To be precise, in [BGI14] the authors provide a (valid) proof but under a stronger definition of SNARKs in which the adversarial prover and the extractor are independent PPT machines without *common* auxiliary input: a notion for which we are not aware of standard model constructions. In contrast, if one attempts to prove the succinct functional signatures of [BGI14] using the standard definition of SNARKs, one incurs the same issues illustrated above, i.e., the proof would not go through.

In this work we address this problem by providing both negative and positive results to the feasibility of extraction in the presence of oracles. On one hand, our negative results provide an explanation of why the above proofs do not go through so easily. On the other hand, our positive results eventually provide some guidelines to formally state and prove the security of the cryptographic constructions mentioned above (albeit with various restrictions).

### 1.2 An Overview of Our Results

**Defining SNARKs in the presence of oracles.** As a first step, we formalize the definition of non-black-box extraction in the presence of oracles by proposing a notion of *SNARKs in the presence of oracles* (O-SNARKs, for short). In a nutshell, an O-SNARK is like a SNARK except that adaptive proof of knowledge must hold with respect to adversaries that have access to an oracle $\mathcal{O}$ sampled from some oracle family $\mathbb{O}$.[5] Slightly more in detail, we require that for any adversary $\mathcal{A}^{\mathcal{O}}$ with access to $\mathcal{O}$ there is an extractor $\mathcal{E}_{\mathcal{A}}$ such that, whenever $\mathcal{A}^{\mathcal{O}}$ outputs a valid proof, $\mathcal{E}_{\mathcal{A}}$ outputs a valid witness, by running on the same input of $\mathcal{A}$, plus the transcript of oracle queries-answers of $\mathcal{A}$.

**Existence of O-SNARKs.** Once having defined their notion, we study whether O-SNARKs exist and under what assumptions. Our results are summarized in the following paragraphs.

O-SNARKs in the random oracle model. As a first positive result, we show that the construction of Computationally Sounds (CS) proofs of Micali [Mic00] yields an O-SNARK for *every* oracle family, in the random oracle model. This result follows from the work of Valiant [Val08] which shows that Micali's construction already allows for extraction. More precisely, using the power of the random oracle model, Valiant shows a *black-box* extractor. This powerful extractor can then be used to build an O-SNARK extractor that works for any oracle family.

[5] In fact the notion is parametrized by the family $\mathbb{O}$, i.e., we say $\Pi$ is an O-SNARK for $\mathbb{O}$.

Insecurity of O-SNARKs for every oracle family, in the standard model. Although the above result gives a candidate O-SNARK, it only works in the random oracle model, and it is tailored to one construction [Mic00]. It is therefore interesting to understand whether extraction with oracles is feasible in the *standard model*. And it would also be interesting to see if this is possible based on the classical SNARK notion. Besides its theoretical interest, the latter question has also a practical motivation since there are several efficient SNARK constructions proposed in the last years that one might like to use in place of CS proofs. Our first result in this direction is that assuming existence of collision-resistance hash functions (CRHFs) there do not exist O-SNARKs for $\mathsf{NP}$ with respect to *every* oracle family. More precisely, we show the following:

**Theorem 1 (Informal).** *Assume the existence of CRHFs. Then for any polynomial $p(\cdot)$ there is a family of oracles $\mathbb{O}_p$ such that any candidate O-SNARK for $\mathsf{NP}$, that is correct and succinct with proofs of length bounded by $p(\cdot)$, cannot satisfy adaptive proof of knowledge with respect to oracles from $\mathbb{O}_p$.*

A basic intuition behind our result is that oracles provide additional auxiliary input to adversaries and, as formerly shown in [BCPR14, BP15], this can create issues for extraction. In fact, to obtain our result we might also have designed an oracle that simply outputs a binary string following a distribution with respect to which extraction is impossible due to [BCPR14, BP15]. However, in this case the result should additionally assume the existence of indistinguishability (or differing-input) obfuscation. In contrast, our result shows that such impossibility holds by only assuming existence of CRHFs, which is a much weaker assumption.

Insecurity of O-SNARKs for every family of signing oracles. The impossibility result of Theorem 1 is very general and, in some sense, not surprising as it considers any possible oracle family. Therefore we further study the question of O-SNARKs existence for more specific oracle families. In particular, motivated by the three applications mentioned earlier,[6] we study the existence of O-SNARKs with respect to *signing oracles*, and prove the following impossibility result:

**Theorem 2 (Informal).** *Assume unforgeable signatures exist. Then for any polynomial $p(\cdot)$ there is an unforgeable signature scheme $\Sigma_p$ such that any candidate O-SNARK, that is correct and succinct with proofs of length bounded by $p(\cdot)$, cannot satisfy adaptive proof of knowledge with respect to signing oracles corresponding to $\Sigma_p$.*

This result can be seen as a strengthening of Theorem 1 as it also gives us an impossibility of O-SNARKs for general oracles that can be based on weaker assumptions (OWFs vs. CRHFs). In addition, it tells us that one cannot assume existence of O-SNARKs that work for *any* signature scheme, which shows why the security proofs of the primitives considered earlier do not go through.

Existence of O-SNARKs for specific families of signing oracles. We study ways to circumvent our impossibility result for signing oracles of Theorem 2. Indeed, the above result can be interpreted as saying that there exist (perhaps degenerate) signature schemes such that there are no O-SNARKs with respect to the corresponding signing oracle family. This is not ruling out that O-SNARKs may exist for specific signature schemes, or – even better – for specific classes of signature schemes. We provide the following results:

[6] We do believe that many more applications along the same line – proving knowledge of valid signatures – are conceivable.

1. We show that hash-and-sign signatures, where the hash function is a random oracle, yield "safe oracles", i.e., oracles for which any SNARK is an O-SNARK for that oracle, in the ROM.

2. We turn our attention to the standard model setting. We show that any classical SNARK is an O-SNARK for signing oracles if the message space of the signature scheme is properly bounded, and O-SNARK adversaries query "almost" the entire message space. More precisely, we show the following:

**Theorem 3 (Informal).** *Let $\Sigma$ be a signature scheme with message space $\mathcal{M}$ where $|\mathcal{M}| = \mathsf{poly}(\lambda)$ (resp. $|\mathcal{M}| = \lambda^{\omega(1)}$), and let $Q = |\mathcal{M}| - c$ for a constant $c \in \mathbb{N}$. Then a classical SNARK that is polynomially (resp. sub-exponentially) secure is also an O-SNARK for the family of signing oracles corresponding to $\Sigma$, for adversaries that make $|\mathcal{M}| - c$ signing queries.*

This positive existence result is useful in applications where one uses SNARKs with signing oracles, under the condition that adversaries make signing queries on almost the entire message space.

Non-Adaptive O-SNARKs. Finally, we consider a relaxed notion of O-SNARKs in which adversaries are required to declare in advance (i.e., before seeing the common reference string) all the oracle queries. For this weaker notion we show that, in the standard model, every SNARK (supporting arbitrary auxiliary inputs) is a non-adaptive O-SNARK.

**Applications of O-SNARKs.** A nice feature of the O-SNARK notion is that it lends itself to easy and intuitive security proofs in all those applications where one needs to execute extractors in interactive security games with oracles. We show that by replacing SNARKs with O-SNARKs (for appropriate oracle families) we can formally prove the security of the constructions of homomorphic signatures, succinct functional signatures and SNARKs on authenticated data that we mentioned in the previous section. By combining these O-SNARK-based constructions with our existence results mentioned earlier we eventually reach conclusions about the possible secure instantiations of these constructions.

First, one can instantiate them by using Micali's CS proofs as an O-SNARK: this solution essentially yields secure instantiations in the random oracle model that work with a specific proof system [Mic00] (perhaps not the most efficient one in practice). The second option is to instantiate them using hash-and-sign signatures, apply our result on hash-and-sign signatures mentioned above, and then conjecture that replacing the random oracle with a suitable hash function preserves the overall security.[7] Third, one can instantiate the constructions using a classical SNARK scheme $\Pi$ and signature scheme $\Sigma$, and then conjecture that $\Pi$ is also an O-SNARK with respect to the family of signing oracles corresponding to $\Sigma$. Compared to the first solution, the last two ones have the advantage that one could use some of the recently proposed efficient SNARKs (e.g., [PHGR13, BSCG+13]); on the other hand, these solutions have the drawback that security is based only on a heuristic argument. Finally, as a fourth option we provide security proofs of these primitives under a weak, non-adaptive, notion where adversaries declare all their queries in advance. Security in this weaker model can be proven assuming *non-adaptive O-SNARKs*, and thus classical SNARKs. The advantage of this fourth option is that one obtains a security proof for these instantiations based on clear – not newly crafted – assumptions, although under a much weaker security notion.

Finally, worth noting is that we cannot apply the positive result of Theorem 3 to the O-SNARK-based constructions of homomorphic signatures, functional signatures and SNARKs on authenti-

[7] The need of this final heuristic step is that hash-and-sign signatures use a random oracle in verification and in our applications the SNARK is used to prove knowledge of valid signatures, i.e., one would need a SNARK for $\mathsf{NP}^{\mathcal{O}}$.

cated data that we provide, and thus conclude their security under classical SNARKs. The inapplicability is due to the restriction of Theorem 3, for which adversaries have to query almost the entire message space.[8]

**Interpretation of our results.** In line with recent work [BCPR14, BP15] on the feasibility of extraction in the presence of auxiliary input, our results indicate that additional care must be taken when considering extraction *in the presence of oracles*. While for auxiliary input impossibility of extraction is known under obfuscation-related assumptions, in the case of oracles we show that extraction becomes impossible even by only assuming one-way functions. Furthermore, our work establishes a framework that eases the analysis and the use of SNARK extractors in all those security experiments where these are given access to an oracle. Finally, we remark that our counterexamples are of artificial nature and do not give evidence of extraction's impossibility in the presence of "natural" oracles. In this sense, they can be seen of theoretical interest. Yet, given the importance of provable security and considered the increasing popularity of SNARKs in more practical scenarios, we believe these results give a useful message to protocol designers.

## 2 Preliminaries

In this section, we review the notation and some basic definitionsthat we use in our work.

**Notation.** We denote with $\lambda \in \mathbb{N}$ the security parameter. We say that a function $\epsilon(\lambda)$ is *negligible* if it vanishes faster than the inverse of any polynomial in $\lambda$. If not explicitly specified otherwise, negligible functions are negligible with respect to $\lambda$. If $S$ is a set, $x \xleftarrow{\$} S$ denotes the process of selecting $x$ uniformly at random in $S$. If $\mathcal{A}$ is a probabilistic algorithm, $x \xleftarrow{\$} \mathcal{A}(\cdot)$ denotes the process of running $\mathcal{A}$ on some appropriate input and assigning its output to $x$. For binary strings $x$ and $y$, we denote by $x|y$ their concatenation and by $x_i$ the $i$-th bit of $x$. For a positive integer $n$, we denote by $[n]$ the set $\{1, \ldots, n\}$. For a random-access machine $M$ we denote by $\#M(x, w)$ the number of running steps needed by $M$ to accept on input $(x, w)$.

**The Universal Relation and NP Relations.** We recall the notion of universal relation from [BG08], here adapted to the case of non-deterministic computations.

**Definition 1.** *The* universal relation *is the set $\mathcal{R}_\mathcal{U}$ of instance-witness pairs $(y, w) = ((M, x, t), w)$, where $|y|, |w| \leq t$ and $M$ is a random-access machine such that $M(x, w)$ accepts after running at most $t$ steps. The* universal language *$\mathcal{L}_\mathcal{U}$ is the language corresponding to $\mathcal{R}_\mathcal{U}$.*

For any constant $c \in \mathbb{N}$, $\mathcal{R}_c$ denotes the subset of $\mathcal{R}_\mathcal{U}$ of pairs $(y, w) = ((M, x, t), w)$ such that $t \leq |x|^c$. $\mathcal{R}_c$ is a "generalized" NP relation that is decidable in some fixed time polynomial in size of the instance.

### 2.1 Succinct Non-Interactive Arguments

In this section we provide formal definitions for the notion of succinct non-interactive arguments of knowledge (SNARKs).

[8] The exact reason is rather technical and requires to see the precise definitions and constructions of these primitives first. For the familiar reader, the intuition is that in these primitives/constructions an adversary that queries almost the entire message space of the underlying signature scheme becomes able to break the security.

**Definition 2 (SNARGs).** *A* succinct non-interactive argument *(SNARG) for a relation $\mathcal{R} \subseteq \mathcal{R}_{\mathcal{U}}$ is a triple of algorithms $\Pi = (\mathsf{Gen}, \mathsf{Prove}, \mathsf{Ver})$ working as follows*

$\mathsf{Gen}(1^\lambda, T) \to \mathsf{crs}$**:** *on input a security parameter $\lambda \in \mathbb{N}$ and a time bound $T \in \mathbb{N}$, the generation algorithm outputs a common reference string $\mathsf{crs} = (\mathsf{prs}, \mathsf{vst})$ consisting of a public prover reference string $\mathsf{prs}$ and a verification state $\mathsf{vst}$.*

$\mathsf{Prove}(\mathsf{prs}, y, w) \to \pi$**:** *given a prover reference string $\mathsf{prs}$, an instance $y = (M, x, t)$ with $t \leq T$ and a valid witness $w$, i.e., $(y, w) \in \mathcal{R}$, the prover algorithm produces a proof $\pi$.*

$\mathsf{Ver}(\mathsf{vst}, y, \pi) \to b$**:** *on input a verification state $\mathsf{vst}$, an instance $y$, and a proof $\pi$, the verifier algorithm outputs $b = 0$ (reject) or $b = 1$ (accept).*

*and satisfying* completeness, succinctness, *and* (adaptive) soundness*as described below:*

- ***Completeness.*** *For every time bound $T \in \mathbb{N}$, every valid $(y, w) \in \mathcal{R}$ with $y = (M, x, t)$ and $t \leq T$, there exists a negligible function $\mathsf{negl}$ such that*

$$\Pr\left[\mathsf{Ver}(\mathsf{vst}, y, \pi) = 0 \;\middle|\; \begin{array}{c}(\mathsf{prs}, \mathsf{vst}) \leftarrow \mathsf{Gen}(1^\lambda, T) \\ \pi \leftarrow \mathsf{Prove}(\mathsf{prs}, y, w)\end{array}\right] \leq \mathsf{negl}(\lambda)$$

- ***Succinctness.*** *There exists a fixed polynomial $p(\cdot)$ independent of $\mathcal{R}$ such that for every large enough security parameter $\lambda \in \mathbb{N}$, every time bound $T \in \mathbb{N}$, and every instance $y = (M, x, t)$ such that $t \leq T$, we have*
  - $\mathsf{Gen}$ *runs in time* $\begin{cases} p(\lambda + \log T) & \text{for a fully-succinct SNARG} \\ p(\lambda + T) & \text{for a pre-processing SNARG} \end{cases}$
  - $\mathsf{Prove}$ *runs in time* $\begin{cases} p(\lambda + |M| + |x| + t + \log T) & \text{for fully-succinct SNARG} \\ p(\lambda + |M| + |x| + T) & \text{for pre-processing SNARG} \end{cases}$
  - $\mathsf{Ver}$ *runs in time $p(\lambda + |M| + |x| + \log T)$*
  - *a honestly generated proof has size $|\pi| = p(\lambda + \log T)$.*
- ***Adaptive Soundness.*** *For every non-uniform $\mathcal{A}$ of size $s(\lambda) = \mathsf{poly}(\lambda)$ there is a negligible function $\epsilon(\lambda)$ such that for every time bound $T \in \mathbb{N}$,*

$$\Pr\left[\begin{array}{c}\mathsf{Ver}(\mathsf{vst}, y, \pi) = 1 \\ \wedge\; y \notin \mathcal{L}_{\mathcal{R}}\end{array} \;\middle|\; \begin{array}{c}(\mathsf{prs}, \mathsf{vst}) \leftarrow \mathsf{Gen}(1^\lambda, T) \\ (y, \pi) \leftarrow \mathcal{A}(\mathsf{prs})\end{array}\right] \leq \epsilon(\lambda)$$

*Furthermore, we say that $\Pi$ has $(s, \epsilon)$-adaptive soundness if the above condition holds for concrete values $s$ and $\epsilon$.*

The notion of SNARG can be extended to be an argument of knowledge (a SNARK) by replacing soundness by an appropriate proof of knowledge property.

**Definition 3 (SNARKs [BCC+14]).** *A* succinct non-interactive argument of knowledge *(SNARK) for a relation $\mathcal{R} \subseteq \mathcal{R}_{\mathcal{U}}$ is a triple of algorithms $\Pi = (\mathsf{Gen}, \mathsf{Prove}, \mathsf{Ver})$ that constitutes a SNARG (as per Definition 2) except that soundness is replaced by the following property:*

- ***Adaptive Proof of Knowledge.*** *For every non-uniform prover $\mathcal{A}$ of size $s(\lambda) = \mathsf{poly}(\lambda)$ there exists a non-uniform extractor $\mathcal{E}_{\mathcal{A}}$ of size $t(\lambda) = \mathsf{poly}(\lambda)$ and a negligible function $\epsilon(\lambda)$ such that for every auxiliary input $aux \in \{0,1\}^{poly(\lambda)}$, and every time bound $T \in \mathbb{N}$,*

$$\Pr\left[\begin{array}{c}\mathsf{Ver}(\mathsf{vst}, y, \pi) = 1 \\ \wedge \\ (y, w) \notin \mathcal{R}\end{array} \;\middle|\; \begin{array}{c}(\mathsf{prs}, \mathsf{vst}) \leftarrow \mathsf{Gen}(1^\lambda, T) \\ (y, \pi) \leftarrow \mathcal{A}(\mathsf{prs}, aux) \\ w \leftarrow \mathcal{E}_{\mathcal{A}}(\mathsf{prs}, aux)\end{array}\right] \leq \epsilon(\lambda)$$

*Furthermore, we say that $\Pi$ satisfies $(s, t, \epsilon)$-adaptive proof of knowledge if the above condition holds for concrete values $(s, t, \epsilon)$.*

*Remark 1 (Publicly verifiable vs. designated verifier).* If security (adaptive PoK) holds against adversaries that have also access to the verification state $\mathsf{vst}$ (i.e., $\mathcal{A}$ receives the whole $\mathsf{crs}$) then the SNARK is called *publicly verifiable*, otherwise it is *designated verifier. In this work (and from now on) we consider publicly verifiable SNARKs.*

*Remark 2 (About extraction and auxiliary input).* First, we stress that in the PoK property the extractor $\mathcal{E}_{\mathcal{A}}$ takes exactly the same input of $\mathcal{A}$, including its random tape. Second, the PoK definition can also be relaxed to hold with respect to auxiliary inputs from specific distributions (instead of arbitrary ones). Namely, let $\mathcal{Z}$ be a probabilistic algorithm (called the auxiliary input generator) that outputs a string $aux$, and let compactly denote this process as $aux \leftarrow \mathcal{Z}$. Then we say that adaptive proof of knowledge holds for $\mathcal{Z}$ if the above definition holds for every $aux \leftarrow \mathcal{Z}$, where $\mathcal{Z}$ is also a non-uniform polynomial-size algorithm. More formally, we have the following definition.

**Definition 4 ($\mathcal{Z}$-auxiliary input SNARKs).** *$\Pi$ is a $\mathcal{Z}$-auxiliary input SNARK if $\Pi$ is a SNARK as in Definition 3 except that adaptive proof of knowledge holds for every auxiliary input $aux \leftarrow \mathcal{Z}$.*

For ease of exposition, in our proofs we compactly denote by $\mathsf{AdPoK}(\lambda, T, \mathcal{A}, \mathcal{E}_{\mathcal{A}}, \mathcal{Z})$ the adaptive proof of knowledge experiment executed with adversary $\mathcal{A}$, extractor $\mathcal{E}_{\mathcal{A}}$ and auxiliary input generator $\mathcal{Z}$, and that outputs 1 if $\mathsf{Ver}(\mathsf{vst}, y, \pi) = 1$ and $(y, w) \notin \mathcal{R}$. See below its description:

$\mathsf{AdPoK}(\lambda, T, \mathcal{A}, \mathcal{E}_{\mathcal{A}}, \mathcal{Z})$
$\mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda, T)$; $aux \leftarrow \mathcal{Z}(1^\lambda)$
$(y, \pi) \leftarrow \mathcal{A}(\mathsf{crs}, aux)$
$w \leftarrow \mathcal{E}_{\mathcal{A}}(\mathsf{crs}, aux)$
`if` $\mathsf{Ver}(\mathsf{crs}, y, \pi) = 1 \wedge (y, w) \notin \mathcal{R}$ `return` 1
`else return` 0

We say that $\Pi$ satisfies adaptive proof of knowledge for $\mathcal{Z}$-auxiliary input if for every non-uniform $\mathcal{A}$ of size $s(\lambda) = \mathsf{poly}(\lambda)$ there is a non-uniform extractor of size $t(\lambda) = \mathsf{poly}(\lambda)$ and a negligible function $\epsilon(\lambda)$ such that for every time bound $T$ we have

$$\Pr[\mathsf{AdPoK}(\lambda, T, \mathcal{A}, \mathcal{E}_{\mathcal{A}}, \mathcal{Z}) \Rightarrow 1] \leq \epsilon.$$

Furthermore, $\Pi$ has $(s, t, \epsilon)$-adaptive proof of knowledge for $\mathcal{Z}$-auxiliary input if the above condition holds for concrete $(s, t, \epsilon)$.

SNARKs for NP. A SNARK for the universal relation $\mathcal{R}_{\mathcal{U}}$ is called a universal SNARK. SNARKs for $\mathsf{NP}$ are instead SNARKs in which the verification algorithm $\mathsf{Ver}$ takes as additional input a constant $c > 0$, and adaptive proof of knowledge is restricted to hold only for relations $\mathcal{R}_c \subset \mathcal{R}_{\mathcal{U}}$. More formally,

**Definition 5 (SNARKs for NP).** *A SNARK for $\mathsf{NP}$ is a tuple of algorithms $\Pi = (\mathsf{Gen}, \mathsf{Prove}, \mathsf{Ver})$ satisfying Definition 3 except that the adaptive proof of knowledge property is replaced by the following one:*

– ***Adaptive Proof of Knowledge for*** $\mathsf{NP}$. *For every non-uniform polynomial-size prover* $\mathcal{A}$ *there exists a non-uniform polynomial-size extractor* $\mathcal{E}_{\mathcal{A}}$ *such that for every large enough* $\lambda \in \mathbb{N}$*, every auxiliary input* $aux \in \{0,1\}^{poly(\lambda)}$*, and every time bound* $T \in \mathbb{N}$*, and every constant* $c > 0$*,*

$$\Pr\left[\begin{array}{c|c} \mathsf{Ver}_c(\mathsf{vst}, y, \pi) = 1 & \mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda, T) \\ \wedge & (y, \pi) \leftarrow \mathcal{A}(\mathsf{crs}, aux) \\ (y, w) \notin \mathcal{R}_c & w \leftarrow \mathcal{E}_{\mathcal{A}}(\mathsf{crs}, aux) \end{array}\right] \leq negl(\lambda)$$

In the case of fully-succinct SNARKs for $\mathsf{NP}$, it is not necessary to provide a time bound as one can set $T = \lambda^{\log \lambda}$. In this case we can write $\mathsf{Gen}(1^\lambda)$ as a shorthand for $\mathsf{Gen}(1^\lambda, \lambda^{\log \lambda})$.

**Zero-Knowledge SNARKs.** Here we recall the zero-knowledge definition for SNARKs.

**Definition 6 (zk-SNARKs).** *$\Pi = (\mathsf{Gen}, \mathsf{Prove}, \mathsf{Ver})$ is a (statistical) zero-knowledge SNARK for a relation $\mathcal{R} \subseteq \mathcal{R}_{\mathcal{U}}$ if $\Pi$ is a SNARK and, moreover, satisfies the following property:*

ZERO-KNOWLEDGE. *There exists a stateful interactive polynomial-size simulator $S = (S^{\mathsf{crs}}, S^{\mathsf{Prove}})$ such that for all stateful interactive distinguishers $\mathcal{D}$, for every large enough security parameter $\lambda \in \mathbb{N}$, every auxiliary input $aux \in \{0,1\}^{poly(\lambda)}$, and every time bound $T \in \mathbb{N}$,*

$$\Pr\left[\begin{array}{c|c} (y, w) \in \mathcal{R} & (\mathsf{prs}, \mathsf{vst}) \leftarrow \mathsf{Gen}(1^\lambda, T) \\ \wedge & (y, w) \leftarrow \mathcal{D}(\mathsf{prs}, \mathsf{vst}, aux) \\ \mathcal{D}(\pi) = 1 & \pi \leftarrow \mathsf{Prove}(\mathsf{prs}, y, w) \end{array}\right] - \Pr\left[\begin{array}{c|c} (y, w) \in \mathcal{R} & (\mathsf{prs}, \mathsf{vst}, \mathsf{tr}) \leftarrow S^{\mathsf{crs}}(1^\lambda, T) \\ \wedge & (y, w) \leftarrow \mathcal{D}(\mathsf{prs}, \mathsf{vst}, aux) \\ \mathcal{D}(\pi) = 1 & \pi \leftarrow S^{\mathsf{Prove}}(\mathsf{prs}, \mathsf{tr}, y, aux) \end{array}\right] \leq negl(\lambda)$$

## 3 SNARKs in the presence of oracles

In this section we formalize the notion of extraction in the presence of oracles for SNARKs. We do this by proposing a suitable adaptive proof of knowledge definition, and we call a SNARK satisfying this definition a *SNARK in the presence of oracles* (O-SNARK, for short). As we shall see, the advantage of O-SNARKs is that this notion lends itself to easy and intuitive security proofs in all those applications where one needs to execute extractors in interactive security games with oracles (with a secret state). Below we provide the definition while the existence of O-SNARKs is discussed in Section 4.

### 3.1 O-SNARKs: SNARKs in the presence of oracles.

Let $\mathbb{O} = \{\mathcal{O}\}$ be a family of oracles. We denote by $\mathcal{O} \leftarrow \mathbb{O}$ the process of sampling an oracle $\mathcal{O}$ from the family $\mathbb{O}$ according to some (possibly probabilistic) process. For example, $\mathbb{O}$ can be a random oracle family, i.e., $\mathbb{O} = \{\mathcal{O} : \{0,1\}^\ell \to \{0,1\}^L\}$ for all possible functions from $\ell$-bits strings to $L$-bits strings, in which case $\mathcal{O} \leftarrow \mathbb{O}$ consists of choosing a function $\mathcal{O}$ uniformly at random in $\mathbb{O}$. As another example, $\mathbb{O}$ might be the signing oracle corresponding to a signature scheme, in which case the process $\mathcal{O} \leftarrow \mathbb{O}$ consists of sampling a secret key of the signature scheme according to the key generation algorithm (and possibly a random tape for signature generation in case the signing algorithm is randomized).

For any oracle family $\mathbb{O}$, we define an O-SNARK $\Pi$ for $\mathbb{O}$ as follows.

**Definition 7 ($\mathcal{Z}$-auxiliary input O-SNARKs for $\mathbb{O}$).** *We say that $\Pi$ is a $\mathcal{Z}$-auxiliary input O-SNARK for the oracle family $\mathbb{O}$, if $\Pi$ satisfies the properties of completeness and succinctness as in Definition 3, and the following property of adaptive proof of knowledge for $\mathbb{O}$:*

– ***Adaptive Proof of Knowledge for*** $\mathbb{O}$**.** *Consider the following experiment for security parameter* $\lambda \in \mathbb{N}$*, time bound* $T \in \mathbb{N}$*, adversary* $\mathcal{A}$*, extractor* $\mathcal{E}_{\mathcal{A}}$*, auxiliary input generator* $\mathcal{Z}$ *and oracle family* $\mathbb{O}$*:*

$\mathsf{O\text{-}AdPoK}(\lambda, T, \mathcal{A}, \mathcal{E}_{\mathcal{A}}, \mathcal{Z}, \mathbb{O})$
  $\mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda, T)$; $aux \leftarrow \mathcal{Z}(1^\lambda)$; $\mathcal{O} \leftarrow \mathbb{O}$
  $(y, \pi) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{crs}, aux)$
  $w \leftarrow \mathcal{E}_{\mathcal{A}}(\mathsf{crs}, aux, \mathsf{qt})$
  `if` $\mathsf{Ver}(\mathsf{crs}, y, \pi) = 1 \wedge (y, w) \notin \mathcal{R}$ `return` $1$
  `else return` $0$

*where* $\mathsf{qt} = \{q_i, \mathcal{O}(q_i)\}$ *is the transcript of all oracle queries and answers made and received by* $\mathcal{A}$ *during its execution.*

*Π satisfies adaptive proof of knowledge with respect to oracle family* $\mathbb{O}$ *and auxiliary input from* $\mathcal{Z}$ *if for every non-uniform* oracle *prover* $\mathcal{A}^{\mathcal{O}}$ *of size* $s(\lambda) = \mathsf{poly}(\lambda)$ *making at most* $Q(\lambda) = \mathsf{poly}(\lambda)$ *queries there exists a non-uniform extractor* $\mathcal{E}_{\mathcal{A}}$ *of size* $t(\lambda) = \mathsf{poly}(\lambda)$ *and a negligible function* $\epsilon(\lambda)$ *such that for every time bound* $T$*,*

$$\Pr[\mathsf{O\text{-}AdPoK}(\lambda, T, \mathcal{A}, \mathcal{E}_{\mathcal{A}}, \mathcal{Z}, \mathbb{O}) \Rightarrow 1] \leq \epsilon(\lambda)$$

*Furthermore, we say that Π satisfies* $(s, t, Q, \epsilon)$*-adaptive proof of knowledge with respect to oracle family* $\mathbb{O}$ *and auxiliary input from* $\mathcal{Z}$ *if the above condition holds for concrete values* $(s, t, Q, \epsilon)$*.*

### 3.2 Non-Adaptive O-SNARKs

In this section we define a relaxation of O-SNARKs in which the adversary is non-adaptive in making its queries to the oracle. Namely, we consider adversaries that first declare all their oracle queries $q_1, \ldots, q_Q$ and then run on input the common reference string as well as the queries' outputs $\mathcal{O}(q_1), \ldots, \mathcal{O}(q_Q)$. More formally,

**Definition 8 (**$\mathcal{Z}$**-auxiliary input non-adaptive O-SNARKs for** $\mathbb{O}$**).** *We say that Π is a* $\mathcal{Z}$*-auxiliary input* non-adaptive *O-SNARK for the oracle family* $\mathbb{O}$*, if Π satisfies the properties of completeness and succinctness as in Definition 3, and the following property of non-adaptive queries proof of knowledge for* $\mathbb{O}$*:*

– ***Non-Adaptive Proof of Knowledge for*** $\mathbb{O}$**.** *Consider the following experiment for security parameter* $\lambda \in \mathbb{N}$*, time bound* $T \in \mathbb{N}$*, adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, extractor* $\mathcal{E}_{\mathcal{A}}$*, auxiliary input generator* $\mathcal{Z}$ *and oracle family* $\mathbb{O}$*:*

$\mathsf{O\text{-}NonAdPoK}(\lambda, T, \mathcal{A}, \mathcal{E}_{\mathcal{A}}, \mathcal{Z}, \mathbb{O})$
  $(q_1, \ldots, q_Q, st) \leftarrow \mathcal{A}_1(1^\lambda)$
  $\mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda, T)$; $aux \leftarrow \mathcal{Z}(1^\lambda)$; $\mathcal{O} \leftarrow \mathbb{O}$
  $\mathsf{qt} = (q_1, \mathcal{O}(q_1), \ldots, q_Q, \mathcal{O}(q_Q))$
  $(y, \pi) \leftarrow \mathcal{A}_2(st, \mathsf{crs}, aux, \mathsf{qt})$
  $w \leftarrow \mathcal{E}_{\mathcal{A}}(\mathsf{crs}, aux, \mathsf{qt})$
  `if` $\mathsf{Ver}(\mathsf{crs}, y, \pi) = 1 \wedge (y, w) \notin \mathcal{R}$ `return` *1*
  `else return` *0*

*where st is simply a state information shared between* $\mathcal{A}_1$ *and* $\mathcal{A}_2$*.*

*Π satisfies non-adaptive proof of knowledge with respect to oracle family* $\mathbb{O}$ *and auxiliary input from* $\mathcal{Z}$ *if for every non-uniform prover* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *of size* $s(\lambda) = \mathsf{poly}(\lambda)$ *making at most*

$Q(\lambda) = \mathsf{poly}(\lambda)$ *non-adaptive queries there exists a non-uniform extractor* $\mathcal{E}_{\mathcal{A}}$ *of size* $t(\lambda) = \mathsf{poly}(\lambda)$ *and a negligible function* $\epsilon(\lambda)$ *such that for every time bound* $T$*,*

$$\Pr[\mathsf{O\text{-}NonAdPoK}(\lambda, T, \mathcal{A}, \mathcal{E}_{\mathcal{A}}, \mathcal{Z}, \mathbb{O}) \Rightarrow 1] \leq \epsilon(\lambda)$$

*Furthermore, we say that* $\Pi$ *satisfies* $(s, t, Q, \epsilon)$*-non-adaptive proof of knowledge with respect to oracle family* $\mathbb{O}$ *and auxiliary input from* $\mathcal{Z}$ *if the above condition holds for concrete values* $(s, t, Q, \epsilon)$*.*

**Existence of Non-Adaptive O-SNARKs from SNARKs.** Below we prove a simple result showing that non-adaptive O-SNARKs follow directly from classical SNARKs for which the proof of knowledge property holds for arbitrary auxiliary input distributions.

The idea of the proof is that the second stage adversary $\mathcal{A}_2$ of non-adaptive O-SNARKs is very much like a classical SNARK adversary that makes no queries and receives a certain auxiliary input which contains the set of oracle queries chosen by $\mathcal{A}_1$ with corresponding answers. The fact that the auxiliary input includes the set of queries chosen by $\mathcal{A}_1$, which is an arbitrary adversary, implies that the SNARK must support arbitrary, *not necessarily benign*, auxiliary inputs (i.e., it is not sufficient to fix an auxiliary input distribution that depends only on the oracle family $\mathbb{O}$).

**Theorem 4.** *Let* $\mathbb{O}$ *be any oracle family. If* $\Pi$ *is a SNARK satisfying* $(s, t, \epsilon)$*-adaptive PoK (for any auxiliary input), then* $\Pi$ *is a non-adaptive O-SNARK for* $\mathbb{O}$ *satisfying* $(s, t, Q, \epsilon)$*-non-adaptive PoK.*

*Proof.* Given the first stage adversary $\mathcal{A}_1$, and the oracle family $\mathbb{O}$ we define the following auxiliary input distribution:

$\mathcal{Z}_{\mathcal{A}_1,\mathbb{O}}(1^\lambda)$
  $(\{q_1, \ldots, q_Q\}, st) \leftarrow \mathcal{A}_1(1^\lambda)$
  $\mathcal{O} \xleftarrow{\$} \mathbb{O}$
  `return` $\langle st, \{q_i, \mathcal{O}(q_i)\}_{i=1}^{Q} \rangle$

Then, for any $(\mathcal{A}_1, \mathcal{A}_2)$ we can build the following SNARK adversary $\mathcal{B}$ taking $z \leftarrow \mathcal{Z}_{\mathcal{A}_1,\mathbb{O}}$:

$\mathcal{B}(\mathsf{crs}, z)$
  Parse $z = \langle st, q_1, y_1, \ldots, q_Q, y_Q \rangle$
  Run $\mathcal{A}_2(st, \mathsf{crs}, \mathsf{qt} = (q_1, y_1, \ldots, q_Q, y_Q)) \rightarrow (y, \pi)$
  Return the same $(y, \pi)$ returned by $\mathcal{A}_2$.

Since $\Pi$ is by assumption a SNARK, for $\mathcal{B}$ there exists an extractor $\mathcal{E}_{\mathcal{B}}$ such that, for *any* auxiliary input (and in particular for auxiliary input from $\mathcal{Z}_{\mathcal{A}_1,\mathbb{O}}$), it holds

$$\Pr[\mathsf{O\text{-}AdPoK}(\lambda, T, \mathcal{B}, \mathcal{E}_{\mathcal{B}}, \mathcal{Z}_{\mathcal{A}_1,\mathbb{O}}) \Rightarrow 1] \leq \epsilon(\lambda)$$

Finally, we simply define $\mathcal{E}_{\mathcal{A}} = \mathcal{E}_{\mathcal{B}}$. Since $\mathcal{B}$'s simulation of $\mathcal{A}_2$ is perfect, it is easy to see that for any $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ this extractor $\mathcal{E}_{\mathcal{A}}$ is such that

$$\Pr[\mathsf{O\text{-}NonAdPoK}(\lambda, T, \mathcal{A}, \mathcal{E}_{\mathcal{A}}, \mathbb{O}) \Rightarrow 1] \leq \epsilon(\lambda)$$

□

## 4 On the existence of O-SNARKs

In this section we study whether O-SNARKs exist and under what assumptions. In the following sections we give both positive and negative answers to this question.

### 4.1 O-SNARKs in the random oracle model from Micali's CS proofs

In this section we briefly discuss how the construction of CS proofs of Micali [Mic00] can be seen as an O-SNARK for *any* oracle family, albeit in the random oracle model. To see this, we rely on the result of Valiant [Val08] who shows that Micali's construction is a "CS proof of knowledge" in the random oracle model. The main observation is in fact that Valiant's proof works by showing a *black-box* extractor working for any prover.

**Proposition 1.** *Let $\mathbb{O}$ be any oracle family and $\mathsf{RO}$ be a family of random oracles. Let $\Pi_{\mathsf{Mic}}$ be the CS proof construction from [Mic00]. Then $\Pi_{\mathsf{Mic}}$ is an O-SNARK for $(\mathsf{RO}, \mathbb{O})$, in the random oracle model.*

*Proof (Sketch).* Let $\mathcal{E}^{\mathsf{RO}}$ be Valiant's black-box extractor[9] which takes as input the code of the prover and outputs a witness $w$. For any adversary $\mathcal{A}^{\mathsf{RO},\mathcal{O}}$ we can define its extractor $\mathcal{E}_{\mathcal{A}}$ as the one that, on input the query transcript $\mathsf{qt}$ of $\mathcal{A}$, executes $w \leftarrow \mathcal{E}^{\mathsf{RO}}(\mathcal{A})$ by simulating all the random oracle queries of $\mathcal{E}^{\mathsf{RO}}$ using $\mathsf{qt}$, and finally outputs the same $w$. The reason why $\mathsf{qt}$ suffices to $\mathcal{E}_{\mathcal{A}}$ for simulating random oracle queries to $\mathcal{E}^{\mathsf{RO}}$ is that Valiant's extractor $\mathcal{E}^{\mathsf{RO}}$ makes exactly the same queries of the prover.

### 4.2 Impossibility of O-SNARKs for every family of oracles, in the standard model

In this section we show that, *in the standard model*, there do not exist O-SNARKs with respect to *every* family of oracles. We show this under the assumption that CRHFs exist. In other words, we show an oracle family in the presence of which any candidate O-SNARK that is correct and succinct cannot satisfy adaptive proof of knowledge with respect to that oracle family.

Before giving its formal statement, we provide an intuition of this result. Given a CRHF $h$, consider the $\mathsf{NP}$ binary relation $\tilde{R}_h = \{(x, w) : h(w) = x\}$, and let $\Pi$ be a SNARK for $\mathsf{NP}$ and consider $p(\cdot)$ the polynomial for which $\Pi$ is succinct. The idea is to show an oracle family $\tilde{\mathbb{O}}$ and an adversary $\bar{\mathcal{A}}$ for which there is no extractor unless $h$ is not collision-resistant. For every polynomial $p(\cdot)$, the oracle family contains oracles $\mathcal{O}_p$ that given a query $q$, interpret $q$ as the description of a program $\mathcal{P}(\cdot,\cdot)$, samples a random $w$, computes $x = h(w)$, and outputs $x$ along with $\pi \leftarrow \mathcal{P}(x, w)$. If $\mathcal{P}(\cdot,\cdot) = \mathsf{Prove}(\mathsf{crs}, \cdot, \cdot)$, then the oracle is simply returning an hash image with a proof of knowledge of its (random) preimage. The adversary $\bar{\mathcal{A}}^{\mathcal{O}_p}$ is the one that on input $q = \mathcal{P}(\cdot,\cdot) = \mathsf{Prove}(\mathsf{crs}, \cdot, \cdot)$ simply asks one query $(x, \pi) \leftarrow \mathcal{O}_p(q)$ and outputs $(x, \pi)$. Now, the crucial point that entails the non-existence of an extractor is that, provided that $w$ is sufficiently longer than $\pi$, every valid extractor for such $\bar{\mathcal{A}}$ that outputs a valid $w'$ immediately implies a collision $(w, w')$ for $h$.[10]

A last detail must be added to avoid that $\pi \leftarrow \mathcal{P}(\cdot,\cdot)$ reveals significant information about $w$ as an adversarially chosen program $\mathcal{P}$ may output something else than a SNARK proof. To take into

[9] The CS proofs of knowledge definition used by Valiant considers adversaries that are non-adaptive in choosing the statement. However it easy to see that the construction and the proof work also for the adaptive case.

[10] This relies on the fact that sufficiently many bits of $w$ remain unpredictable, even given $\pi$.

account this issue, the oracle $\mathcal{O}_p$ is defined such that for every $\pi$ obtained by running $\mathcal{P}$ it returns $\pi$ only if this is of length $|\pi| \leq p(\lambda)$; otherwise it simply returns 0 and does not reveal anything about the input.

Remark that the oracle $\mathbb{O}_p$ does not depend on the specific O-SNARK construction $\Pi$ but only on the universal polynomial $p(\cdot)$ bounding the succinctness of $\Pi$.

More formally, we prove the following theorem:

**Theorem 5.** *Assume that collision resistant hash functions exist. Then for every polynomial $p(\cdot)$ there exists an oracle family $\mathbb{O}_p$ such that every candidate O-SNARK $\Pi$ for $\mathsf{NP}$, that is correct and succinct with proofs of length bounded by $p(\cdot)$, does not satisfy adaptive proof of knowledge for $\mathbb{O}_p$.*

*Proof.* Let $\mathcal{H} = \{h : \{0,1\}^{q(\lambda)} \to \{0,1\}^{\ell(\lambda)}\}$ be a family of CRHFs where $q(\lambda) \geq p(\lambda) + \ell(\lambda) + \lambda$. Let $M_h(x,w)$ be the machine that on input $(x,w)$ accepts iff $h(w) = x$, and $\mathcal{R}_h$ be the relation consisting of all pairs $(y,w)$ such that, for $y = (M_h, x, t)$, $M_h(x,w)$ accepts in at most $t$ steps.

Given the CRHF family $\mathcal{H}$ and the polynomial $p(\cdot)$, we define the oracle family $\mathbb{O}_p = \{\mathcal{O}_p : \{0,1\}^* \to \{0,1\}^{t(\lambda)}\}$ where every oracle $\mathcal{O}_p$ is associated with a function $h \in \mathcal{H}$ and is defined as follows:

$\mathcal{O}_p(q)$**:** interprets the query $q$ as the description of a program $\mathcal{P}(\cdot,\cdot)$
samples $w \xleftarrow{\$} \{0,1\}^{q(\lambda)}$,
computes $x = h(w)$, $t = \#M_h(x,w)$,
sets $y = (M_h, x, t)$, and runs $\pi \leftarrow \mathcal{P}(y,w)$.
tests that $\pi$ is of length $|\pi| \leq p(\lambda)$,
if yes returns $(y,\pi)$, otherwise return 0.

Moreover, the process of selecting $\mathcal{O}_p \leftarrow \mathbb{O}_p$ is that of sampling a function $h \leftarrow \mathcal{H}$ as well as a suitable random tape. Essentially, $\mathcal{O}_p$ is returning $y = (M_h, x, t)$ containing an hash image $x$ together with $\pi$, a proof of knowledge of its preimage $w$.

To show that any $\Pi$ with proofs of length bounded by $p(\cdot)$ is not an O-SNARK for $\mathbb{O}_p$ (under the assumption that CRHFs exist), we will prove that there is a polynomial-size adversary $\bar{\mathcal{A}}$ against $\Pi$ such that for every polynomial-size candidate extractor $\mathcal{E}$

$$\Pr[\mathsf{O\text{-}AdPoK}(\lambda, \bar{\mathcal{A}}, \mathcal{E}, \mathbb{O}_p) \Rightarrow 0] \leq \nu_{\mathcal{H}}(\lambda) + 2^{-\lambda}$$

where $\nu_{\mathcal{H}}(\lambda)$ is the advantage of any adversary against $\mathcal{H}$'s collision resistance.

The adversary $\bar{\mathcal{A}}(\mathsf{crs})$ is very simple: It sets $q := \mathcal{P}(\cdot,\cdot) = \mathsf{Prove}(\mathsf{crs}, \cdot, \cdot)$ (i.e., the $\mathsf{Prove}$ algorithm of $\Pi$ with hardcoded $\mathsf{crs}$) and it makes a single query $(y,\pi) \leftarrow \mathcal{O}(q)$ and returns $(y,\pi)$. Now, assume by contradiction there is a polynomial-size extractor $\mathcal{E}$ such that the above probability is greater than some non-negligible $\epsilon(\lambda)$. We show how to build an adversary $\mathcal{C}$ that breaks collision resistance with probability at least $\epsilon(\lambda) - 2^{-\lambda}$.

$\mathcal{C}$ receives an instance $h$ of $\mathcal{H}$, generates $\mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda)$, and runs $\bar{\mathcal{A}}^{\mathcal{O}_p}(\mathsf{crs})$ simulating the oracle $\mathcal{O}_p$ on the single query $q := \mathcal{P}(\cdot,\cdot) = \mathsf{Prove}(\mathsf{crs}, \cdot, \cdot)$ asked by $\bar{\mathcal{A}}$. In particular, let $w$ be the string chosen by $\mathcal{C}$ for answering this query. Notice that such simulation can be done perfectly in a straightforward way, and that $\bar{\mathcal{A}}$'s output is the same pair $(y,\pi)$ created by $\mathcal{C}$. Next, $\mathcal{C}$ runs $w' \leftarrow \mathcal{E}(\mathsf{crs}, \mathsf{qt} = (\mathcal{P}(\cdot,\cdot), (y,\pi)))$, and outputs $(w,w')$.

By correctness of $\Pi$ it holds that the pair $(y,\pi)$ returned by $\bar{\mathcal{A}}$ satisfies $\mathsf{Ver}(\mathsf{crs}, y, \pi) = 1$. Thus, by our contradiction assumption, with probability $\geq \epsilon(\lambda)$, $\mathcal{E}$ outputs $w'$ such that $(y,w') \in \mathcal{R}_h$.

Namely, $h(w') = x = h(w)$. To show that this is a collision, we argue that, information-theoretically, $w' \neq w$ with probability $\geq 1 - 1/2^{\lambda}$. Intuitively, this follows from the fact that $w$ is randomly chosen of length $q(\lambda) \leq p(\lambda) + \ell(\lambda) + \lambda$ and the only information about $w$ which is leaked to $\mathcal{E}$ is through $\pi$ and $x = h(w)$ whose lengths are at most $p(\lambda) + \ell(\lambda)$. Therefore there are at least $\lambda$ bits of entropy in $w$, from which $\Pr[w' = w] \leq 2^{-\lambda}$ over the random choice of $w$. Hence, $\mathcal{C}$ can break the collision resistance of $\mathcal{H}$ with probability $\geq \epsilon(\lambda) - 2^{-\lambda}$. □

*Remark 3 (Extension to signing oracles.).* We point out that the code of our oracle can be made part of the signing algorithm of a signature scheme. Namely, one can modify any signature scheme so that, by interpreting the message as a program description, it does the same as oracle $\mathcal{O}_p$, and appends $(y, \pi)$ to the signature. It is possible to prove that such signature scheme remains unforgeable, yet the corresponding signing oracle provides the same functionality as the oracle family $\mathbb{O}_p$ for which O-SNARKs do not exist. However, we do not go further with the formalization of this result for signing oracles as in the next section we can prove a stronger impossibility, which relies only on the existence of secure signatures.

### 4.3 Impossibility of O-SNARKs for every family of signing oracles

In this section we show that, assuming that unforgeable signature schemes exist, there do not exist O-SNARKs with respect to *every* family of signing oracles (no matter if the corresponding signature scheme is secure). Precisely, we show a secure signature scheme $\Sigma_p$ such that every correct and succinct O-SNARK $\Pi$ cannot satisfy adaptive proof of knowledge in the presence of the signing oracle corresponding to $\Sigma_p$.

For every signature scheme $\Sigma = (\mathsf{kg}, \mathsf{sign}, \mathsf{vfy})$ we let $\mathbb{O}_\Sigma$ be the family of oracles $\mathcal{O}(m) = \mathsf{sign}(\mathsf{sk}, m)$, where every family member $\mathcal{O}$ is described by a secret key $\mathsf{sk}$ of the signature scheme, i.e., the process $\mathcal{O} \leftarrow \mathbb{O}_\Sigma$ corresponds to obtaining $\mathsf{sk}$ through a run of $(\mathsf{sk}, \mathsf{vk}) \xleftarrow{\$} \mathsf{kg}(1^\lambda)$. For the sake of simplicity, we also assume that the oracle allows for a special query, say $\mathcal{O}(\text{`}vk'),$[11] whose answer is the public verification key $\mathsf{vk}$.

**Theorem 6.** *Assume that* UF-CMA*-secure signature schemes exist. Then for every polynomial $p(\cdot)$ there exists a* UF-CMA*-secure signature scheme $\Sigma_p$ such that every candidate O-SNARK $\Pi$ for* NP*, that is correct and succinct with proofs of length bounded by $p(\cdot)$, does not satisfy adaptive proof of knowledge with respect to $\mathbb{O}_{\Sigma_p}$.*

To prove the theorem, we proceed in two main steps. First, we describe the construction of the signature scheme $\Sigma_p$ based on any other UF-CMA-secure signature scheme $\widehat{\Sigma}$ with message space $\mathcal{M} = \{0,1\}^*$, and we prove that $\Sigma_p$ is secure as long as $\widehat{\Sigma}$ is secure. Second, we show that, when considering the oracle family $\mathbb{O}_{\Sigma_p}$ corresponding to the signature scheme $\Sigma_p$, a correct $\Pi$ with succinctness $p(\cdot)$ cannot be an O-SNARK for $\mathbb{O}_{\Sigma_p}$, i.e., we show an efficient O-SNARK adversary $\mathcal{A}_p^{\mathcal{O}}$ (with access to a $\Sigma_p$ signing oracle $\mathcal{O}(\cdot) = \mathsf{sign}(\mathsf{sk}, \cdot)$), for which there is no extractor unless the signature scheme $\Sigma_p$ is insecure.

**The counterexample signature scheme $\Sigma_p$.** Let $\widehat{\Sigma}$ be any UF-CMA-secure scheme with message space $\mathcal{M} = \{0,1\}^*$. We construct the signature scheme $\Sigma_p$ from $\widehat{\Sigma}$ and using the NP relation $\mathcal{R}_{\Sigma_p}$ defined as follows:

[11] Here $vk$ is an arbitrary choice; any symbol not in $\mathcal{M}$ would do so. Introducing the extra query simplifies the presentation, otherwise $\mathsf{vk}$ should be treated as an auxiliary input from a distribution generated together with the oracle sampling.

Let $M_{\Sigma_p}(x,w)$ be the random-access that on input a pair $(x=(\mathsf{vk},m), w=\sigma)$, $M_{\Sigma_p}$ accepts iff $\mathsf{vfy}(\mathsf{vk},m,\sigma)=1$. Associated to such machine there is also a polynomial bound $t_{\Sigma_p}(k)=k^{e_{\Sigma_p}}$ which depends on the running time of $\Sigma_p$'s verification algorithm. We call $\mathcal{R}_{\Sigma_p}$ the $\mathsf{NP}$ binary relation consisting of all pairs $(y,w)$ such that, parsing $y=(M_{\Sigma_p},x,t)$, $M_{\Sigma_p}(x,w)$ accepts in at most $t$ steps and $t \leq t_{\Sigma_p}(|x|)$.

As it will be clear from the construction, there is no circularity in building $\Sigma_p$ using $\mathcal{R}_{\Sigma_p}$ as the relation uses only the verification algorithm and $\Sigma_p$ uses $\mathcal{R}_{\Sigma_p}$ only in the signing algorithm.

The scheme $\Sigma_p$ has message space $\mathcal{M}=\{0,1\}^*$; its algorithms work as follows:

$\mathsf{kg}(1^\lambda)$: Run $(\widehat{\mathsf{vk}},\widehat{\mathsf{sk}}) \xleftarrow{\$} \widehat{\Sigma}.\mathsf{kg}(1^\lambda)$, set $\mathsf{vk}=\widehat{\mathsf{vk}}$, $\mathsf{sk}=\widehat{\mathsf{sk}}$, and also set $\delta=p(\lambda)+\lambda$.
$\mathsf{sign}(\mathsf{sk},m)$: Signing works as follows
- sample $r \xleftarrow{\$} \{0,1\}^\delta$, and compute $\hat{\sigma} \leftarrow \widehat{\Sigma}.\mathsf{sign}(\widehat{\mathsf{sk}}, r|m)$;
- sample $m' \xleftarrow{\$} \{0,1\}^\lambda$, $r' \xleftarrow{\$} \{0,1\}^\delta$, compute $\hat{\sigma}' \leftarrow \widehat{\Sigma}.\mathsf{sign}(\widehat{\mathsf{sk}}, r'|m')$, and set $\sigma'=(r',\hat{\sigma}',0,0)$;
- set $x=(\mathsf{vk},m')$ and $w=\sigma'$;
- let $t=\#M_{\Sigma_p}(x,w)$, and $y=(M_{\Sigma_p},x,t)$;
- interpret $m$ as the description of program $\mathcal{P}(\cdot,\cdot)$ and thus run $\pi \leftarrow \mathcal{P}(y,w)$;
- if $|\pi|>p(\lambda)$ set $\pi'=0$ and $m'=0$, else $\pi'=\pi$.
- output $\sigma=(r,\hat{\sigma},m',\pi')$.

$\mathsf{vfy}(\mathsf{vk},m,\sigma)$: Parse $\sigma=(r,\hat{\sigma},m',\pi')$ where $r$ is $\delta$-bits long, and return the output of $\widehat{\Sigma}.\mathsf{vfy}(\widehat{\mathsf{vk}}, r|m, \hat{\sigma})$.

It is trivial to check that, as long as $\widehat{\Sigma}$ is correct, $\Sigma_p$ is also correct. Before proving the security of $\Sigma_p$, we provide some intuitions about the rational of the above construction:

- The signing algorithm consists of two main steps. First, given the message $m$, we sign $r|m$ (i.e., we prepend the random string $r$ to $m$) using $\widehat{\Sigma}.\mathsf{sign}$. Second, we generate another signature on a randomly chosen message $m' \in \{0,1\}^\lambda$. This is done following the same process as for $m$, i.e., by sampling a random $r' \xleftarrow{\$} \{0,1\}^\delta$ and signing $r'|m'$ using $\widehat{\Sigma}.\mathsf{sign}$. Next, we construct a theorem $(y,w)$ for $\mathcal{R}_{\Sigma_p}$ to prove existence of a valid signature on message $m'$. Finally, one interprets the input message $m$ as a program $\mathcal{P}$ description and runs $\pi \leftarrow \mathcal{P}(y,w)$. Its output, together with $m'$, is included in the signature as an extra information. As one can see, if $\mathcal{P}(\cdot,\cdot)=\mathsf{Prove}(\mathsf{crs},\cdot,\cdot)$ then $\pi$ is a valid proof. Yet, before returning $\pi$ we make sure that (regardless of its validity) $\pi$ is short enough.
- The meaning of $\delta$: The reason of adding the random string $r$ to the signature is to increase the entropy of the signatures generated by $\Sigma_p$. This is crucial for the signatures $\sigma'$ that are used to generate $\pi$. Basically, we want to make sure that $\pi$ cannot leak enough information about $\sigma'$. Since the output of the program is of length $p$, taking $r'$ to be sufficiently long – of $p(\lambda)+\lambda$ bits – guarantees that some information about $\sigma'$ is inevitably lost.

**Lemma 1.** *If $\widehat{\Sigma}$ is $\mathsf{UF\text{-}CMA}$-secure scheme, then $\Sigma_p$ is $\mathsf{UF\text{-}CMA}$-secure. More precisely, for any PPT adversary $\mathcal{A}_p$ that has advantage $\epsilon(\lambda)$ in breaking the security of $\Sigma_p$ by making $Q$ signing queries there is a PPT adversary $\hat{\mathcal{A}}$ that breaks the security of $\hat{\Sigma}$ with advantage $> \epsilon(\lambda) - Q/2^\lambda$ by making $2Q$ queries.*

*Proof.* Assume by contradiction that there exists an adversary $\mathcal{A}_p$ that has non-negligible advantage $\epsilon(\lambda)$ against the $\mathsf{UF\text{-}CMA}$ security of $\Sigma_p$ while running in polynomial time and making $Q$ queries to the $\mathsf{sign}(\mathsf{sk},\cdot)$ oracle. Starting from this adversary $\mathcal{A}_p$, we construct a PPT adversary $\hat{\mathcal{A}}$ that is able to break the $\mathsf{UF\text{-}CMA}$ security of $\widehat{\Sigma}$ with non-negligible advantage and by making at most $2Q$ queries to its signing oracle $\widehat{\Sigma}.\mathsf{sign}(\widehat{\mathsf{sk}},\cdot)$.

We define $\hat{\mathcal{A}}$ (which gets the public key $\widehat{\mathsf{vk}}$ and makes queries to $\widehat{\Sigma}.\mathsf{sign}(\widehat{\mathsf{sk}},\cdot)$ oracle) as follows:

$\hat{\mathcal{A}}^{\widehat{\Sigma}.\mathsf{sign}(\widehat{\mathsf{sk}},\cdot)}(\widehat{\mathsf{vk}})$

Run $(m^*, \sigma^*) \leftarrow \mathcal{A}_p{}^{\mathsf{sign}(\widehat{\mathsf{sk}},\cdot)}(\widehat{\mathsf{vk}})$ and simulate queries $m$ to $\mathsf{sign}(\widehat{\mathsf{sk}}, \cdot)$ as follows:
sample $r \xleftarrow{\$} \{0,1\}^\delta$, and query $\hat{\sigma} \leftarrow \widehat{\Sigma}.\mathsf{sign}(\widehat{\mathsf{sk}}, r|m)$;
sample $m' \xleftarrow{\$} \{0,1\}^\lambda, r' \xleftarrow{\$} \{0,1\}^\delta$;
query $\hat{\sigma}' \leftarrow \widehat{\Sigma}.\mathsf{sign}(\widehat{\mathsf{sk}}, r'|m')$, and set $\sigma' = (r', \hat{\sigma}', 0, 0)$;
set $x = (\mathsf{vk}, m'_\lambda, \tau)$, $w = (m'_\lambda, \sigma')$, where $\tau = m_1| \cdots |m_{\lambda-1}$;
let $t = \#M_{\Sigma_p,I}(x, w)$, and set $y = (M_{\Sigma_p,I}, x, t)$, where $I$ is the identity function;
Interpret $m$ as the description of program $\mathcal{P}(\cdot,\cdot)$ and thus run $\pi \leftarrow \mathcal{P}(y, w)$;
if $|\pi| > p(\lambda)$ set $\pi' = 0$ and $m' = 0$, else $\pi' = \pi$;
output $\sigma = (r, \hat{\sigma}, m', \pi')$.
Parse $\sigma^* = (r^*, \hat{\sigma}^*, \cdot, \cdot)$ and return $(r^*|m^*, \hat{\sigma}^*)$

Let us now show that whenever $\mathcal{A}_p$ succeeds in the simulation described above, $\hat{\mathcal{A}}$ succeeds in breaking the $\mathsf{UF\text{-}CMA}$ security of the scheme $\widehat{\Sigma}$, with all but negligible probability. To this end we have to first show that the simulation provided by $\hat{\mathcal{A}}$ work correctly, and then show that $\hat{\mathcal{A}}$ outputs a valid forgery as long as $\mathcal{A}_p$ outputs a forgery.

To ease the analysis, consider the set of all queries (and corresponding responses) made by $\mathcal{A}_p$:

$$\mathcal{Q}^* = \{m_i, \sigma_i = (r_i, \hat{\sigma}_i, m'_i, \pi_i) \,|\, i = 1 \dots Q\}$$

Then the set of $\hat{\mathcal{A}}$'s queries is $\widehat{\mathcal{Q}} = \widehat{\mathcal{Q}}^* \cup \widehat{\mathcal{Q}}'$ with

$$\widehat{\mathcal{Q}}^* = \{(\hat{m}_i = r_i|m_i,\ \hat{\sigma}_i) \,\big|\, i = 1 \dots Q\}$$

$$\widehat{\mathcal{Q}}' = \{(\hat{m}'_j = r'_j|m'_j,\ \hat{\sigma}'_j) \,\big|\, j = 1 \dots Q\}$$

Precisely, the first set $\widehat{\mathcal{Q}}^*$ consists of all signing queries asked by $\hat{\mathcal{A}}$ to its oracle for signing the messages $m_i$ queried by $\mathcal{A}_p$. The second set $\widehat{\mathcal{Q}}'$ instead, comprises the extra queries asked by $\hat{\mathcal{A}}$ in the simulation for signing the sampled messages $m'_j$.

It is easy to see that $\hat{\mathcal{A}}$ provides a perfect simulation to $\mathcal{A}_p$ as $\hat{\mathcal{A}}$ can correctly answer every query of $\mathcal{A}_p$ using its own signing oracle.

So the main fact to show is that the message $m^*$ used in the forgery leads to a *new* message $\hat{m}^*$ in the game played by $\hat{\mathcal{A}}$.

Let $(m^*, \sigma^* = (r^*, \hat{\sigma}^*, \cdot, \cdot))$ be a valid forgery for $\Sigma_p$ (i.e., $m^* \neq m_i$, for all $m_i \in \mathcal{Q}^*$), and let us consider the following undesired cases:

1. $(\hat{m}^* = r^*|m^*, \cdot) \in \widehat{\mathcal{Q}}^*$: Since $m^* \neq m_i\ \forall m_i \in \mathcal{Q}^*$ this case cannot occur even if the corresponding strings $r_i$ and $r^*$ match.
2. $(\hat{m}^* = r^*|m^*, \cdot) \in \widehat{\mathcal{Q}}'$: It must be that $\hat{m}^* = r^*|m^* = r'_j|m'_j = \hat{m}'_j$ for some $j \in \{1, \dots, Q\}$. In what follows we bound the probability that such equality happens and show that it is negligible.

Both $r^*$ and $r'_j$ are parsed as strings of the *same* length $\delta = p(\lambda) + \lambda$. Hence, $r^*|m^* = r'_j|m'_j$ immediately implies $m^* = m'_j$, which may be possible since $m^*$ is of adversarial choice. To bound the probability of match we thus only look at the event that $r^* = r'_j$.

Now, the crucial observation is that the adversary $\mathcal{A}_p$ never sees the strings $r'_j$ explicitly, and thus the probability of the match can be upper bounded by the probability that the adversary $\mathcal{A}_p$ guesses correctly the string $r'_j \in \{0,1\}^\delta$ where $\delta = p(\lambda) + \lambda$.

Below we argue that this happens with negligible probability $\leq \frac{Q}{2^\lambda}$. For $j \in \{1, \dots Q\}$ let $\mathsf{Bad}_j$ be the event that $r^* = r'_j$ for $(\hat{m}'_j, \hat{\sigma}'_j) \in \hat{\mathcal{Q}}'$ and let $\mathsf{Bad} = \bigvee_{j=1}^{Q} \mathsf{Bad}_j$. Using the union bound we have:

$$\Pr[\mathsf{Bad}] = \Pr\left[\bigvee_{j=1}^{Q} \mathsf{Bad}_j\right] \leq \sum_{j=1}^{Q} \Pr\left[\mathsf{Bad}_j\right]$$

Now, we will bound the probability of $\mathsf{Bad}_j$ for *any* fixed $j$. The value $\pi_j \leftarrow \mathcal{P}(y, w)$ the adversary $\mathcal{A}_p$ gets from the $j$-th query is the only one that can reveal some information about $\sigma'_j = (r'_j, \hat{\sigma}'_j, 0, 0)$ and implicitly about $r'_j$. We show that the information gained from $\pi_j$ does not give any advantage to the adversary.

Let us represent the process of running $\pi \leftarrow \mathcal{P}(y, (m'_\lambda, \sigma'_j = (R, \hat{\sigma}'_j, 0, 0)))$ and returning $\pi$ only if $|\pi| \leq p(\lambda)$ as a function $f(R)$ such that $f : \{0,1\}^\delta \to \{0,1\}^p$. Namely, we fix $\mathcal{P}$ and all its inputs but $R$. Observe that, for random chosen inputs, any such $f$ is essentially performing a lossy data compression of $\delta - p(\lambda) = \lambda$ bits.

We have that $\forall f : \{0,1\}^\delta \to \{0,1\}^p$, the probability that any algorithm $\mathcal{A}$ guesses the random string $r'_j$ on input $f(r'_j)$ is less than $\frac{2^p}{2^\delta} = \frac{1}{2^\lambda}$. The same holds for $\mathcal{A}_p$ and $f$ defined as before. Hence, summing up these probabilities for all $j$ and observing that $Q = \mathsf{poly}(\lambda)$, we obtain:

$$\Pr[\mathsf{Bad}] \leq \sum_{j=1}^{Q} \Pr\left[Bad_j\right] \leq \frac{Q}{2^\lambda}$$

that is negligible.

The proof is concluded by observing that

$$\mathbf{Adv}^{\mathsf{UF\text{-}CMA}}_{\bar{\mathcal{A}}, \widehat{\Sigma}}(\lambda) \geq \epsilon(\lambda)(1 - \Pr[\mathsf{Bad}]) \geq \epsilon(\lambda) - \frac{Q}{2^\lambda}$$

□

**Impossibility of O-SNARKs for $\mathbb{O}_{\Sigma_p}$.** To show that $\Pi$ is not an O-SNARK for $\mathbb{O}_{\Sigma_p}$ (under the assumption that $\widehat{\Sigma}$ is secure), we will prove that there is an adversary $\bar{\mathcal{A}}$ such that every candidate extractor $\mathcal{E}$ fails in the adaptive proof of knowledge game.

**Lemma 2.** *If $\widehat{\Sigma}$ is* UF-CMA*-secure then every $\Pi$ for* NP *that is correct and succinct with proofs of length $p(\cdot)$ is not an O-SNARK for $\mathbb{O}_{\Sigma_p}$.*

*Proof.* Let $\bar{\mathcal{A}}$ be the following adversary: on input $\mathsf{crs}$, encode the $\mathsf{Prove}$ algorithm of $\Pi$ with hardcoded $\mathsf{crs}$ as a program $\mathcal{P}(\cdot,\cdot) := \mathsf{Prove}(\mathsf{crs}, \cdot, \cdot)$; let $q$ be $\mathcal{P}$'s description, and make a single query $\sigma = (r, \hat{\sigma}, m', \pi') \leftarrow \tilde{\mathcal{O}}(q)$; return $(y, \pi')$ where $y = (M_{\Sigma_p}, x, t)$ is appropriately reconstructed.

We show that for every polynomial-size extractor $\mathcal{E}$ it holds

$$\Pr[\mathsf{O\text{-}AdPoK}(\lambda, \bar{\mathcal{A}}, \mathcal{E}, \mathbb{O}_{\Sigma_p}) \Rightarrow 0] \leq \nu_{\widehat{\Sigma}}(\lambda) + 2^{-\lambda}$$

where $\nu_{\widehat{\Sigma}}(\lambda) = \mathbf{Adv}^{\mathsf{UF\text{-}CMA}}_{\mathcal{B}, \widehat{\Sigma}}(\lambda)$ is the advantage of an adversary $\mathcal{B}$ against the UF-CMA-security of $\widehat{\Sigma}$. This means that there is no extractor unless the signature scheme $\widehat{\Sigma}$ is not secure.

We proceed by contradiction assuming the existence of a polynomial-size extractor $\mathcal{E}$ such that the above probability is greater than some non-negligible $\epsilon$. We show how to build an adversary $\mathcal{B}$ that breaks UF-CMA-security of $\widehat{\Sigma}$ with probability at least $\epsilon$.

$\mathcal{B}^{\widehat{\Sigma}.\mathsf{sign}}$ receives a verification key $\widehat{\mathsf{vk}}$ for $\widehat{\Sigma}$, generates $\mathsf{crs}\leftarrow\mathsf{Gen}(1^\lambda)$ and runs $\bar{\mathcal{A}}^{\mathcal{O}}(\mathsf{crs})$ simulating its single query $q$ to oracle $\mathcal{O}$ by running exactly as $\Sigma_p.\mathsf{sign}$ except that the signatures $\hat{\sigma}$ and $\sigma'$ are generated by issuing queries to the signing oracle $\widehat{\Sigma}.\mathsf{sign}$. Let $(y, \pi')$ be $\bar{\mathcal{A}}$'s output, and observe that by correctness of $\Pi$ it holds that $\mathsf{Ver}(\mathsf{crs}, y, \pi') = 1$.

Next, $\mathcal{B}$ runs $w^*\sigma^*\leftarrow\mathcal{E}(\mathsf{crs}, \mathsf{qt})$, where $\mathsf{qt}$ contains $q = \mathcal{P}, \sigma = (r, \hat{\sigma}, m', \pi')$. Finally, parsing $\sigma^* = (r^*, \hat{\sigma}^*, \cdot, \cdot)$, $\mathcal{B}$ outputs $(r^*|m', \hat{\sigma}^*)$.

Note that $\mathcal{B}$'s simulation is perfect. So, by our contradiction assumption, with probability $\epsilon$, $\mathcal{E}$ outputs $w^* = \sigma^*$ such that $(y, w^*) \in \mathcal{R}_{\Sigma_p}$, i.e., $\mathsf{vfy}(\mathsf{vk}, m', \sigma^*) = 1$, that is $\widehat{\Sigma}.\mathsf{vfy}(\widehat{\mathsf{vk}}, r^*|m', \hat{\sigma}^*) = 1$.

We argue that $(r^*|m', \hat{\sigma}^*)$ is a forgery by showing that, with overwhelming probability, the message $r^*|m'$ is different from the two messages $\hat{m}_1 = r|\mathsf{crs}$ and $\hat{m}_2 = r'|m'$ asked by $\mathcal{B}$ to its signing oracle. As one can see, $\Pr[\hat{m}_1 = r^*|m'] \leq 2^{-\lambda}$ as that is the probability that a randomly chosen $m' \in \{0,1\}^\lambda$ hits $\mathsf{crs}$ (even by assuming the unlikely case that $|\mathsf{crs}| = \lambda$).

Second, also notice that $\Pr[\hat{m}_2 = r^*|m'] \leq 2^{-\lambda}$ since that is the probability bound that $r^* = r'$ over the random choice of $r'$. This bound follows by the same argument used in the proof of Lemma 1, which is based on the fact that at least $\lambda$ bits of $r'$ are information theoretically hidden to $\mathcal{E}$. Therefore, $\mathcal{B}$ can break the $\mathsf{UF\text{-}CMA}$ security of $\widehat{\Sigma}$ with non-negligible probability $> \epsilon - 2^{-\lambda+1}$. □

### 4.4 O-SNARKs for signing oracles from SNARKs in the random oracle model

In this section we show that it is possible to "immunize" *any* signature scheme in such a way that any classical SNARK is also an O-SNARK for the signing oracle corresponding to the transformed scheme. The idea is very simple and consists into applying the hash-then-sign approach using a hash function that will be modeled as a random oracle. In addition to holding only in the random oracle model, a limitation of this result is that, since the verification algorithm uses a random oracle, in all those applications where the SNARK is used to prove knowledge of valid signatures, one would need a SNARK for $\mathsf{NP}^{\mathcal{O}}$. Hence, the best one can do is to conjecture that this still works when replacing the random oracle with a suitable hash function.

Let us now state formally our result. To this end, for any signature scheme $\Sigma$ and polynomial $Q(\cdot)$ we define $\mathcal{Z}_{Q,\Sigma}$ as the distribution on tuples $\langle \mathsf{vk}, m_1, \sigma_1, \ldots, m_Q, \sigma_Q \rangle$ obtained by running the following probabilistic algorithm:

$\mathcal{Z}_{Q,\Sigma}(1^\lambda)$
  `let` $Q = Q(\lambda)$
  $(\mathsf{sk}, \mathsf{vk})\leftarrow\mathsf{kg}(1^\lambda)$
  $\tilde{\mathcal{M}} \xleftarrow{\$} \mathsf{MsgSample}(\mathcal{M}, Q)$
  `let` $\tilde{\mathcal{M}} = \{m_1, \ldots, m_Q\}$
  `for` $i = 1$ `to` $Q$ `do` :
    $\sigma_i\leftarrow\mathsf{sign}(\mathsf{sk}, m_i)$
  `return` $\langle \mathsf{vk}, \{m_i, \sigma_i\}_{i=1}^{Q} \rangle$

where $\mathsf{MsgSample}(\mathcal{M}, Q)$ is an algorithm that returns $Q$ distinct messages, each randomly chosen from $\mathcal{M}$.

**Theorem 7.** *Let $\Sigma$ be a $\mathsf{UF\text{-}CMA}$-secure signature scheme, and $\mathcal{H}$ be a family of hash functions modeled as a random oracle. Let $\mathcal{U}_n$ be the uniform distribution over strings of length $n$, and $\mathcal{Z}_{Q,\Sigma}$ be the distribution defined above. Then there exists a signature scheme $\Sigma_{\mathcal{H}}$ such that every*

*$(\mathcal{Z},\mathcal{U},\mathcal{Z}_{\Sigma,Q})$-auxiliary input SNARK $\Pi$ is a $\mathcal{Z}$-auxiliary input O-SNARK for $(\mathbb{O}_{\mathcal{H}},\mathbb{O}_{\Sigma_{\mathcal{H}}})$ where $\mathbb{O}_{\mathcal{H}}$ is a random oracle.*

*Proof.* The proof consists of two steps. First we define the signature scheme $\Sigma_{\mathcal{H}}$ and prove its security from $\Sigma$. Next, we show that $\Pi$ is an O-SNARK for $(\mathbb{O}_{\mathcal{H}},\mathbb{O}_{\Sigma_{\mathcal{H}}})$.

THE SIGNATURE SCHEME $\Sigma_{\mathcal{H}}$. The signature scheme $\Sigma_{\mathcal{H}}$ is essentially an application of the hash-then-sign paradigm to scheme $\Sigma$. The only difference is that, instead of hashing only messages, we also hash a short random string of $\lambda$ bits.

Let $\Sigma = (\Sigma.\mathsf{kg}, \Sigma.\mathsf{sign}, \Sigma.\mathsf{vfy})$ be a signature scheme with message space $\mathcal{M} = \{0,1\}^L$, and let $\mathcal{H}\{H : \{0,1\}^* \to \{0,1\}^L\}$ be a family of hash functions modeled as a random oracle. The signature scheme $\Sigma_{\mathcal{H}} = (\Sigma_{\mathcal{H}}.\mathsf{kg}, \Sigma_{\mathcal{H}}.\mathsf{sign}, \Sigma_{\mathcal{H}}.\mathsf{vfy})$ works as follows:

$\Sigma_{\mathcal{H}}.\mathsf{kg}(1^\lambda)$: Run $(\mathsf{sk},\mathsf{vk}) \leftarrow \Sigma.\mathsf{kg}(1^\lambda)$. Output $(\mathsf{sk},\mathsf{vk})$.
$\Sigma_{\mathcal{H}}.\mathsf{sign}(\mathsf{sk}, m)$: Sample $s \xleftarrow{\$} \{0,1\}^\lambda$, compute $h \leftarrow H(s|m)$, $\hat{\sigma} \leftarrow \Sigma.\mathsf{sign}(\mathsf{sk}, h)$, and output $\sigma = (s, \hat{\sigma})$.
$\Sigma_{\mathcal{H}}.\mathsf{vfy}(\mathsf{vk}, m, \sigma)$: Parse $\sigma = (s, \hat{\sigma})$, compute $h \leftarrow H(s|m)$, return the same output of $\Sigma.\mathsf{vfy}(\mathsf{vk}, h, \hat{\sigma})$.

**Lemma 3.** *If $\Sigma$ is an* UF-CMA*-secure signature scheme, so is $\Sigma_{\mathcal{H}}$ in the random oracle model.*

The security proof of the lemma is straightforward and is omitted.

$\Pi$ IS AN O-SNARK FOR $(\mathbb{O}_{\mathcal{H}},\mathbb{O}_{\Sigma_{\mathcal{H}}})$. We are going to prove[12] that for every non-uniform polynomial-size oracle adversary $\mathcal{A}$ there exists a non-uniform polynomial-size extractor $\mathcal{E}_{\mathcal{A}}$ and a negligible function $\epsilon(\lambda)$ such for every time bound $T$,

$$\Pr[\mathsf{O\text{-}AdPoK}(\lambda, T, \mathcal{A}, \mathcal{E}_{\mathcal{A}}, (\mathbb{O}_{\mathcal{H}}, \mathbb{O}_{\Sigma_{\mathcal{H}}})) \Rightarrow 1] \leq \epsilon(\lambda)$$

As a first step, we show that for every non-uniform polynomial-size algorithm $\mathcal{A}$, making $q$ queries to $\mathcal{O}_{\mathcal{H}}$ and $Q$ queries to $\mathcal{O}_{\Sigma_{\mathcal{H}}}$ (where both $q, Q = \mathsf{poly}(\lambda)$), there is a non-uniform, polynomial-size, non-interactive algorithm $\mathcal{B}$.

$\mathcal{B}$ takes as input $(\mathsf{crs}, o, z)$, where $o \leftarrow \mathcal{U}_{q\cdot L + Q\cdot\lambda}$, $z \leftarrow \mathcal{Z}_{Q,\Sigma}$, and these are parsed as follows: $o = \langle r'_1, \ldots, r'_q, s_1, \ldots, s_Q\rangle$ with $r'_j \in \{0,1\}^L$ and $s_i \in \{0,1\}^\lambda$, $z = \langle \mathsf{vk}, r_1, \hat{\sigma}_1, \ldots, r_Q, \hat{\sigma}_Q\rangle$.

$\mathcal{B}(\mathsf{crs}, o, z)$
  Run $\mathcal{A}^{\mathcal{O}_{\mathcal{H}}, \mathcal{O}_{\Sigma_{\mathcal{H}}}}(\mathsf{crs}) \to (y, \pi)$ and simulate queries to $\mathcal{O}_{\mathcal{H}}$ and $\mathcal{O}_{\Sigma_{\mathcal{H}}}$ as follows:
    Query $\mathcal{O}_{\mathcal{H}}(x_j), j \in \{1, \ldots, q\}$:
      Answer with $\mathcal{O}_{\mathcal{H}}(x_j) = r'_j$ using $r'_j$ from $o$
    Query $\mathcal{O}_{\Sigma_{\mathcal{H}}}(m_i), i \in \{1, \ldots, Q\}$:
      If $\mathcal{O}_{\mathcal{H}}(s_i|m_i)$ has been already answered before, abort.
      Else, set $\mathcal{O}_{\mathcal{H}}(s_i|m_i) = r_i$, and answer $\mathcal{O}_{\Sigma_{\mathcal{H}}}(m_i) = (s_i, \hat{\sigma}_i)$,
        where $r_i, \hat{\sigma}_i$ are taken from $z$ and $s_i$ from $o$.
  Return the same $(y, \pi)$ returned by $\mathcal{A}$.

The simulation provided by $\mathcal{B}$ to $\mathcal{A}$ is perfect unless $\mathcal{B}$ aborts during a signing query. This event – let us formally call it $\mathsf{Abort}$ – happens if there exist $i \in [Q]$ and $j \in [q]$ such that $x_j = s_i|m_i$.

[12] For simplicity we are ignoring the additional auxiliary input distribution $\mathcal{Z}$, as it is easy to see that it essentially "carries over".

However, over the random choice of $s_i \xleftarrow{\$} \{0,1\}^\lambda$ in the $i$-th query (indeed $s_i$ was never used before and is thus hidden to $\mathcal{A}$), we have that

$$\Pr[\mathsf{Abort}] \leq \sum_{i=1}^{Q} \frac{q}{2^\lambda} \leq \frac{q \cdot Q}{2^\lambda}$$

Hence, if $\mathcal{A}$ succeeds in producing an instance-proof pair $(y, \pi)$, so does $\mathcal{B}$ with overwhelming probability.

Then, by the adaptive PoK property we have that for every such $\mathcal{B}$ there is an extractor $\mathcal{E}_\mathcal{B}$ such that for every polynomial $n, Q$

$$\Pr[\mathsf{O\text{-}AdPoK}(\lambda, T, \mathcal{B}, \mathcal{E}_\mathcal{B}, (\mathcal{U}_n, \mathcal{Z}_{Q,\Sigma})) \Rightarrow 1] \leq \epsilon(\lambda)$$

for a negligible $\epsilon$.

So far, we have that for every $\mathcal{A}$ there is an extractor $\mathcal{E}_\mathcal{B}$. In what follows, we use this $\mathcal{E}_\mathcal{B}$ to define the extractor $\mathcal{E}_\mathcal{A}$. The extractor $\mathcal{E}_\mathcal{A}$ takes as input $\mathsf{crs}$ and the transcript

$$\mathsf{qt} = \langle x_1, r'_1, \ldots, x_q, r'_q, r_1, \ldots, r_Q, m_1, s_1, \hat{\sigma}_1, \ldots, m_Q, s_Q, \hat{\sigma}_Q \rangle$$

of queries made by $\mathcal{A}$, and proceeds as follows:

$\mathcal{E}_\mathcal{A}(\mathsf{crs}, \mathsf{qt})$

Run $w \leftarrow \mathcal{E}_\mathcal{B}(\mathsf{crs}, o, z)$

Return the same $w$ returned by $\mathcal{E}_\mathcal{B}$

Basically, it rearranges the data in $\mathsf{qt}$ to fulfill the format of $(o, z)$ from distributions $\mathcal{U}_{qL+Q\lambda}, \mathcal{Z}_{Q,\Sigma}$. It is not hard to see that from the above construction we have

$$\begin{aligned}\Pr[\mathsf{O\text{-}AdPoK}(\lambda, T, \mathcal{A}, \mathcal{E}_\mathcal{A}, (\mathbb{O}_\mathcal{H}, \mathbb{O}_{\Sigma_\mathcal{H}})) \Rightarrow 1] &\leq \Pr[\mathsf{O\text{-}AdPoK}(\lambda, T, \mathcal{B}, \mathcal{E}_\mathcal{B}, (\mathcal{U}_n, \mathcal{Z}_{Q,\Sigma})) \Rightarrow 1] + \frac{qQ}{2^\lambda} \\ &\leq \epsilon(\lambda) + \frac{qQ}{2^\lambda}\end{aligned}$$

□

### 4.5 O-SNARKs for signing oracles from SNARKs

In this section we give a positive result showing that any SNARK $\Pi$ is an O-SNARK for the signing oracle of signature scheme $\Sigma$ if: (i) the message space of $\Sigma$ is appropriately bounded (to be polynomially or at most superpolynomially large); (ii) $\Pi$ tolerates auxiliary input consisting of the public key of $\Sigma$ plus a collection of signatures on randomly chosen messages; (iii) one considers O-SNARK adversaries that query the signing oracle on almost the entire message space. Furthermore, in case of superpolynomially large message spaces, one needs to assume sub-exponential hardness for $\Pi$.

The intuition behind this result is to simulate the O-SNARK adversary by using a (non-interactive) SNARK adversary that receives the public key and a set of signatures on (suitably chosen) messages as its auxiliary input. If these messages exactly match those queried by the O-SNARK adversary, the simulation is perfect. However, since the probability of matching exactly all the $Q = \mathsf{poly}(\lambda)$ queries may decrease exponentially in $Q$ (making the simulation meaningless),

we show how to put proper bounds so that the simulation can succeed with probability depending only on the message space size.

More formally, our result is stated as follows. Let $\Sigma$ be a signature scheme with message space $\mathcal{M}$, and let $Q := Q(\cdot)$ be a function of the security parameter. Let $\mathcal{Z}_{Q,\Sigma}$ be the following auxiliary input distribution

$\mathcal{Z}_{Q,\Sigma}(1^\lambda)$
  $\texttt{let } Q = Q(\lambda)$
  $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{kg}(1^\lambda)$
  $\tilde{\mathcal{M}} \xleftarrow{\$} \mathsf{MsgSample}(\mathcal{M}, Q)$
  $\texttt{let } \tilde{\mathcal{M}} = \{m_1, \ldots, m_Q\}$
  $\texttt{for } i = 1 \texttt{ to } Q \texttt{ do :}$
    $\sigma_i \leftarrow \mathsf{sign}(\mathsf{sk}, m_i)$
  $\texttt{return } \langle \mathsf{vk}, \{m_i, \sigma_i\}_{i=1}^{Q} \rangle$

where $\mathsf{MsgSample}(\mathcal{M}, Q)$ is a probabilistic algorithm that returns a subset $\tilde{\mathcal{M}} \subseteq \mathcal{M}$ of cardinality $Q$ chosen according to some strategy that we discuss later. At this point we only assume a generic strategy such that $\delta(|\mathcal{M}|, Q) = \Pr[\mathsf{MsgSample}(\mathcal{M}, Q) = \mathcal{M}^*]$ for any $\mathcal{M}^* \subseteq \mathcal{M}$ of cardinality $Q$.

**Theorem 8.** *Let $\Sigma$ be a signature scheme with message space $\mathcal{M}$, let $\mathbb{O}_\Sigma$ be the associated family of signing oracles, and let $\mathcal{Z}_{Q,\Sigma}$ be as defined above. If $\Pi$ is a $\mathcal{Z}_{Q,\Sigma}$-auxiliary input SNARK satisfying $(s, t, \epsilon)$-adaptive PoK, then $\Pi$ is an O-SNARK for $\mathbb{O}_\Sigma$ satisfying $(s', t', Q, \epsilon')$-adaptive PoK, where $\epsilon' = \epsilon / \delta(|\mathcal{M}|, Q)$, $s' = s - O(Q \cdot \log |\mathcal{M}|)$, and $t' = t$.*

*Proof.* The basic idea for the proof is to show that for any O-SNARK adversary $\mathcal{A}$ against $\Pi$ there is a non-interactive SNARK adversary $\mathcal{B}_\mathcal{A}$ against $\Pi$ that can perfectly simulate $\mathcal{A}$, provided that the auxiliary input from $\mathcal{Z}_{Q,\Sigma}$ "matches" the queries made by $\mathcal{A}$. More precisely, for any adversary $\mathcal{A}^\mathcal{O}$ making $Q$ oracle queries we define the following adversary $\mathcal{B}_\mathcal{A}$:

$\mathcal{B}_\mathcal{A}(\mathsf{crs}, aux = \langle \mathsf{vk}, \{m_1, \sigma_i\}_{i=1}^{Q} \rangle)$
  Run $(y, \pi) \leftarrow \mathcal{A}^\mathcal{O}(\mathsf{crs})$ and simulate every query $\mathcal{O}(m)$ as follows:
    $\texttt{let } \tilde{\mathcal{M}} = \{m_1, \ldots, m_Q\}$
    $\texttt{if } m = 0 \texttt{ return } \mathsf{vk}$
    $\texttt{else if } m = m_i \in \tilde{\mathcal{M}} \texttt{ return } \sigma_i$
    $\texttt{else return } \perp$
  $\texttt{return } (y, \pi)$

As one can see, the adversary $\mathcal{B}_\mathcal{A}$ fits the syntax of a SNARK adversary in the adaptive proof of knowledge property, by which (if $\Pi$ is a SNARK) there exists an extractor $\mathcal{E}_{\mathcal{B}_\mathcal{A}}$ such that for every time bound $T$ and every $Q$

$$\Pr\left[\mathsf{AdPoK}(\lambda, T, \mathcal{B}_\mathcal{A}, \mathcal{E}_{\mathcal{B}_\mathcal{A}}, \mathcal{Z}_{Q,\Sigma}) \Rightarrow 1\right] \leq \epsilon$$

For every $\mathcal{A}$ we define the extractor $\mathcal{E}_\mathcal{A} = \mathcal{E}_{\mathcal{B}_\mathcal{A}}$, and our goal is to bound

$$\Pr[\mathsf{O\text{-}AdPoK}(\lambda, T, \mathcal{A}, \mathcal{E}_\mathcal{A}, \mathbb{O}_{\mathsf{sign}}) \Rightarrow 1] \leq \epsilon'$$

To prove this we proceed by contradiction. Namely, let us assume that there is $\mathcal{A}$ (and thus corresponding $\mathcal{E}_\mathcal{A} = \mathcal{E}_{\mathcal{B}_\mathcal{A}}$) such that:

$$\Pr[\mathsf{O\text{-}AdPoK}(\lambda, T, \mathcal{A}, \mathcal{E}_{\mathcal{A}}, \mathbb{O}_{\mathsf{sign}}) \Rightarrow 1] \geq \epsilon'$$

We define $\mathsf{G}_1$ to be the following variation of experiment $\mathsf{O\text{-}AdPoK}(\lambda, T, \mathcal{A}, \mathcal{E}_{\mathcal{A}}, \mathbb{O}_{\mathsf{sign}})$: it generates $aux' \leftarrow \mathcal{Z}_{Q,\Sigma}$ (using the same keypair generated when sampling oracle $\mathcal{O} \leftarrow \mathbb{O}_{\mathsf{sign}}$), and at the end $\mathsf{G}_1$ outputs 1 if $\mathsf{qt} = aux'$ holds in addition to $\mathsf{Ver}(\mathsf{vst}, y, \pi) = 1 \wedge (y, w) \notin \mathcal{R}$. Let $\mathsf{QueryMatch}$ be the event that in $\mathsf{G}_1$ it occurs $\mathsf{qt} = aux'$. Clearly, $\Pr[\mathsf{QueryMatch}] = \delta(Q, |\mathcal{M}|)$, and thus

$$\begin{aligned}\Pr[\mathsf{G}_1 \Rightarrow 1] &= \Pr[\mathsf{QueryMatch}] \cdot \Pr[\mathsf{O\text{-}AdPoK}(\lambda, T, \mathcal{A}, \mathcal{E}_{\mathcal{A}}, \mathbb{O}_{\mathsf{sign}}) \Rightarrow 1] \\ &\geq \delta(Q, |\mathcal{M}|) \cdot \epsilon' = \epsilon\end{aligned}$$

To conclude the proof, and reach the contradiction, we show below that

$$\Pr[\mathsf{AdPoK}(\lambda, T, \mathcal{B}_{\mathcal{A}}, \mathcal{E}_{\mathcal{B}_{\mathcal{A}}}, \mathcal{Z}_{Q,\Sigma}) \Rightarrow 1] \geq \Pr[\mathsf{G}_1 \Rightarrow 1]$$

To see this, consider experiment $\mathsf{AdPoK}(\lambda, T, \mathcal{B}_{\mathcal{A}}, \mathcal{E}_{\mathcal{B}_{\mathcal{A}}}, \mathcal{Z}_{Q,\Sigma})$ and let $\mathsf{QueryMatch}'$ be the event that, inside the execution of $\mathcal{B}_{\mathcal{A}}$, all the messages queried by $\mathcal{A}$ coincide with those in $aux$. Since $\mathcal{E}_{\mathcal{A}} = \mathcal{E}_{\mathcal{B}_{\mathcal{A}}}$ one can see that experiment $\mathsf{G}_1$ is basically identical to $\mathsf{AdPoK}(\lambda, T, \mathcal{B}_{\mathcal{A}}, \mathcal{E}_{\mathcal{B}_{\mathcal{A}}}, \mathcal{Z}_{Q,\Sigma}) \Rightarrow 1 \wedge \mathsf{QueryMatch}'$. Namely,

$$\begin{aligned}\Pr[\mathsf{AdPoK}(\lambda, T, \mathcal{B}_{\mathcal{A}}, \mathcal{E}_{\mathcal{B}_{\mathcal{A}}}, \mathcal{Z}_{Q,\Sigma}) \Rightarrow 1] &\geq \Pr[\mathsf{AdPoK}(\lambda, T, \mathcal{B}_{\mathcal{A}}, \mathcal{E}_{\mathcal{B}_{\mathcal{A}}}, \mathcal{Z}_{Q,\Sigma}) \Rightarrow 1 \wedge \mathsf{QueryMatch}'] \\ &= \Pr[\mathsf{G}_1 \Rightarrow 1] \geq \epsilon\end{aligned}$$

which concludes the proof. □

**Implications of Theorem 8.** The statement of Theorem 8 is parametrized by values $|\mathcal{M}|, Q$ and the function $\delta(|\mathcal{M}|, Q)$, which in turn depends on the query guessing strategy. As for the $\mathsf{MsgSample}(\mathcal{M}, Q)$ algorithm, let us consider the one that *samples a random subset* $\tilde{\mathcal{M}} \subseteq \mathcal{M}$ of cardinality $Q$. For this algorithm we have $\delta(|\mathcal{M}|, Q) = \frac{1}{\binom{|\mathcal{M}|}{Q}}$.

Notice that $\delta(|\mathcal{M}|, Q)$ is governing the success probability of our reduction, and thus we would like this function not to become negligible. However, since $Q = \mathsf{poly}(\lambda)$ is a parameter under the choice of the adversary, it might indeed be the case that $\delta(|\mathcal{M}|, Q) \approx 2^{-Q} \approx 2^{-\lambda}$, which would make our reduction meaningless. To avoid this bad case, we restrict our attention to adversaries for which $Q = |\mathcal{M}| - c$ for some constant $c \geq 1$, i.e., adversaries that ask for signatures on the entire message but a constant number of messages. For this choice of $Q$ we indeed have that $\delta(|\mathcal{M}|, Q) = \frac{1}{|\mathcal{M}|^c}$ depends only on the cardinality of $|\mathcal{M}|$. This gives us the following corollary:

**Corollary 1.** *Let $\Sigma$ be a signature scheme with message space $\mathcal{M}$ where $|\mathcal{M}| = \mathsf{poly}(\lambda)$ (resp. $|\mathcal{M}| = \lambda^{\omega(1)}$), and let $Q = |\mathcal{M}| - c$ for constant $c \in \mathbb{N}$. If $\Pi$ is a polynomially (resp. sub-exponentially) secure $\mathcal{Z}_{Q,\Sigma}$-auxiliary input SNARK, then $\Pi$ is an O-SNARK for $\mathbb{O}_{\Sigma}$ (for adversaries making $Q$ queries).*

### 4.6 O-SNARKs for (pseudo)random oracles from SNARKs

In this section we show a positive result on the existence of O-SNARKs for (pseudo)random oracles, based on classical SNARKs that are assumed to satisfy proof of knowledge with respect to randomly-distributed auxiliary input. While we do not have a direct application of this result, we think that it can be useful and of independent interest.

Let $\mathbb{O} = \{\mathcal{O} : \{0,1\}^\ell \to \{0,1\}^L\}$ be a random oracle family, i.e., a family where sampling a member $\mathcal{O} \leftarrow \mathbb{O}$ consists of sampling a suitably long random tape (of size $2^\ell \cdot L$). Let us stress that here, when we refer to a random oracle family, we do not necessarily consider the random oracle model. We simply consider the case of an interactive game in which $\mathcal{A}$ has oracle access to a random function.

**Theorem 9.** *Let $\mathbb{O}$ be a random oracle family. Let $\mathcal{Z}$ be some distribution, $\mathcal{U}_n$ be the uniform distribution over strings of length $n$, and denote by $(\mathcal{Z},\mathcal{U})$ the distribution over pairs $(aux, o)$ such that $aux \leftarrow \mathcal{Z}$ and $o \leftarrow \mathcal{U}$. If $\Pi$ is a $(\mathcal{Z}_q,\mathcal{U})$-auxiliary input SNARK for every $q = \mathsf{poly}(\lambda)$, then $\Pi$ is a $\mathcal{Z}$-auxiliary input O-SNARK for $\mathbb{O}$.*

*Proof.* Completeness and succinctness follow immediately. So, we are left to prove that (adaptive) proof of knowledge implies the corresponding property in the presence of an oracle $\mathcal{O} \leftarrow \mathbb{O}$.

Formally, we have to show that for any efficient oracle prover $\mathcal{A}^\mathcal{O}$ there exists an efficient extractor algorithm $\mathcal{E}_\mathcal{A}$ such that the joint probability that $\mathcal{A}^\mathcal{O}$ outputs a valid proof but $\mathcal{E}_\mathcal{A}$ returns a wrong witness is negligible. At a high level, we do this proof by showing that $\mathcal{E}_\mathcal{A}$'s existence is equivalent to the existence of a SNARK extractor $\bar{\mathcal{E}}$, which is assured under the adaptive proof of knowledge whenever $\mathcal{A}^\mathcal{O}$ exists.

First, for any $\mathcal{A}^\mathcal{O}$ we construct another adversary $\bar{\mathcal{A}}$. Precisely, let $\mathcal{A}$ be a non-uniform algorithm that takes as input $(\mathsf{crs}, aux)$ and (without loss of generality) makes exactly $Q$ queries to $\mathcal{O}$ for some $Q = \mathsf{poly}(\lambda)$. Then, we can define an adversary $\bar{\mathcal{A}}$ that on input $(\mathsf{crs}, \overline{aux})$ – where $\overline{aux} = (aux, o) \leftarrow (\mathcal{Z},\mathcal{U}_{Q \cdot L})$ – works as follows:

$\bar{\mathcal{A}}(\mathsf{crs}, \overline{aux})$
  Run $\mathcal{A}^\mathcal{O}(\mathsf{crs}, aux) \to (y, \pi)$ simulating $\mathcal{O}$ queries as follows:
    given the $i$-th query $\mathcal{O}(q_i)$, answer with the substring
    $\mathcal{O}(q_i) = a_i = o_{L\cdot(i-1)+1} | \cdots | o_{L\cdot i}$
  Return the same $(y, \pi)$ returned by $\mathcal{A}$

The simulation of $\mathcal{O}$ provided by $\bar{\mathcal{A}}$ to $\mathcal{A}$ is clearly perfect. Hence, if $\mathcal{A}^\mathcal{O}$ succeeds in producing an accepting instance-proof pair $(y, \pi)$, so does $\bar{\mathcal{A}}$. Then, by the adaptive PoK property we have that for every such $\bar{\mathcal{A}}$ there is an extractor $\bar{\mathcal{E}}$ that takes the same input of $\bar{\mathcal{A}}$ and outputs a value $w$ such that the probability that $\bar{\mathcal{A}}$ is successful in outputting an accepting pair $(y, \pi)$ and the value $w$ returned by $\bar{\mathcal{E}}$ is a wrong witness (i.e., $(y, w) \notin \mathcal{R}$) is bounded by a negligible function $\epsilon$.

Basically, above we showed that for every $\mathcal{A}^\mathcal{O}$ there exists an extractor $\bar{\mathcal{E}}$. As a last step, we use this $\bar{\mathcal{E}}$ to show the existence of the extractor $\mathcal{E}_\mathcal{A}$.

For simplicity, parse the transcript of queries $\mathsf{qt}$ made by $\mathcal{A}^\mathcal{O}$ as a pair $(\boldsymbol{q}, \boldsymbol{a})$, where $\boldsymbol{q}$ are all the queries and $\boldsymbol{a}$ all the corresponding answers, in order. Then we define the extractor $\mathcal{E}_\mathcal{A}$ in the following way.

$\mathcal{E}_\mathcal{A}(\mathsf{crs}, aux, \mathsf{qt})$
  Run $w \leftarrow \bar{\mathcal{E}}(\mathsf{crs}, \overline{aux})$ where $\overline{aux} = (aux, \boldsymbol{a})$
  Return the same $w$ returned by $\bar{\mathcal{E}}$

It is easy to see that, except for some syntactic changes, $\mathcal{E}_\mathcal{A}$ is the same as $\bar{\mathcal{E}}$. Combining all the above arguments, we have that for every $\mathcal{A}$ there is an extractor $\mathcal{E}_\mathcal{A}$ such that adaptive proof of knowledge for $\mathbb{O}$ holds with probability $\epsilon$. □

**The case of pseudorandom functions** As an interesting corollary of Theorem 9 we show that a similar result holds even for the case in which adversaries get access to a pseudorandom function as an oracle.

**Corollary 2.** *Let* $\mathsf{F} : \{0,1\}^\kappa \times \{0,1\}^\ell \to \{0,1\}^L$ *be a family of pseudorandom functions. Let* $\mathbb{O}_\mathsf{F}$ *be the family of oracles* $\mathcal{O}_\mathsf{F}(x) = \mathsf{F}_K(x)$*, where every family member* $\mathcal{O}$ *is described by a seed* $K$ *of the pseudorandom function, and thus the process* $\mathcal{O}_\mathsf{F} \leftarrow \mathbb{O}_\mathsf{F}$ *corresponds to sampling a random seed* $K \xleftarrow{\$} \{0,1\}^\kappa$*. Let* $\mathcal{Z}$ *be some distribution,* $\mathcal{U}_n$ *be the uniform distribution over strings of length* $n$*, and denote by* $(\mathcal{Z},\mathcal{U})$ *the distribution over pairs* $(aux, o)$ *such that* $aux$ *follows* $\mathcal{Z}$ *and* $o$ *follows* $\mathcal{U}$*.*

*If* $\Pi$ *is a* $(\mathcal{Z}_q,\mathcal{U})$*-auxiliary input adaptive SNARK for every* $q = \mathsf{poly}(\lambda)$*, and* $\mathsf{F}$ *is pseudorandom, then* $\Pi$ *is a* $\mathcal{Z}$*-auxiliary input O-SNARK for* $\mathbb{O}_\mathsf{F}$*.*

*Proof.* The proof of the corollary follows in two steps. The first step relies directly on Theorem 9 to see that $\Pi$ is an O-SNARK for the family $\mathbb{O}$ of random oracles. This means that for every $\mathcal{A}^\mathcal{O}$ there is an extractor $\mathcal{E}_\mathcal{A}$ such that adaptive proof of knowledge for $\mathbb{O}$ holds with probability $\delta$.

The second step consists, informally, in replacing a random oracle with a PRF oracle and then showing that the success probability cannot change too much, unless the function is not pseudorandom.

For every $\tilde{\mathcal{A}}^{\mathcal{O}_\mathsf{F}}$, we can construct another adversary $\mathcal{A}^\mathcal{O}$ that simply runs $\tilde{\mathcal{A}}^{\mathcal{O}_\mathsf{F}}$, simulates all queries using its oracle, and returns the same output of $\tilde{\mathcal{A}}$. By adaptive proof of knowledge for $\mathbb{O}$ there is an extractor $\mathcal{E}_\mathcal{A}$. We then define the extractor $\mathcal{E}_{\tilde{\mathcal{A}}} = \mathcal{E}_\mathcal{A}$.

Let us now consider the adaptive PoK experiment with $\tilde{\mathcal{A}}^{\mathcal{O}_\mathsf{F}}$ and $\mathcal{E}_{\tilde{\mathcal{A}}}$ and let $\tilde{\delta}$ be the probability that the final condition holds. It is easy to prove that by the pseudorandomness of $\mathsf{F}$ there is a negligible function $\nu$ such that $\tilde{\delta} \leq \delta + \nu$. □

## 5 Applications of O-SNARKs

In this section we show three applications of O-SNARKs for building homomorphic signatures [BF11], succinct functional signatures [BGI14], and SNARKs on authenticated data [BBFR15].

Generally speaking, our results show constructions of these primitives based on a signature scheme $\Sigma$ and a succinct non-interactive argument $\Pi$, and show their security by assuming that $\Pi$ is an O-SNARK for signing oracles corresponding to $\Sigma$. Once these results are established, we can essentially reach the following conclusions about the possible secure instantiations of these constructions. First, one can instantiate them by using Micali's CS proofs as O-SNARK (cf. Section 4.1): this solution essentially yields secure instantiations in the random oracle model that work with a specific proof system (perhaps not the most efficient one in practice). Second, one can instantiate them with a classical SNARK and a hash-and-sign signature scheme (cf. Section 4.4), and conjecture that replacing the random oracle with a suitable hash function preserves the overall security. Third, one can instantiate the constructions using a classical SNARK construction $\Pi$ and signature scheme $\Sigma$, and then conjecture that $\Pi$ is an O-SNARK with respect to the family of signing oracles corresponding to $\Sigma$. Compared to the first solution, the last two ones have the advantage that one could use some of the recently proposed efficient SNARK schemes (e.g., [PHGR13, BSCG+13]); on the other hand these solutions have the drawback that the security of the instantiations would be heavily based on a heuristic argument. Finally, a fourth option that we provide are security proofs of these primitives which consider only non-adaptive adversaries (i.e., adversaries that declare all their queries in advance). In this case we can prove security based on non-adaptive O-SNARKs, and

thus based on classical SNARKs (applying our Theorem 4). The advantage of this fourth option is that one obtains a security proof for these instantiations based on classical, not new, assumptions, although the proof holds only for a much weaker security notion.

### 5.1 Homomorphic Signatures

As first application of O-SNARKs we revisit a "folklore" construction of homomorphic signatures from SNARKs. This construction has been mentioned several times in the literature (e.g., [BF11, GW13, CF13, CFW14, GVW15]) and is considered as the 'straightforward' approach for constructing this primitive. In this section we formalize this construction, and notice that its security proof is quite subtle as one actually incurs the extraction issues that we mentioned in the introduction. Namely, one needs to run an extractor in an interactive security game in the presence of a signing oracle. Here we solve this issue by giving a simple proof based on our notion of O-SNARKs (for families of signing oracles).

**Definition of Homomorphic Signatures.** We begin by recalling the definition of homomorphic signatures. The definition below can be seen as the public key version of the notion of homomorphic message authenticators for labeled programs of Gennaro and Wichs [GW13].

Labeled Programs [GW13]. A labeled program consists of a tuple $\mathcal{P} = (F, \tau_1, \dots \tau_n)$ such that $F : \mathcal{M}^n \to \mathcal{M}$ is a function on $n$ variables (e.g., a circuit), and $\tau_i \in \{0,1\}^\ell$ is the label of the $i$-th variable input of $F$. Let $F_{id} : \mathcal{M} \to \mathcal{M}$ be the canonical identity function and $\tau \in \{0,1\}^\ell$ be a label. We consider $\mathcal{I}_\tau = (F_{id}, \tau)$ as the identity program for input label $\tau$. Given $t$ labeled programs $\mathcal{P}_1, \dots \mathcal{P}_t$ and a function $G : \mathcal{M}^t \to \mathcal{M}$, the composed program $\mathcal{P}^*$ is the one obtained by evaluating $G$ on the outputs of $\mathcal{P}_1, \dots \mathcal{P}_t$, and is compactly denoted as $\mathcal{P}^* = G(\mathcal{P}_1, \dots \mathcal{P}_t)$. The labeled inputs of $\mathcal{P}^*$ are all distinct labeled inputs of $\mathcal{P}_1, \dots \mathcal{P}_t$, i.e., all inputs with the same label are grouped together in a single input of the new program.

**Definition 9 (Homomorphic Signatures for Labeled Programs).** *A homomorphic signature scheme* $\mathsf{HomSig}$ *is a tuple of probabilistic, polynomial-time algorithms* $(\mathsf{HomKG}, \mathsf{HomSign}, \mathsf{HomVer}, \mathsf{HomEval})$ *that work as follows*

$\mathsf{HomKG}(1^\lambda)$ *takes a security parameter* $\lambda$ *and outputs a public key* $\mathsf{VK}$ *and a secret key* $\mathsf{SK}$*. The public key* $\mathsf{VK}$ *defines implicitly a message space* $\mathcal{M}$*, the label space* $\mathcal{L}$*, and a set* $\mathcal{F}$ *of admissible functions.*

$\mathsf{HomSign}(\mathsf{SK}, \tau, m)$ *takes a secret key* $\mathsf{SK}$*, a label* $\tau \in \mathcal{L}$ *and a message* $m \in \mathcal{M}$*, and it outputs a signature* $\sigma$*.*

$\mathsf{HomEval}(\mathsf{VK}, F, (\sigma_1, \dots \sigma_n))$ *takes a public key* $\mathsf{VK}$*, a function* $F \in \mathcal{F}$ *and a tuple of signatures* $(\sigma_1, \dots \sigma_n)$*. It outputs a new signature* $\sigma$*.*

$\mathsf{HomVer}(\mathsf{VK}, \mathcal{P}, m, \sigma)$ *takes a public key* $\mathsf{VK}$*, a labeled program* $\mathcal{P} = (F, (\tau_1 \dots \tau_n))$ *with* $F \in \mathcal{F}$*, a message* $m \in \mathcal{M}$*, and a signature* $\sigma$*. It outputs either* 0 *(reject) or* 1 *(accept).*

*and satisfy* authentication correctness*,* evaluation correctness*,* succinctness*, and* security*, as described below.*

- ***Authentication Correctness.*** *Informally, we require that signatures generated by* $\mathsf{HomSign}(\mathsf{SK}, \tau, m)$ *verify correctly for* $m$ *as the output of the identity program* $\mathcal{I} = (F_{id}, \tau)$*. Formally,* $\mathsf{HomSig}$ *has authentication correctness if for all key pairs* $(\mathsf{SK}, \mathsf{VK}) \leftarrow \mathsf{HomKG}(1^\lambda)$*, any label* $\tau \in \mathcal{L}$*, message* $m \in \mathcal{M}$*, and any signature* $\sigma \leftarrow \mathsf{HomSign}(\mathsf{SK}, \tau, m)$*,* $\mathsf{HomVer}(\mathsf{VK}, \mathcal{I} = (F_{id}, \tau), m, \sigma)$ *outputs* 1 *with all but negligible probability.*

– ***Evaluation Correctness.*** *Intuitively, we require that running the evaluation algorithm on signatures* $(\sigma_1, \dots \sigma_n)$*, where* $\sigma_i$ *is a signature for* $m_i$ *on label* $\tau_i$*, produces a signature* $\sigma$ *which verifies for* $F(m_1, \dots m_n)$*. Formally, fix a key pair* $(\mathsf{SK}, \mathsf{VK}) \leftarrow \mathsf{HomKG}(1^\lambda, \mathcal{L})$*, a function* $G : \mathcal{M}^t \to \mathcal{M}$ *and any set of program/message/signature triples* $\{(\mathcal{P}_i, m_i, \sigma_i)\}_{i=1\dots t}$ *such that* $\mathsf{HomVer}(\mathsf{VK}, \mathcal{P}_i, m_i, \sigma_i) = 1$*. If* $m^* = G(m_1 \dots m_t)$*,* $\mathcal{P}^* = G(\mathcal{P}_1, \dots \mathcal{P}_t)$ *and* $\sigma^* = \mathsf{HomEval}(\mathsf{VK}, G, (\sigma_1, \dots, \sigma_t))$*, then* $\mathsf{HomVer}(\mathsf{VK}, \mathcal{P}^*, m^*, \sigma^*) = 1$ *holds with all but negligible probability.*
– ***Succinctness.*** *For every large enough security parameter* $\lambda \in \mathbb{N}$*, there is a polynomial* $p(\cdot)$ *such that for every* $(\mathsf{SK}, \mathsf{VK}) \leftarrow \mathsf{HomKG}(1^\lambda)$ *the output size of* HomSign *and* HomEval *is bounded by* $p(\lambda)$ *for any choice of their inputs.*
– ***Security.*** *A homomorphic signature scheme* HomSig *is secure if for every PPT adversary* $\mathcal{A}$ *there is a negligible function* $\epsilon$ *such that* $\Pr[\mathbf{Exp}^{\mathsf{HomSig\text{-}UF}}_{\mathcal{A},\mathsf{HomSig}}(\lambda) = 1] \leq \epsilon(\lambda)$ *where the experiment* $\mathbf{Exp}^{\mathsf{HomSig\text{-}UF}}_{\mathcal{A},\mathsf{HomSig}}(\lambda)$ *is described in the following:*

  Key generation: *Run* $(\mathsf{VK}, \mathsf{SK}) \leftarrow \mathsf{HomKG}(1^\lambda)$ *and give* VK *to* $\mathcal{A}$*.*

  Signing queries: $\mathcal{A}$ *can adaptively submit queries of the form* $(\tau, m)$*, where* $\tau \in \mathcal{L}$ *and* $m \in \mathcal{M}$*. The challenger initializes an empty list* $T$ *and proceeds as follows:*
    * *If* $(\tau, m)$ *is the first query with label* $\tau$*, then the challenger computes* $\sigma \leftarrow \mathsf{HomSign}(\mathsf{SK}, \tau, m)$*, returns* $\sigma$ *to* $\mathcal{A}$ *and updates the list of queries* $T \leftarrow T \cup \{(\tau, m)\}$*.*
    * *If* $(\tau, m) \in T$ *(i.e., the adversary had already queried the tuple* $(\tau, m)$*), then the challenger replies with the same signature generated before.*
    * *If* $T$ *contains a tuple* $(\tau, m_0)$ *for some different message* $m_0 \neq m$*, then the challenger ignores the query.*

  *Note that each label* $\tau$ *can be queried only once.*

  Forgery: *After the adversary is done with the queries of the previous stage, it outputs a tuple* $(\mathcal{P}^*, m^*, \sigma^*)$*. Finally, the experiment outputs 1 iff the tuple returned by the adversary is a forgery (as defined below).*

  *Forgeries are tuples* $(\mathcal{P}^* = (F^*, (\tau_1^*, \dots \tau_n^*)), m^*, \sigma^*)$ *such that* $\mathsf{HomVer}(\mathsf{VK}, \mathcal{P}^*, m^*, \sigma^*) = 1$ *and they satisfy one the following conditions:*
    * *Type 1 Forgery: There is* $i \in [n]$ *such that* $(\tau_i^*, \cdot) \notin T$ *(i.e., no message* $m$ *has ever been signed w.r.t. label* $\tau_i^*$ *during the experiment).*
    * *Type 2 Forgery:* All *labels* $\tau_i^*$ *have been queried—*$\forall i \in [n], (\tau_i^*, m_i) \in T$*—but* $m^* \neq F^*(m_1, \dots m_n)$ *(i.e.,* $m^*$ *is not the correct output of the labeled program* $\mathcal{P}^*$ *when executed on the previously signed messages*$(m_1, \dots m_n)$*).*

**Context-Hiding.** Another property which is useful for homomorphic signatures is context-hiding. Intuitively, this property says that a signature on the output of a function does not reveal anything about its inputs, beyond what can be trivially learned by the verifier. Here we recall a (statistical) version of the definition proposed in [BF11] (also adapted to our syntax).

**Definition 10 (Weak Context-Hiding).** *A homomorphic signature scheme* HomSig *is weakly context hiding if there exists a PPT simulator* $S = (S^{\mathsf{KG}}, S^{\mathsf{Eval}})$ *such that, for any fixed choice of function* $F$*, tuple of messages* $m_1, \dots, m_n \in \mathcal{M}$*, set of labels* $\tau_1, \dots, \tau_n \in \mathcal{L}$*, it holds that for any distinguisher* $\mathcal{D}$ *and large enough security parameter* $\lambda \in \mathbb{N}$*:*

$$\Pr\left[\mathcal{D}(\mathsf{VK}, \mathsf{SK}, \{\sigma_i\}_{i\in[n]}, \bar{\sigma}) = 1 \;\middle|\; \begin{array}{r}(\mathsf{VK}, \mathsf{SK}) \leftarrow \mathsf{HomKG}(1^\lambda) \\ \sigma_i \leftarrow \mathsf{HomSign}(\mathsf{SK}, \tau_i, m_i) \forall i \in [n] \\ \bar{\sigma} \leftarrow \mathsf{HomEval}(\mathsf{VK}, F, (\sigma_1, \dots \sigma_n))\end{array}\right] -$$

$$\Pr\left[\mathcal{D}(\mathsf{VK},\mathsf{SK},\{\sigma_i\}_{i\in[n]},\bar{\sigma})=1 \;\middle|\; \begin{array}{r}(\mathsf{VK},\mathsf{SK})\leftarrow S^{\mathsf{KG}}(1^\lambda)\\ \sigma_i\leftarrow\mathsf{HomSign}(\mathsf{SK},\tau_i,m_i)\forall i\in[n]\\ \bar{\sigma}\leftarrow S^{\mathsf{Eval}}(\mathcal{P},F(m_1,\ldots,m_n))\end{array}\right]\leq negl(\lambda)$$

Note that since the definition holds for fixed $F,\{m_i\}_i,\{\tau_i\}_i$, these values can also be given to $\mathcal{D}$. The notion is called *weak* context-hiding in contrast to a *strong* notion where one can also hide the fact that a homomorphic evaluation took place.

**Homomorphic Signatures from O-SNARKs.** To build the homomorphic signature we use a regular signature scheme $\Sigma$ (cf. Appendix A.1 for their definition) and a fully-succinct O-SNARK $\Pi$ for $\mathsf{NP}$. The resulting scheme is homomorphic for all functions $F$ whose running time is upper bounded by some fixed polynomial $t_{\mathcal{F}}(\cdot)$, and the scheme is 1-hop, i.e., it is not possible to apply $\mathsf{HomEval}$ on signatures obtained from other executions of $\mathsf{HomEval}$.[13]

Defining the machine $M_{\Sigma,F}$. Let $\Sigma$ be a signature scheme, and $F$ be the description of a function $F:\mathcal{X}^n\to\mathcal{X}$ where $\mathcal{X}$ is some appropriate domain (e.g., $\mathcal{X}=\{0,1\}^\mu$). Then $M_{\Sigma,F}(x,w)$ is the random-access machine that works as follows. It takes inputs $(x,w)$ where values $x$ are of the form $x=(\mathsf{vk},m,\tau_1,\ldots,\tau_n)$ where $\mathsf{vk}$ is a public key of the scheme $\Sigma$, $m\in\mathcal{X}$ is a message and $\tau_i\in\{0,1\}^\ell$ are labels, for $1\leq i\leq n$. The values $w$ are instead tuples $w=(m_1,\sigma_1,\ldots,m_n,\sigma_n)$ where for every $i\in[n]$, $m_i\in\mathcal{X}$ is a message and $\sigma_i$ is a signature of the scheme $\Sigma$. On input such a pair $(x,w)$, $M_{\Sigma,F}(x,w)$ accepts iff

$$m=F(m_1,\ldots,m_n)\ \wedge\ \mathsf{vfy}(\mathsf{vk},\tau_i|m_i,\sigma_i)=1,\forall i=1,\ldots,n$$

Associated to such machine there is also a polynomial time bound $t_{\Sigma,\mathcal{F}}(k)=k^{e_{\Sigma,\mathcal{F}}}$, such that $M_{\Sigma,F}$ rejects if it does more than $t_{\Sigma,F}(|x|)$ steps. Finally, we note that given a polynomial bound $t_{\mathcal{F}}(k)=k^{e_{\mathcal{F}}}$ on the running time of every $F$ supported by the scheme, a polynomial bound $t_\Sigma(k)=k^{e_\Sigma}$ on the running time of $\Sigma$'s verification algorithm, and values $n,\mu,\ell$, one can efficiently deduce the constant exponent $e_{\Sigma,\mathcal{F}}$ for the time bound $t_{\Sigma,\mathcal{F}}(|x|)=|x|^{e_{\Sigma,\mathcal{F}}}$.

We call $\mathcal{R}_\Sigma$ the $\mathsf{NP}$ binary relation consisting of all pairs $(y,w)$ such that, parsing $y=(M_{\Sigma,F},x,t)$, $M_{\Sigma,F}(x,w)$ accepts in at most $t$ steps and $t\leq t_{\Sigma,\mathcal{F}}(|x|)$.

The construction. Let $\Sigma=(\mathsf{kg},\mathsf{sign},\mathsf{vfy})$ be a signature scheme and $\Pi=(\mathsf{Gen},\mathsf{Prove},\mathsf{Ver})$ be a fully-succinct O-SNARK for $\mathsf{NP}$. The homomorphic signature scheme $\mathsf{HomSig}[\Sigma,\Pi]$ is defined as follows.

$\mathsf{HomKG}(1^\lambda)$: Run $(\mathsf{sk},\mathsf{vk})\leftarrow\mathsf{kg}(1^\lambda)$ and $\mathsf{crs}\leftarrow\mathsf{Gen}(1^\lambda)$. Define $\mathsf{SK}=\mathsf{sk}$ and $\mathsf{VK}=(\mathsf{vk},\mathsf{crs})$. Let the message be $\mathcal{M}=\{0,1\}^\mu$ and the label space be $\mathcal{L}=\{0,1\}^\ell$. Output $(\mathsf{SK},\mathsf{VK})$.

$\mathsf{HomSign}(\mathsf{SK},\tau,m)$: Run $\sigma\leftarrow\mathsf{sign}(\mathsf{sk},\tau|m)$. Output $\bar{\sigma}=(\mathit{signature},(\tau,m,\sigma))$.

$\mathsf{HomEval}(\mathsf{VK},m,F,(\bar{\sigma}_1,\ldots,\bar{\sigma}_n))$: Parse every $\bar{\sigma}_i=(\mathit{signature},(\tau_i,m_i,\sigma_i))$, compute $m=F(m_1,\ldots,m_n)$, reconstruct an instance $y=(M_{\Sigma,F},x,t)$ where $x=(\mathsf{vk},m,\tau_1,\ldots,\tau_n)$ and $t=|x|^{e_{\Sigma,\mathcal{F}}}$, and the witness $w=(m_1,\sigma_1,\ldots,m_n,\sigma_n)$. Finally, run $\pi\leftarrow\mathsf{Prove}(\mathsf{crs},y,w)$ and output $\bar{\sigma}=(\mathit{proof},\pi)$.

[13] Previous work hinted the possibility of achieving multi-hop homomorphic signatures by using SNARKs with recursive composition. However, given the issues we already notice in using classical SNARKs, it is at the moment unclear to us whether such a multi-hop construction would allow for a proof.

$\mathsf{HomVer}(\mathsf{VK}, \mathcal{P} = (F, (\tau_1, \ldots \tau_n)), m, \bar{\sigma})$: Parse the signature $\bar{\sigma} = (flag, \cdot)$ and output the bit $b$ computed as follows:
If $\bar{\sigma} = (signature, (\tau, m, \sigma))$ and $\mathcal{P} = \mathcal{I} = (F_{id}, \tau)$ run $\mathsf{vfy}(\mathsf{vk}, \tau|m, \sigma) \to b$.
If $\bar{\sigma} = (proof, \pi)$ run $\mathsf{Ver}_{e_{\Sigma,\mathcal{F}}}(\mathsf{crs}, y, \pi) \to b$ where $y = (M_{\Sigma,F}, x = (\mathsf{vk}, m, \tau_1, \ldots, \tau_n), |x|^{e_{\Sigma,\mathcal{F}}})$.
Recall that in a SNARK for $\mathsf{NP}$, $\mathsf{Ver}_c$ is given a constant $c > 0$ and only works for relation $\mathcal{R}_c$.

In what follows we show that the scheme above is a homomorphic signature.

**Authentication Correctness.** For the present scheme the definition asks that signatures $\bar{\sigma} = (signature, (\tau, m, \sigma))$ as generated by $\mathsf{HomSign}(\mathsf{SK}, \tau, m)$ verify correctly for $m$ as the output of the identity program $\mathcal{I} = (F_{id}, \tau)$. Observe that for all key pairs $(\mathsf{SK}, \mathsf{VK}) \leftarrow \mathsf{HomKG}(1^\lambda)$, any label $\tau \in \mathcal{L}$ and any $\bar{\sigma} \leftarrow \mathsf{HomSign}(\mathsf{SK}, \tau, m)$, the algorithm $\mathsf{HomVer}(\mathsf{VK}, \mathcal{I} = (F_{id}, \tau), \bar{\sigma})$ performs the verification by running $\mathsf{vfy}(\mathsf{vk}, \tau|m, \sigma)$. By the correctness of the signature scheme $\Sigma$, for any signature $\sigma \leftarrow \mathsf{sign}(\mathsf{sk}, \tau|m)$, $\mathsf{vfy}(\mathsf{vk}, \tau|m, \sigma)$ outputs 1 with all but negligible probability.

Therefore authentication correctness of $\mathsf{HomSig}(\Sigma, \Pi)$ follows.

**Evaluation Correctness.** To see evaluation correctness, fix a key pair $(\mathsf{SK}, \mathsf{VK}) \leftarrow \mathsf{HomKG}(1^\lambda)$, and a set of triples of the form $\{(\tau_i, m_i, \bar{\sigma}_i)\}_{i=1\ldots n}$ where $\bar{\sigma}_i = (signature, (\tau_i, m_i, \sigma_i))$ and for which $\mathsf{HomVer}(\mathsf{VK}, \mathcal{I}_i = (F_{id}, \tau_i), m_i, \bar{\sigma}_i) = 1$.
Let $\bar{\sigma}^* \leftarrow \mathsf{HomEval}(\mathsf{VK}, F^*, (\bar{\sigma}_1, \ldots, \bar{\sigma}_n))$, where $\bar{\sigma}^* = (proof, \pi)$ with $\pi$ a proof generated by $\mathsf{Prove}(\mathsf{prs}, y, w)$ by setting $y = (M_{\Sigma,F}, x = (\mathsf{vk}, m^*, \tau_1, \ldots, \tau_n), |x|^{e_{\Sigma,\mathcal{F}}})$ and $w = (m_1, \sigma_1, \ldots, m_n, \sigma_n)$.

If $m^* = F^*(m_1 \ldots m_n)$ (for an $F^*$ that is in the class of supported functions) and since $\mathsf{HomVer}(\mathsf{VK}, \mathcal{I}_i, m_i, \bar{\sigma}_i)$ means $\mathsf{vfy}(\mathsf{vk}, \tau_i|m_i, \sigma_i) = 1$, we have that $(y, w) \in \mathcal{R}_\Sigma$. Thus $\mathsf{HomVer}(\mathsf{VK}, \mathcal{P}^*, m^*, \bar{\sigma}^*) = 1$ holds with all but negligible probability by the correctness of the SNARK $\mathsf{Ver}_{e_{\Sigma,\mathcal{F}}}(\mathsf{crs}, y, \pi) = 1$.

**Succinctness.** As the output of the $\mathsf{HomEval}$ algorithm is a proof $\pi$ obtained by running the $\mathsf{Prove}$ algorithm, the succinctness of $\mathsf{HomSign}(\Sigma, \Pi)$ follows from the succinctness of $\Pi$.

**Security.** As in Section 4.3, for every signature scheme $\Sigma = (\mathsf{kg}, \mathsf{sign}, \mathsf{vfy})$ we denote by $\mathbb{O}_\Sigma$ the family of oracles $\mathcal{O}(m) = \mathsf{sign}(\mathsf{sk}, m)$ (where the verification key is returned as output of a special query $\mathcal{O}(\text{`}vk')$).

We show the security of the scheme $\mathsf{HomSig}[\Sigma, \Pi]$ via the following theorem.

**Theorem 10.** *Let $\Sigma$ be a signature scheme, and $\mathbb{O}_\Sigma$ be its corresponding signing oracle family as described above. If $\Pi$ is an O-SNARK for $\mathbb{O}_\Sigma$, and $\Sigma$ is* UF-CMA*-secure, then* $\mathsf{HomSig}[\Sigma, \Pi]$ *is a secure homomorphic signature scheme.*

*Proof.* To prove that this scheme is a secure homomorphic signature, we assume by contradiction the existence of an efficient adversary $\mathcal{A}$ that is able to output a forgery of one of the two types with non-negligible probability $\delta$. Starting from this algorithm $\mathcal{A}$, we further construct a successful forger $\mathcal{B}$ against the UF-CMA-security of $\Sigma$, which leads to a contradiction. Along the way of this reduction, we also rely on the fact that $\Pi$ is an O-SNARK satisfying adaptive proof of knowledge for $\mathbb{O}_\Sigma$.

By looking at the definition of the security experiment $\mathbf{Exp}^{\mathsf{HomSig\text{-}UF}}_{\mathcal{A},\mathsf{HomSig}}(\lambda)$ for the $\mathsf{HomSig}$ scheme, adversary $\mathcal{A}$ is almost equivalent to an O-SNARK adversary $\tilde{\mathcal{A}}^{\mathcal{O}}$ for oracle $\mathcal{O} \leftarrow \mathbb{O}_{\mathsf{sign}}$. Stated more formally, for every adversary $\mathcal{A}$ against $\mathsf{HomSig}$ that outputs a forgery $\mathcal{P}^* = (F^*, (\tau_1^*, \ldots \tau_n^*)), m^*, \bar{\sigma}^* = (proof, \pi)$, it is possible to construct another adversary $\tilde{\mathcal{A}}^{\mathcal{O}}(\mathsf{crs})$ working as follows: it queries

$\mathsf{vk} \leftarrow \mathcal{O}(\text{`}vk')$; it runs $\mathcal{A}^{\mathsf{HomSign}(\mathsf{SK},\cdot)}(\mathsf{VK} = (\mathsf{vk}, \mathsf{crs}))$ simulating $\mathcal{A}$'s oracle queries using its own oracle $\mathcal{O}$; finally it returns the value $(y, \pi)$, where $y = (M_{\Sigma,F^*}, x, |x|^{e_{\Sigma,\mathcal{F}}})$, with $x = (\mathsf{vk}, m^*, \tau_1^*, \ldots, \tau_n^*)$, is obtained from $\mathcal{A}$'s output. The adversary $\tilde{\mathcal{A}}$ perfectly fits the O-SNARK definition by which we know that there exists an extractor $\mathcal{E}_{\tilde{\mathcal{A}}}$ that, given the same input of $\tilde{\mathcal{A}}^{\mathcal{O}}$ and the transcript of oracle queries/answers made and received by $\tilde{\mathcal{A}}^{\mathcal{O}}$, outputs a correct witness $w$ (i.e., such that $(y, w) \in \mathcal{R}_\Sigma$) with all but negligible probability.

Hence, we have that for every successful adversary $\mathcal{A}$ against $\mathsf{HomSig}$ there exists extractor $\mathcal{E}_{\tilde{\mathcal{A}}}$, that takes the very same input of $\mathcal{A}$ (plus, the list of oracle answers). Starting from this adversary $\mathcal{A}$, we construct the forger $\mathcal{B}^{\mathcal{O}}$ that breaks the $\mathsf{UF\text{-}CMA}$ security of $\Sigma$. We build $\mathcal{B}$ (which gets the public key $\mathsf{vk}$ and can make queries to $\mathcal{O} = \mathsf{sign}(\mathsf{sk}, \cdot)$ oracle) as follows:

$\mathcal{B}^{\mathcal{O}}(\mathsf{vk})$ :

Initialize $\mathsf{qt} \leftarrow (\text{`}vk', \mathsf{vk})$
Generate $\mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda)$ and run $\mathcal{A}(\mathsf{VK} = (\mathsf{crs}, \mathsf{vk}))$
Simulate queries $(\tau, m)$ to $\mathcal{O}$ as follows:
  query $\sigma \leftarrow \mathcal{O}(\tau|m)$ and add $(\tau|m, \sigma)$ to $\mathsf{qt}$
  output $\sigma$
When $\mathcal{A}$ outputs $(\mathcal{P}^* = (F^*, (\tau_1^*, \ldots \tau_n^*)), m^*, \bar{\sigma}^*)$ parse $\bar{\sigma}^* = (proof, \pi^*)$
Run $\mathcal{E}_{\tilde{\mathcal{A}}}(\mathsf{crs}, \mathsf{qt})$ and obtain the witness $w = (m_1^*, \sigma_1^*, \ldots, m_n^*, \sigma_n^*)$
Check that $(y, w) \in \mathcal{R}_\Sigma$, i.e., that $M_\Sigma(x, w) = 1$ in at most $|x|^{e_{\Sigma,\mathcal{F}}}$ steps
where $M_\Sigma(x, w) := (m^* = F(m_1^*, \ldots, m_n^*) \ \wedge \ \mathsf{vfy}(\mathsf{vk}, \tau_i^*|m_i^*, \sigma_i^*) = 1, \forall i \in [n])$
  [ Fail ]  Abort if $(y, w) \notin \mathcal{R}_\Sigma$.
Else proceed:
  [Type 1]  If $\exists j \in [n]$ such that $(\tau_j^*|\cdot, \cdot) \notin \mathsf{qt}$ return $(\tau_j^*|m_j^*, \sigma_j^*)$
  [Type 2]  If $\forall i \in [n]$ there is a tuple $(\tau_i^*|m_i, \sigma_i) \in \mathsf{qt}$
  and there $\exists j \in [n]$ such that $m_j^* \neq m_j$ return $(\tau_j^*|m_j^*, \sigma_j^*)$

Let us now show that whenever $\mathcal{A}$ succeeds in the simulation described above, $\mathcal{B}$ is also successful in breaking the $\mathsf{UF\text{-}CMA}$ security of $\Sigma$, unless the "Fail" event happens.

First, let us condition on the event that $\mathcal{B}$ does not abort, i.e., "Fail" does not occur, that is the extractor $\mathcal{E}_{\tilde{\mathcal{A}}}$ is correct in returning a valid witness. If $\mathcal{A}$ outputs the first type of forgery, since the witness $w$ is valid— $\mathsf{vfy}(\mathsf{vk}, \tau_i^*|m_i^*, \sigma_i^*) = 1, \forall i = 1, \ldots, n$— it follows that the signature forgery $(\tau_j^*|m_j^*, \sigma_j^*)$ is a valid one.

If the forgery returned by $\mathcal{A}$ is of the second type, recall this means that $m^* \neq F(m_1, \ldots, m_n)$ where all inputs of $F$ are the ones in the transcript $\mathsf{qt}$, i.e., $\mathsf{qt}$ contains the tuples $(\tau_1^*|m_1, \sigma_1), \ldots (\tau_n^*|m_n, \sigma_n)$. Combining this with the validity of $w$ we have that $F(m_1^*, \ldots, m_n^*) = m^* \neq F(m_1, \ldots, m_n)$, and thus $(m_1^*, \ldots, m_n^*) \neq (m_1, \ldots, m_n)$. This means that there exists at least one $j$ such that $(\tau_j^*|m_j, \sigma) \in \mathsf{qt}$ but $m_j^* \neq m_j$. Moreover, by witness validity it also holds $\mathsf{vfy}(\mathsf{vk}, \tau_j^*|m_j^*, \sigma_j^*) = 1$. Thus we can conclude that, even in this case, $(\tau_j^*|m_j^*, \sigma_j^*)$ is a valid forgery for $\Sigma$.

So far, we have proved that whenever the adversary $\mathcal{A}$ is able to output a valid forgery and $\mathcal{B}$ does not abort, then $\mathcal{B}$ is a successful adversary against the $\mathsf{UF\text{-}CMA}$ security of $\Sigma$. However, whenever $\mathcal{A}$ is successful (with non-negligible probability), by the O-SNARK definition we have that "Fail" occurs with negligible probability at most $\epsilon$. Therefore, if $\mathcal{A}$ is successful with probability at least $\delta$, then $\mathcal{B}$ is successful with non-negligible probability $\geq \delta - \epsilon$. □

**Non-adaptive security.** Alternatively, one can modify the previous proof to show that the scheme has security against homomorphic signature adversaries that make non-adaptive signing queries, as-

suming the weaker assumption that $\Pi$ is a *non-adaptive* O-SNARK (see Definition 8). In particular, combining this change with the result of Theorem 4 one obtains the following:

**Theorem 11.** *If $\Pi$ is a SNARK, and $\Sigma$ is a* $\mathsf{UF\text{-}CMA}$*-secure signature scheme, then* $\mathsf{HomSig}[\Sigma, \Pi]$ *is a* non-adaptive *secure homomorphic signature scheme.*

*Proof.* The proof would be very similar to that of Theorem 10 and thus we only sketch the main changes to apply to that proof. To work with non-adaptive adversaries the only main change is that for every non-adaptive adversary $\mathcal{A}$ one can define a corresponding non-adaptive O-SNARK adversary $\tilde{\mathcal{A}}$. Then the only difference is that the non-adaptive queries of $\mathcal{A}$ can be used to define the non-adaptive queries of $\tilde{\mathcal{A}}$. The rest of the proof proceeds analogously.

*Remark 4 (On the applicability of Corollary 1).* Finally, we note that we cannot apply the positive result of Corollary 1 to our result of Theorem 10 to conclude that the security of the homomorphic signature scheme holds under classical SNARKs. The inapplicability of Corollary 1 is due to its restriction for which adversaries have to query almost the entire message space. Indeed, by looking at the construction of $\mathsf{HomSig}$ (and the definition of homomorphic signatures too) it is possible to note that an adversary who queries almost the entire message space of the underlying signature scheme can trivially break the security (for example he could obtain signatures on two distinct messages under the same label).

**Insecurity of $\mathsf{HomSig}[\Sigma_p, \Pi]$.** Here we show that the signature scheme $\Sigma_p$ described in Section 4.3 can be slightly modified in such a way that, not only extraction is impossible in the presence of its signing oracle, but also that it can directly make our homomorphic signature construction insecure.

Note that this insecurity result does not contradict our Theorem 10, but is closely related with (and confirms) our impossibility of O-SNARKs for any signing oracles (Theorem 6).

Consider the $\mathsf{HomSig}$ construction in which messages are bits (i.e., $\mu = 1$) and labels are strings of length $\ell = \mathsf{poly}(\lambda)$. Then let $\Sigma^*$ be the same as $\Sigma_p$ except that the signing algorithm works as follows:

$\mathsf{sign}(\mathsf{sk}, m)$**:** Signing works as follows

- sample $r \xleftarrow{\$} \{0,1\}^\delta$, and compute $\hat{\sigma} \leftarrow \widehat{\Sigma}.\mathsf{sign}(\widehat{\mathsf{sk}}, r|m)$;
- sample $m' \xleftarrow{\$} \{0,1\}^{\ell+1}$, $r' \xleftarrow{\$} \{0,1\}^\delta$, compute $\hat{\sigma}' \leftarrow \widehat{\Sigma}.\mathsf{sign}(\widehat{\mathsf{sk}}, r'|m')$, and set $\sigma' = (r', \hat{\sigma}', 0, 0)$;
- parse $m' \in \{0,1\}^\ell$ bit by bit as $(m'_1|\cdots|m'_\ell)$, set $\tau = m_1|\cdots|m_\ell$, $x = (\mathsf{vk}, m'_{\ell+1}, \tau)$, $w = (m'_{\ell+1}, \sigma')$, and let $\mathcal{I} : \{0,1\} \to \{0,1\}$ be the identity function;
- let $t = \#M_{\Sigma^*,\mathcal{I}}(x, w)$, and set $y = (M_{\Sigma^*,\mathcal{I}}, x, t)$;
- Interpret $m$ as the description of program $\mathcal{P}(\cdot,\cdot)$ and thus run $\pi \leftarrow \mathcal{P}(y, w)$;
- if $|\pi| > p(\lambda)$ set $\pi' = 0$ and $y = 0$, else $\pi' = \pi$.
- output $\sigma = (r, \hat{\sigma}, m', \pi')$.

The main changes are the sampling of $m' \xleftarrow{\$} \{0,1\}^{\ell+1}$ (instead of $m' \xleftarrow{\$} \{0,1\}^\lambda$), which is now interpreted as a $\ell$-bits long label $\tau$, and a one-bit message $m'_{\ell+1}$, i.e., $m' = (\tau, m'_{\ell+1})$. Then, the theorem-proof $(y, w)$ is now intended for the relation $\mathcal{R}_{\Sigma^*}$ with respect to the identity function $\mathcal{I}$. In particular, recall that the definition of $M_{\Sigma^*,\mathcal{I}}(x, w)$ from Section 5.1: this is the machine that on inputs $x = (\mathsf{vk}, m, \tau)$, where $\mathsf{vk}$ is a public key of the scheme $\Sigma^*$, $m$ is a bit and $\tau \in \{0,1\}^\ell$ and $w = (m^*, \sigma^*)$ accepts iff

$$m = \mathcal{I}(m^*) \ \wedge \ \mathsf{vfy}(\mathsf{vk}, \tau|m, \sigma^*) = 1$$

Below we show an adversary $\mathcal{A}$ that has non-negligible probability of winning in $\mathbf{Exp}^{\mathsf{HomSig\text{-}UF}}_{\mathcal{A},\mathsf{HomSig}[\Sigma^*,\Pi]}(\lambda)$. Our adversary wins by producing a Type 1 forgery with respect to an identity function. For ease of exposition, we give the following security experiment which is a specialization of $\mathbf{Exp}^{\mathsf{HomSig\text{-}UF}}$ to adversaries that only output Type 1 forgeries for identity functions:

Experiment $\mathbf{Exp}^{\mathsf{Type\text{-}1},Id}_{\mathcal{A},\mathsf{HomSig}[\Sigma,\Pi]}(\lambda)$

$(\mathsf{SK},\mathsf{VK}) \xleftarrow{\$} \mathsf{HomKG}(1^\lambda)$
$(\mathcal{I}^* = (F_{id},\tau^*), m^*, \bar{\sigma}^*) \xleftarrow{\$} \mathcal{A}^{\mathsf{HomSign}(\mathsf{SK},\cdot)}(\mathsf{VK})$
If $\mathsf{HomVer}(\mathsf{VK},\mathcal{I}^*,m^*,\bar{\sigma}^*) = 1$ and $\tau^*$ is "new", output 1
Else output 0

**Lemma 4.** *Let $\Pi$ be an O-SNARK and $\Sigma^*$ be the* UF-CMA*-secure signature scheme defined above. Then there exists an efficient adversary $\mathcal{A}$ such that*

$$\Pr[\mathbf{Exp}^{\mathsf{Type\text{-}1},Id}_{\mathcal{A},\mathsf{HomSig}[\Sigma^*,\Pi]}(\lambda) = 1] = 1 - negl(\lambda)$$

.

*Proof.* Below is the description of our adversary $\mathcal{A}$:

Adversary $\mathcal{A}^{\mathsf{HomSign}(\mathsf{SK},\cdot)}(\mathsf{VK})$

1 Query the signing oracle on $(\tau, 0)$ with $\tau = \mathsf{P}$
where $\mathsf{P}$ is the description of $\mathsf{Prove}_{\mathsf{prs}}(\cdot,\cdot)$
2 Parse the answer $\sigma = (r, \hat{\sigma}, m', \pi')$ and $m' = \tau^* | m^*$
3 Return $(\mathcal{I}^* = (F_{id},\tau^*), m^*, \bar{\sigma} = (proof, \pi'))$

Note that, except with probability $2^{-\ell}$, it holds $\tau^* \neq \tau$. Moreover, by the correctness of $\Pi$ (and its succinctness), the answer to $\mathcal{A}$'s oracle query contains a valid proof $\pi'$ that verifies for the identity function with message $m^*$ and label $\tau^*$. In other words the output of $\mathcal{A}$ constitutes a Type-1 forgery with probability $1 - 2^{-\ell}$. □

*Remark 5.* We interpret the above counterexample as saying that the issue with proving the security of this construction is not a mere difficulty in doing the proof, but that the construction can actually become insecure, if one simply relies on *any* signature scheme.

**Context-Hiding of** HomSig**.**

**Theorem 12.** *If $\Pi$ is a zero-knowledge O-SNARK then* HomSig *is weakly context-hiding.*

*Proof.* To show weakly context-hiding for $\mathsf{HomSig}[\Sigma,\Pi]$ we define a simulator $S^{\mathsf{Hide}} = (S^{\mathsf{KG}}, S^{\mathsf{Eval}})$. For this purpose we use the PPT simulator $S^{\Pi} = (S^{\mathsf{crs}}, S^{\mathsf{Prove}})$ from the zero-knowledge of $\Pi$.

$S^{\mathsf{KG}}(1^\lambda)$ :
Run $(\mathsf{vk},\mathsf{sk}) \leftarrow \mathsf{kg}(1^\lambda)$
Run $(\mathsf{prs},\mathsf{vst},\mathsf{tr}) \leftarrow S^{\mathsf{crs}}(1^\lambda, T)$
Set $\mathsf{SK} = (\mathsf{sk},\mathsf{prs})$ and $\mathsf{VK} = (\mathsf{vk},\mathsf{vst})$
Output $(\mathsf{VK},\mathsf{SK})$

$S^{\mathsf{Eval}}(\mathcal{P} = (F,\tau_1,\ldots\tau_n)), m^*)$ :
Set $y = (M_{\Sigma,F}, x = (\mathsf{vk}, m^*, \tau_1 \ldots \tau_n), |x|^{e_{\Sigma,\mathcal{F}}})$
Run $\pi \leftarrow S^{\mathsf{Prove}}(\mathsf{prs},\mathsf{tr},y)$
Output $\bar{\sigma} = (proof, \pi)$

For any distinguisher $\mathcal{D}^{\mathsf{Hide}}$ against the weakly context-hiding of $\mathsf{HomSign}[\Sigma, \Pi]$, we can easily construct a distinguisher $\mathcal{D}^{\Pi}$ against zero knowledge O-SNARK property:

$\mathcal{D}^{\Pi}(\mathsf{crs})$ :
Generate a pair $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{kg}(1^\lambda)$
Set $\mathsf{SK} = (\mathsf{sk}, \mathsf{prs})$ and $\mathsf{VK} = (\mathsf{vk}, \mathsf{vst})$
Let $F, m_1, \ldots, m_n, \tau_1 \ldots \tau_n$ be the fixed tuple:
Run $\sigma_i \leftarrow \mathsf{HomSign}(\mathsf{sk}, \tau_i, m_i)\ \forall i \in [n]$
Output $(y, w)$ to its challenger, and get back $\pi$
(where $y = (M_{\Sigma,F}, x = (\mathsf{vk}, m^* = F(m_1, \ldots, m_n), \tau_1 \ldots \tau_n), |x|^{e_{\Sigma,\mathcal{F}}})$
and $w = (m_1, \sigma_1, \ldots, m_n, \sigma_n)$)
run $b \leftarrow \mathcal{D}^{\mathsf{Hide}}(\mathsf{VK}, \mathsf{SK}, \sigma_1, \ldots, \sigma_n, \tilde{\sigma} = (proof, \pi))$
Output $b$.

Clearly, if $\mathcal{D}^{\Pi}$ receives $\pi$ and $\mathsf{crs}$ generated using the real algorithms, it simulates the real game to $\mathcal{D}^{\mathsf{Hide}}$. Otherwise, if these values are generated through the zero-knowledge simulator, then the view of $\mathcal{D}^{\mathsf{Hide}}$ is identical to the case where it receives values generated using our context hiding simulator described above. Therefore, the distinguishing advantage of $\mathcal{D}^{\Pi}$ in distinguishing between a real or a simulated proof is the same as that of the algorithm $\mathcal{D}^{\mathsf{Hide}}$ in distinguish the two distributions in the answers from the context-hiding definition.

### 5.2 Succinct Functional Signatures

As second application of O-SNARKs we revisit the construction of succinct functional signatures of Boyle, Goldwasser, and Ivan [BGI14]. In [BGI14] this construction is proven secure using a notion of SNARKs which significantly differs from the standard one [BCC$^+$14] (for completeness we recall this notion in Appendix B). To the best of our knowledge, there are no known instantiations of SNARKs under this definition, in the standard model (and is not clear whether it is possible to find some). On the other hand, if one wants to prove the security of this construction using the classical SNARK definition, the security proof incurs the same subtleties related to running an extractor in the presence of a signing oracle.

In this section, we revisit the construction of [BGI14], and we prove its security using O-SNARKs. Interestingly, this proof differs a little from the one of homomorphic signature as here we have to consider O-SNARKs for *multiple* signing oracles.

**Definition of Functional Signatures.** First, we recall the definition of (succinct) functional signatures as provided in [BGI14].

**Definition 11 (Functional Signatures [BGI14]).** *A functional signature scheme* $\mathsf{FS}$ *for a message space* $\mathcal{M}$ *and function family* $\mathcal{F} = \{f : \mathcal{D}_f \to \mathcal{M}\}$ *is a tuple of probabilistic, polynomial-time algorithms* $(\mathsf{FS.Setup}, \mathsf{FS.KeyGen}, \mathsf{FS.Sign}, \mathsf{FS.Ver})$ *that work as follows*

$\mathsf{FS.Setup}(1^\lambda)$ *takes a security parameter* $\lambda$ *and outputs a master verification key* $\mathsf{mvk}$ *and a master secret key* $\mathsf{msk}$.

$\mathsf{FS.KeyGen}(\mathsf{msk}, f)$ *takes the master secret key* $\mathsf{msk}$ *and a function* $f \in \mathcal{F}$ *(represented as a circuit) and it outputs a signing key* $\mathsf{sk}_f$ *for* $f$.

$\mathsf{FS.Sign}(\mathsf{mvk}, f, \mathsf{sk}_f, m)$ *takes as input a function* $f \in \mathcal{F}$*, a signing key* $\mathsf{sk}_f$*, and a message* $m \in \mathcal{D}_f$*, and it outputs* $(f(m), \sigma)$*where* $\sigma$ *represents a signature on* $f(m)$.

$\mathsf{FS.Ver}(\mathsf{mvk}, m^*, \sigma)$ *takes as input the master verification key* $\mathsf{mvk}$*, a message* $m^* \in \mathcal{M}$ *and a signature* $\sigma$*, and outputs either* 1 *(accept) or* 0 *(reject).*

*and satisfy* correctness*,* unforgeability*, and* function privacy *as described below.*

- ***Correctness.*** *A functional signature scheme is correct if the following holds with probability 1:*

$$\forall f \in \mathcal{F},\ \forall m \in \mathcal{D}_f,\ (\mathsf{msk}, \mathsf{mvk}) \leftarrow \mathsf{FS.Setup}(1^\lambda),\ \mathsf{sk}_f \leftarrow \mathsf{FS.KeyGen}(\mathsf{msk}, f),$$

$$(m^*, \sigma) \leftarrow \mathsf{FS.Sign}(\mathsf{mvk}, f, \mathsf{sk}_f, m), \mathsf{FS.Ver}(\mathsf{mvk}, m^*, \sigma) = 1$$

- ***Unforgeablity.*** *A functional signature scheme is unforgeable if for every PPT adversary* $\mathcal{A}$ *there is a negligible function* $\epsilon$ *such that* $\Pr[\mathbf{Exp}^{\mathsf{FS\text{-}UF}}_{\mathcal{A},\mathsf{FS}}(\lambda) = 1] \leq \epsilon(\lambda)$ *where the experiment* $\mathbf{Exp}^{\mathsf{FS\text{-}UF}}_{\mathcal{A},\mathsf{FS}}(\lambda)$ *is described in the following:*
  Key generation: *Generate* $(\mathsf{msk}, \mathsf{mvk}) \leftarrow \mathsf{FS.Setup}(1^\lambda)$*, and gives* $\mathsf{mvk}$ *to* $\mathcal{A}$*.*
  Queries: *The adversary is allowed to adaptively query a key generation oracle* $\mathcal{O}_{\mathsf{key}}$ *and a signing oracle* $\mathcal{O}_{\mathsf{sign}}$*, that share a dictionary* $D$ *indexed by tuples* $(f, i) \in \mathcal{F} \times \mathbb{N}$*, whose entries are signing keys. For answering these queries, the challenger proceeds as follows:*
  - $\mathcal{O}_{\mathsf{key}}$ $(f, i)$*:*
    - *If* $(f, i) \in D$ *(i.e., the adversary had already queried the tuple* $(f, i)$*), then the challenger replies with the same key* $\mathsf{sk}^i_f$ *generated before.*
    - *Otherwise, generate a new* $\mathsf{sk}^i_f \leftarrow \mathsf{FS.KeyGen}(\mathsf{msk}, f)$*, add the entry* $(f, i) \rightarrow \mathsf{sk}^i_f$ *in* $D$*, and return* $\mathsf{sk}^i_f$*.*
  - $\mathcal{O}_{\mathsf{sign}}$ $(f, i, m)$*:*
    - *If there is an entry for the key* $(f, i)$ *in* $D$*, then the challenger generates a signature on* $f(m)$ *using this key, i.e.,* $\sigma \leftarrow \mathsf{FS.Sign}(\mathsf{mvk}, f, \mathsf{sk}^i_f, m)$*.*
    - *Otherwise, generate a new key* $\mathsf{sk}^i_f \leftarrow \mathsf{FS.KeyGen}(\mathsf{msk}, f)$*, add an entry* $(f, i) \rightarrow \mathsf{sk}^i_f$ *to* $D$*, and generate a signature on* $f(m)$ *using this key, i.e.,* $\sigma \leftarrow \mathsf{FS.Sign}(\mathsf{mvk}, f, \mathsf{sk}^i_f, m)$*.*

  Forgery: *After the adversary is done with its queries, it outputs a pair* $(m^*, \sigma)$*, and the experiment outputs 1 iff the following conditions hold*
  - $\mathsf{FS.Ver}(\mathsf{mvk}, m^*, \sigma) = 1$*.*
  - *there does not exist* $m$ *such that* $m^* = f(m)$ *for any* $f$ *which was sent as a query to the* $\mathcal{O}_{\mathsf{key}}$ *oracle.*
  - *there does not exist a pair* $(f, m)$ *such that* $(f, m)$ *was a query to the* $\mathcal{O}_{\mathsf{sign}}$ *oracle and* $m^* = f(m)$*.*
- ***Function privacy.*** *Intuitively, function privacy requires that the distribution of signatures on a message* $m$ *that are generated via different keys* $\mathsf{sk}_f$ *should be computationally indistinguishable, even given the secret keys and master signing key. More formally, a functional signature scheme has function privacy if for every PPT adversary* $\mathcal{A}$ *there is a negligible function* $\nu$ *such that* $\Pr[\mathbf{Exp}^{\mathsf{FS\text{-}FPri}}_{\mathcal{A},\mathsf{FS}}(\lambda) = 1] \leq \nu(\lambda)$ *where experiment* $\mathbf{Exp}^{\mathsf{FS\text{-}FPri}}_{\mathcal{A},\mathsf{FS}}(\lambda)$ *works as follows:*
  - *The challenger generates a key pair* $(\mathsf{mvk}, \mathsf{msk}) \leftarrow \mathsf{FS.Setup}(1^\lambda)$ *and gives* $(\mathsf{mvk}, \mathsf{msk})$ *to* $\mathcal{A}$*.*
  - *The adversary chooses a function* $f_0$ *and receives an (honestly generated) secret key* $\mathsf{sk}_{f_0} \leftarrow \mathsf{FS.KeyGen}(\mathsf{msk}, f_0)$*.*
  - *The adversary chooses a second function* $f_1$ *such that* $|f_0| = |f_1|$ *(where padding can be used if there is a known upper bound) and receives an (honestly generated) secret key* $\mathsf{sk}_{f_1} \leftarrow \mathsf{FS.KeyGen}(\mathsf{msk}, f_1)$*.*
  - *The adversary chooses a pair of values* $(m_0, m_1)$ *such that* $|m_0| = |m_1|$ *and* $f_0(m_0) = f_1(m_1)$*.*

- *The challenger selects a random bit* $b \leftarrow \{0,1\}$ *and computes a signature on the image message* $m^* = f_0(m_0) = f_1(m_1)$ *using secret key* $\mathsf{sk}_{f_b}$*, and gives the resulting signature* $\sigma \leftarrow \mathsf{FS.Sign}(\mathsf{sk}_{f_b}, m_b)$ *to* $\mathcal{A}$*.*
- *The adversary outputs a bit* $b'$*, and the experiment outputs 1 iff* $b' = b$*.*

**Definition 12 (Succinct Functional Signatures).** *A functional signature scheme is called* succinct *if there exists a polynomial* $s(\cdot)$ *such that, for every security parameter* $\lambda \in \mathbb{N}$*,* $f \in \mathcal{F}$*,* $m \in \mathcal{D}_f$*, it holds with probability 1 over* $(\mathsf{mvk}, \mathsf{msk}) \leftarrow \mathsf{FS.Setup}(1^\lambda)$*,* $\mathsf{sk}_f \leftarrow \mathsf{FS.KeyGen}(\mathsf{msk}, f)$*,* $(f(m), \sigma) \leftarrow \mathsf{FS.Sign}(\mathsf{sk}_f, m)$ *that* $|\sigma| \leq s(\lambda, |f(m)|)$*. In particular, the size of* $\sigma$ *is independent of the function's size,* $|f|$*, and the function's input size,* $|m|$*.*

**Succinct Functional Signatures from O-SNARKs.**

In the following we show a construction for message space $\mathcal{M}$ and family of functions $\mathcal{F} = \{f : \mathcal{D}_f \to \mathcal{M}\}$ whose running time is bounded by some fixed polynomial $t_{\mathcal{F}}(|m|)$. To build the scheme, we use two $\mathsf{UF\text{-}CMA}$-secure signature schemes, $\Sigma_0 = (\mathsf{kg}_0, \mathsf{sign}_0, \mathsf{vfy}_0)$ for message space $\mathcal{M}_0$ and $\Sigma' = (\mathsf{kg}', \mathsf{sign}', \mathsf{vfy}')$ for message space $\mathcal{D}$, together with a fully succinct zero-knowledge O-SNARK $\Pi = (\mathsf{Gen}, \mathsf{Prove}, \mathsf{Ver})$ for the $\mathsf{NP}$ language $L$ defined below. While in [BGI14] a single signature scheme is used, we prefer to use two different ones as this allows for a more precise statement since we will need to apply different restrictions to $\mathcal{M}_0$ and $\mathcal{D}$ to obtain a precise proof.

Defining the relation $\mathcal{R}_L$. Let $M_L$ be a random-access machine as defined below, and $t_L(k) = k^{e_L}$ be a polynomial. $\mathcal{R}_L$ is the binary relation consisting of all pairs $(y, w)$ such that, parsing $y = (M_L, x, t)$, $M_L(x, w)$ accepts in at most $t$ steps and $t \leq t_L(|x|)$. The values $x$ are of the form $x = (m^*, \mathsf{mvk}_0)$ where $\mathsf{mvk}_0$ is a public key of the scheme $\Sigma_0$, and $m^* \in \mathcal{M}$ is a message. The values $w$ are instead tuples $w = (m, f, \mathsf{vk}', \sigma_{\mathsf{vk}'}, \sigma_m)$ such that $m \in \mathcal{D}_f$ with $\mathcal{D}_f \subset \mathcal{D}$, and $\sigma_{\mathsf{vk}'}$, $\sigma_m$ are signatures for the schemes $\Sigma_0$ and $\Sigma'$ respectively. On input such a pair $(x, w)$, $M_L(x, w)$ is the random-access machine that accepts iff the following conditions (1)&(2)&(3) hold:

$$\begin{aligned}
&(1)\ \ m^* = f(m)\\
&(2)\ \ \mathsf{vfy}'(\mathsf{vk}', m, \sigma_m) = 1\\
&(3)\ \ \mathsf{vfy}_0(\mathsf{mvk}_0, f|\mathsf{vk}', \sigma_{\mathsf{vk}'}) = 1
\end{aligned}$$

Given polynomial bounds on the running times of verification algorithms $\mathsf{vfy}'$ and $\mathsf{vfy}_0$, and a (fixed) bound $t_{\mathcal{F}}(\cdot)$ on the size and running time of every $f \in \mathcal{F}$, one can deduce a polynomial time bound $t_L(|x|) = |x|^{e_L}$ for the machine $M_L$.

The construction. Using the signature schemes $\Sigma_0, \Sigma'$ and a fully-succinct zero-knowledge O-SNARK $\Pi$ for $\mathsf{NP}$, we construct the functional signature scheme $\mathsf{FS}[\Sigma_0, \Sigma', \Pi] = (\mathsf{FS.Setup}, \mathsf{FS.KeyGen}, \mathsf{FS.Sign}, \mathsf{FS.Ver})$ as follows:

$\mathsf{FS.Setup}(1^\lambda)$ :
Generate a pair of keys for $\Sigma_0$: $(\mathsf{msk}_0, \mathsf{mvk}_0) \leftarrow \mathsf{kg}_0(1^\lambda)$.
Generate a $\mathsf{crs}$ for $\Pi$: $\mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda)$.
Set the master secret key $\mathsf{msk} = \mathsf{msk}_0$, and the master verification key $\mathsf{mvk} = (\mathsf{mvk}_0, \mathsf{crs})$.

$\mathsf{FS.KeyGen}(\mathsf{msk}, f)$ :

Generate a new key pair $(\mathsf{sk}', \mathsf{vk}') \leftarrow \mathsf{kg}'(1^\lambda)$ for the scheme $\Sigma'$.
Compute $\sigma_{\mathsf{vk}'} \leftarrow \mathsf{sign}_0(\mathsf{msk}_0, f|\mathsf{vk}')$, and let the certificate $c$ be $c = (f, \mathsf{vk}', \sigma_{\mathsf{vk}'})$.
Output $\mathsf{sk}_f = (\mathsf{sk}', c)$.

$\mathsf{FS.Sign}(\mathsf{mvk}, f, \mathsf{sk}_f, m)$ :

Parse $\mathsf{sk}_f$ as $(\mathsf{sk}', c = (f, \mathsf{vk}', \sigma_{\mathsf{vk}'}))$.
Sign $m$ using $\mathsf{sk}'$ in $\Sigma'$: $\sigma_m \leftarrow \mathsf{sign}'(\mathsf{sk}', m)$.
Set $y = (M_L, x, t)$ with $x = (\mathsf{mvk}_0, f(m))$, $t = |x|^{e_L}$), and $w = (m, f, \mathsf{vk}', \sigma_{\mathsf{vk}'}, \sigma_m)$.
Run $\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, y, w)$ and output $(m^* = f(m), \pi)$.

$\mathsf{FS.Ver}(\mathsf{mvk}, m^*, \pi)$ :

Parse $\mathsf{mvk} = (\mathsf{mvk}_0, \mathsf{crs})$ and set $y = (M_L, x, t)$ where $x = (\mathsf{mvk}_0, m^*)$ and $t = |x|^{e_L}$.
Output the same bit returned by $\mathsf{Ver}_{e_L}(\mathsf{crs}, y, \pi)$.

**Correctness.** It is not hard to see that as long as $\Sigma_0, \Sigma'$ and $\Pi$ are correct, then $\mathsf{FS}$ is also correct.

**Succinctness.** This property immediately follows from the succinctness of $\Pi$.

**Unforgeability.** We prove the security of $\mathsf{FS}$ under the unforgeability of schemes $\Sigma_0$ and $\Sigma'$ and using the notion of O-SNARKs for a specific family of oracles $\mathbb{O}_{\mathsf{m}\Sigma,Q}$ that we define below.

$\mathbb{O}_{\mathsf{m}\Sigma,Q}$ is parametrized by the algorithms of the signature schemes $\Sigma_0$, $\Sigma'$ and by a polynomial $Q = Q(\lambda)$. Every member $\mathcal{O}$ of $\mathbb{O}_{\mathsf{m}\Sigma,Q}$ is described by a set of secret keys $\mathsf{msk}_0, \mathsf{sk}'_1, \ldots, \mathsf{sk}'_Q$ (i.e., the process of sampling $\mathcal{O} \leftarrow \mathbb{O}$ consists of running $(\mathsf{mvk}_0, \mathsf{msk}_0) \xleftarrow{\$} \mathsf{kg}_0(1^\lambda)$ and $(\mathsf{vk}'_i, \mathsf{sk}'_i) \xleftarrow{\$} \mathsf{kg}'_1(1^\lambda), \forall i \in [Q]$). The oracle $\mathcal{O}$ works as follows:

$$\mathcal{O}(i, `vk') = \begin{cases} \mathsf{mvk}_0 & \text{If } i = 0, \\ \mathsf{vk}'_i & \text{otherwise.} \end{cases} \qquad \mathcal{O}(i, `sk') = \begin{cases} \perp & \text{If } i = 0, \\ \mathsf{sk}'_i & \text{otherwise.} \end{cases}$$

$$\mathcal{O}(i, m) = \begin{cases} (\mathsf{Cnt}, \mathsf{sign}_0(\mathsf{msk}_0, m|\mathsf{vk}'_{\mathsf{Cnt}})), \mathsf{Cnt} \leftarrow \mathsf{Cnt} + 1 & \text{If } i = 0 \text{ and } \mathsf{Cnt} \leq Q, \\ \perp & \text{If } i = 0 \text{ and } \mathsf{Cnt} > Q, \\ \mathsf{sign}'(\mathsf{sk}'_i, m) & \text{otherwise.} \end{cases}$$

For the sake of simplicity we compactly denote $\mathcal{O}_0(\cdot) = \mathcal{O}(0, \cdot)$ and $\mathcal{O}'_i(\cdot) = \mathcal{O}(i, \cdot)$ for all $i > 0$. From the above description, note that oracle $\mathcal{O}_0$ is stateful and we assume it starts with $\mathsf{Cnt} = 1$.

Finally, we point out that for some technical reasons that we mention in Remark 7 at the end of this section, it is not possible to use the notion of O-SNARK for a *single* signing oracle to prove the security of the functional signature scheme. This is the reason why we explicitly considered O-SNARKs for this more complex family of multiple signing oracles.

**Theorem 13.** *If $\Pi$ is an O-SNARK for $\mathbb{O}_{\mathsf{m}\Sigma,Q}$ for every $Q = \mathsf{poly}(\lambda)$, and $\Sigma_0, \Sigma'$ are* UF-CMA*-secure, then $\mathsf{FS}[\Sigma_0, \Sigma', \Pi]$ is an unforgeable functional signature.*

*Proof.* Our proof consists of the following steps:

1. We show that for every successful $\mathcal{A}_{\mathsf{FS}}$ against the unforgeability of $\mathsf{FS}$ there exists an O-SNARK adversary $\tilde{\mathcal{A}}$ for an oracle from $\mathbb{O}_{\mathsf{m}\Sigma,Q}$ such that $\tilde{\mathcal{A}}$ outputs a valid proof with the same (non-negligible) probability of success of $\mathcal{A}_{\mathsf{FS}}$. By the adaptive proof of knowledge for $\mathbb{O}_{\mathsf{m}\Sigma,Q}$ we then

obtain that for such $\tilde{\mathcal{A}}$ there exists a suitable extractor $\mathcal{E}_{\tilde{\mathcal{A}}}$ that outputs a valid witness with all but negligible probability.

2. From the previous point, considering adversary $\tilde{\mathcal{A}}$ and the corresponding extractor, we can partition adversary-extractor pairs in two types: (1) those that yield a witness $w$ containing a pair $(f, \mathsf{vk}')$ that was never signed before, and (2) those that yield $w$ containing $(f, \mathsf{vk}')$ that was signed before. We show that adversaries of type (1) can be used to break the security of the signature scheme $\Sigma_0$, whereas adversaries of type (2) can be used to break the security of $\Sigma'$.

Existence of an extractor for $\mathcal{A}_{\mathsf{FS}}$. Consider any adversary $\mathcal{A}_{\mathsf{FS}}$ that while running in $\mathbf{Exp}^{\mathsf{FS\text{-}UF}}_{\mathcal{A}_{\mathsf{FS}},\mathsf{FS}}$ it outputs $(m^*, \pi^*)$ and makes the experiment generate $Q$ secret keys of the scheme $\Sigma'$. For every such $\mathcal{A}_{\mathsf{FS}}$ we show there exists another adversary $\tilde{\mathcal{A}}^{\mathcal{O}}$ that, on input $\mathsf{crs}$, and given oracle $\mathcal{O} \leftarrow \mathbb{O}_{\mathsf{m}\Sigma,Q}$, outputs a pair $(y, \pi^*)$. We describe $\tilde{\mathcal{A}}^{\mathcal{O}}$ below. During its execution it maintains a dictionary $D$ similar to the one in the definition of $\mathbf{Exp}^{\mathsf{FS\text{-}UF}}_{\mathcal{A}_{\mathsf{FS}},\mathsf{FS}}$, except that instead of storing mappings like $(f, i) \to \mathsf{sk}^i_f$, it maps a pair $(f, i)$ to a triple $(j, \mathsf{sk}', c)$ where $j \in [Q]$. Intuitively, this means that a queried pair $(f, i)$ is associated to oracle $\mathcal{O}'_j$.

$\tilde{\mathcal{A}}^{\mathcal{O}}(\mathsf{crs})$ :

Query $\mathsf{mvk}_0 \leftarrow \mathcal{O}_0(\text{`}vk')$ and run $\mathcal{A}^{\mathcal{O}_{\mathsf{key}},\mathcal{O}_{\mathsf{sign}}}_{\mathsf{FS}}(\mathsf{mvk} = (\mathsf{crs}, \mathsf{mvk}_0))$
Simulate queries $(f, i)$ to $\mathcal{O}_{\mathsf{key}}$ as follows:
  if $[(f, i) \to (j, \mathsf{sk}', c)] \in D$: output $\mathsf{sk}^i_f = (\mathsf{sk}', c)$
  else if $[(f, i) \to (j, \cdot, c)] \in D$:
    ask $\mathsf{sk}'_j \leftarrow \mathcal{O}'_j(\text{`}sk')$ and output $\mathsf{sk}^i_f = (\mathsf{sk}'_j, c)$
  else:
    ask $(j, \sigma_{\mathsf{vk}'_j}) \leftarrow \mathcal{O}_0(f)$, $\mathsf{vk}'_j \leftarrow \mathcal{O}'_j(\text{`}vk')$, $\mathsf{sk}'_j \leftarrow \mathcal{O}'_j(\text{`}sk')$
    add $(f, i) \to (j, \mathsf{sk}'_j, c)$ to $D$ with $c = (f, \mathsf{vk}'_j, \sigma_{\mathsf{vk}'_j})$
    output $\mathsf{sk}^i_f = (\mathsf{sk}'_j, c)$
Simulate queries $(f, i, m)$ to $\mathcal{O}_{\mathsf{sign}}$ as follows:
  if $(f, i)$ not assigned in $D$:
    ask $(j, \sigma_{\mathsf{vk}'_j}) \leftarrow \mathcal{O}_0(f)$, $\mathsf{vk}'_j \leftarrow \mathcal{O}'_j(\text{`}vk')$
    add $(f, i) \to (j, \cdot, c)$ to $D$ with $c = (f, \mathsf{vk}'_j, \sigma_{\mathsf{vk}'_j})$
    ask $\sigma_m \leftarrow \mathcal{O}'_j(m)$
    set $x = (\mathsf{mvk}_0, f(m))$, $t = |x|^{e_L}$, $w = (m, f, \mathsf{vk}'_j, \sigma_{\mathsf{vk}'_j}, \sigma_m)$
    run $\pi \leftarrow \mathsf{Prove}(\mathsf{prs}, (M_L, x, t), w)$
    output $(f(m), \pi)$
  if $[(f, i) \to (j, \cdot, c)] \in D$: parse $c = (f, \mathsf{vk}'_j, \sigma_{\mathsf{vk}'_j})$
    ask $\sigma_m \leftarrow \mathcal{O}'_j(m)$
    set $x = (\mathsf{mvk}_0, f(m))$, $t = |x|^{e_L}$, $w = (m, f, \mathsf{vk}'_j, \sigma_{\mathsf{vk}'_j}, \sigma_m)$
    run $\pi \leftarrow \mathsf{Prove}(\mathsf{prs}, (M_L, x, t), w)$
    output $(f(m), \pi)$
When $\mathcal{A}_{\mathsf{FS}}$ outputs $(m^*, \pi^*)$
  set $y = (M_L, x = (\mathsf{mvk}_0, m^*), t = |x|^{e_L})$
  output $(y, \pi^*)$

As one can see, given the definition of oracles from $\mathbb{O}_{\mathsf{m}\Sigma,Q}$, the simulation provided by $\tilde{\mathcal{A}}$ to $\mathcal{A}_{\mathsf{FS}}$ is perfect. So, whenever $\mathcal{A}_{\mathsf{FS}}$ outputs a valid forgery $\tilde{\mathcal{A}}$ outputs a pair $(y, \pi^*)$ that verifies correctly. Moreover, defined in this way, the adversary $\tilde{\mathcal{A}}^{\mathcal{O}}$ fits the definition of adaptive proof of knowledge for $\mathbb{O}_{\mathsf{m}\Sigma,Q}$ by which we know that there exists an extractor $\mathcal{E}_{\tilde{\mathcal{A}}}$ that, given the same input of $\tilde{\mathcal{A}}^{\mathcal{O}}$ and the transcript of oracle queries/answers made and received by $\tilde{\mathcal{A}}^{\mathcal{O}}$, outputs a witness $w$ such that the probability that $(y, \pi^*)$ verifies and $(y, w) \notin \mathcal{R}_L$ is negligible.

We define the following hybrid games that involve running $\tilde{\mathcal{A}}, \mathcal{E}_{\tilde{\mathcal{A}}}$:

$\mathsf{G}_1$ is the same experiment as $\mathsf{O\text{-}AdPoK}(\lambda, \tilde{\mathcal{A}}, \mathcal{E}_{\tilde{\mathcal{A}}}, \mathbb{O}_{\mathsf{m}\Sigma,Q})$ except that its outcome is defined differently. $\mathsf{G}_1$ outputs 1 iff $\mathsf{Ver}(\mathsf{crs}, y, \pi) = 1$ and the value $m^*$ inside $y$ constitutes a forgery according to the oracle queries made by $\tilde{\mathcal{A}}$ during the game.
By construction of $\tilde{\mathcal{A}}$ from $\mathcal{A}_{\mathsf{FS}}$ it holds

$$\Pr[\mathbf{Exp}^{\mathsf{FS\text{-}UF}}_{\mathcal{A}_{\mathsf{FS}},\mathsf{FS}}(\lambda) = 1] = \Pr[\mathsf{G}_1 \Rightarrow 1] \tag{1}$$

$\mathsf{G}_2$ is the same as $\mathsf{G}_1$ except that in order to output 1 it additionally checks that $(y, w) \in \mathcal{R}_L$.
Essentially, the outcome of $\mathsf{G}_2$ differs from that of $\mathsf{G}_1$ only if in $\mathsf{G}_2$ $(y, w) \notin \mathcal{R}_L$. Hence,

$$\Pr[\mathsf{G}_1 \Rightarrow 1] - \Pr[\mathsf{G}_2 \Rightarrow 1] \leq \Pr[\mathsf{O\text{-}AdPoK}(\lambda, \tilde{\mathcal{A}}, \mathcal{E}_{\tilde{\mathcal{A}}}, \mathbb{O}_{\mathsf{m}\Sigma,Q}) \Rightarrow 1] \tag{2}$$

Moreover, let us define the following two events in game $\mathsf{G}_2$.
Let $w = (m, f, \mathsf{vk}', \sigma_{\mathsf{vk}'}, \sigma_m)$ be the witness returned by $\mathcal{E}_{\tilde{\mathcal{A}}}$:

$\mathsf{Ev}_1$ occurs if $\forall j \in [Q] : \mathsf{vk}' \neq \mathsf{vk}'_j$, or $\exists j \in [Q] : \mathsf{vk}' = \mathsf{vk}'_j$ but $\tilde{\mathcal{A}}$ *never* made a query $\mathcal{O}_0(f)$ that returned $(j, \cdot)$;
$\mathsf{Ev}_2$ occurs if $\mathsf{vk}' = \mathsf{vk}'_j$ for some $j \in [Q]$ and $\tilde{\mathcal{A}}$ did make a query $(j, \sigma) \leftarrow \mathcal{O}_0(f)$.

Clearly it holds

$$\Pr[\mathsf{G}_2 \Rightarrow 1] = \Pr[\mathsf{G}_2 \Rightarrow 1 \wedge \mathsf{Ev}_1] + \Pr[\mathsf{G}_2 \Rightarrow 1 \wedge \mathsf{Ev}_2] \tag{3}$$

In the remaining part of the proof we show that both $\Pr[\mathsf{G}_2 \Rightarrow 1 \wedge \mathsf{Ev}_1]$ and $\Pr[\mathsf{G}_2 \Rightarrow 1 \wedge \mathsf{Ev}_2]$ are negligible under the assumption that, respectively, $\Sigma_0$ and $\Sigma'$ are unforgeable.

**Claim 1** *For every efficient adversary $\mathcal{A}_{\mathsf{FS}}$ there is an efficient forger $\mathcal{F}_0$ such that* $\Pr[\mathsf{G}_2 \Rightarrow 1 \wedge \mathsf{Ev}_1] = \mathbf{Adv}^{\mathsf{UF\text{-}CMA}}_{\mathcal{F}_0,\Sigma_0}(\lambda)$.

*Proof.* Let $\mathcal{A}_{\mathsf{FS}}$ be an adversary that runs in $\mathbf{Exp}^{\mathsf{FS\text{-}UF}}_{\mathcal{A}_{\mathsf{FS}},\mathsf{FS}}(\lambda)$, and let $\tilde{\mathcal{A}}, \mathcal{E}_{\tilde{\mathcal{A}}}$ be the pair of algorithms built out of $\mathcal{A}_{\mathsf{FS}}$ as defined before. Below we show how to build an efficient forger $\mathcal{F}_0$ out of $\tilde{\mathcal{A}}, \mathcal{E}_{\tilde{\mathcal{A}}}$ so that its probability of forging against $\Sigma_0$ is at least $\Pr[\mathsf{G}_2 \Rightarrow 1 \wedge \mathcal{E}_1]$. $\mathcal{F}_0$ gets the public key $\mathsf{mvk}_0$ and has access to oracle $\mathcal{O}_{\Sigma_0} = \mathsf{sign}_0(\mathsf{msk}_0, \cdot)$.

$\mathcal{F}_0^{\mathcal{O}_{\Sigma_0}}(\mathsf{mvk}_0)$ :
Initialize $\mathsf{qt} \leftarrow \emptyset$, $\mathcal{T} \leftarrow \emptyset$, $\mathsf{Cnt} \leftarrow 1$
Generate $(\mathsf{sk}'_i, \mathsf{vk}'_i) \xleftarrow{\$} \mathsf{kg}'(1^\lambda)$ $\forall i \in [Q]$
Generate $\mathsf{crs} = (\mathsf{prs}, \mathsf{vst}) \leftarrow \mathsf{Gen}(1^\lambda)$ and run $\tilde{\mathcal{A}}^{\mathcal{O}}(\mathsf{crs})$
Simulate all queries to $\mathcal{O}'_i$ using $\mathsf{sk}'_i, \mathsf{vk}'_i$ and add all queries-answers to $\mathsf{qt}$
Simulate queries $\mathcal{O}_0(m)$ as follows:
if $m =$ '$vk'$: output $\mathsf{mvk}_0$

else:
ask $\sigma \leftarrow \mathcal{O}_{\Sigma_0}(m|\mathsf{vk}'_{\mathsf{Cnt}})$
add $(m|\mathsf{vk}'_j, \sigma)$ to $\mathsf{qt}$, and add $m|\mathsf{vk}'_j$ to $\mathcal{T}$
increment $\mathsf{Cnt} \leftarrow \mathsf{Cnt} + 1$
output $\sigma$

Let $(y, \pi^*)$ be $\tilde{\mathcal{A}}$'s output
Run $w \leftarrow \mathcal{E}_{\tilde{\mathcal{A}}}(\mathsf{crs}, \mathsf{qt})$
Check that $(y, w) \in \mathcal{R}_L$:
[ Fail ] Abort if this does not hold.
Else parse $w = (m, f, \mathsf{vk}', \sigma_{\mathsf{vk}'}, \sigma_m)$ and proceed:
[A] If $(f|\mathsf{vk}') \notin \mathcal{T}$ return $(f|\mathsf{vk}', \sigma_{\mathsf{vk}'})$.
[B] If $(f|\mathsf{vk}') \in \mathcal{T}$ abort.

Algorithm $\mathcal{F}_0$ can perfectly simulate $\mathsf{G}_2$ to $\tilde{\mathcal{A}}$ and $\mathcal{E}_{\tilde{\mathcal{A}}}$. Furthermore, it is easy to see that if $\mathsf{G}_2$ outputs 1 and $\mathsf{Ev}_1$ occurs, then $\mathcal{F}_0$'s simulation ends up exactly in case $(A)$, that is $\mathcal{F}_0$ returns a signature $\sigma_{\mathsf{vk}'}$ on a new message $f|\mathsf{vk}'$. Since $(y, w) \in \mathcal{R}_L$ one has that $\sigma_{\mathsf{vk}'}$ is valid, and thus is a forgery.

Finally, it is worth noting that for this simulation the adversary $\tilde{\mathcal{A}}$ can even ask $\mathcal{O}'_j(\text{`}sk')$ for *all* $j$ oracles without affecting our reduction. □

**Claim 2** *For every efficient adversary $\mathcal{A}_{\mathsf{FS}}$ there is an efficient forger $\mathcal{F}'$ such that* $\Pr[\mathsf{G}_2 \Rightarrow 1 \wedge \mathsf{Ev}_2] \leq Q \cdot \mathbf{Adv}^{\mathsf{UF\text{-}CMA}}_{\mathcal{F}',\Sigma'}(\lambda)$.

*Proof.* Let $\mathcal{A}_{\mathsf{FS}}$ be an adversary that runs in $\mathbf{Exp}^{\mathsf{FS\text{-}UF}}_{\mathcal{A}_{\mathsf{FS}},\mathsf{FS}}(\lambda)$, and let $\tilde{\mathcal{A}}, \mathcal{E}_{\tilde{\mathcal{A}}}$ be the pair of algorithms built out of $\mathcal{A}_{\mathsf{FS}}$ as defined before. Below we show how to build an efficient forger $\mathcal{F}'$ out of $\tilde{\mathcal{A}}, \mathcal{E}_{\tilde{\mathcal{A}}}$ so that its probability of forging against $\Sigma'$ is at least $\Pr[\mathsf{G}_2 \Rightarrow 1 \wedge \mathsf{Ev}_2]/Q$. $\mathcal{F}'$ gets a public key $\mathsf{vk}'$ and has access to oracle $\mathcal{O}_{\Sigma'} = \mathsf{sign}'(\mathsf{sk}', \cdot)$.

$\mathcal{F}'^{\mathcal{O}'}(\mathsf{vk}')$ :
Initialize $\mathsf{qt} \leftarrow \emptyset$, $\mathcal{T} \leftarrow \emptyset$, $\mathsf{Cnt} \leftarrow 0$
Generate $\mathsf{crs} = (\mathsf{prs}, \mathsf{vst}) \leftarrow \mathsf{Gen}(1^\lambda)$
Generate a pair $(\mathsf{msk}_0, \mathsf{mvk}_0) \leftarrow \mathsf{kg}_0(1^\lambda)$
Choose a random $q \xleftarrow{\$} \{1, \dots Q\}$
Generate $(\mathsf{sk}'_i, \mathsf{vk}'_i) \xleftarrow{\$} \mathsf{kg}'(1^\lambda)\ \forall i \in [Q] \setminus \{q\}$
Run $\tilde{\mathcal{A}}^{\mathcal{O}}(\mathsf{crs})$
Simulate all queries to $\mathcal{O}_0$ using $\mathsf{mvk}_0, \mathsf{msk}_0$:
add all queries-answers to $\mathsf{qt}$ and all signed messages $m|\mathsf{vk}'$ to $\mathcal{T}$
Simulate all queries to $\mathcal{O}'_i$ using $\mathsf{vk}'_i, \mathsf{sk}'_i$ for all $i \in [Q] \setminus \{q\}$
add all queries-answers to $\mathsf{qt}$
Simulate queries $\mathcal{O}'_q(m)$ as follows:
if $m = \text{`}vk'$: output $\mathsf{vk}'$ and add $(\text{`}vk', \mathsf{vk}')$ to $\mathsf{qt}$
else if $m = \text{`}sk'$: Abort
else: ask $\sigma \leftarrow \mathcal{O}_{\Sigma'}(m)$ and add $(m, \sigma)$ to $\mathsf{qt}$
output $\sigma$
Let $(y, \pi^*)$ be $\tilde{\mathcal{A}}$'s output
Run $w \leftarrow \mathcal{E}_{\tilde{\mathcal{A}}}(\mathsf{crs}, \mathsf{qt})$

Check that $(y, w) \in \mathcal{R}_L$:
[ Fail ] Abort if this does not hold.
Else parse $w = (m, f, \mathsf{vk}^*, \sigma_{\mathsf{vk}^*}, \sigma_m)$ and proceed:
[A] If $(f|\mathsf{vk}^*) \notin \mathcal{T}$ Abort.
[B] If $(f|\mathsf{vk}^*) \in \mathcal{T}$ and $\mathsf{vk}^* \neq \mathsf{vk}'$ Abort.
[C] If $(f|\mathsf{vk}^*) \in \mathcal{T}$ and $\mathsf{vk}^* = \mathsf{vk}'$ return $(m, \sigma_m)$.

As one can see, unless it aborts, algorithm $\mathcal{F}'$ can perfectly simulate $\mathsf{G}_2$ to $\tilde{\mathcal{A}}$ and $\mathcal{E}_{\tilde{\mathcal{A}}}$. Furthermore, it is easy to see that if $\mathsf{G}_2$ outputs 1, $\mathsf{Ev}_2$ occurs, and there is no abort while answering queries, then the simulation of $\mathcal{F}'$ ends up in cases $(B)$ or $(C)$. However since $(f|\mathsf{vk}^*) \in \mathcal{T}$, we have that $\mathsf{vk}^* = \mathsf{vk}'_j$ for some $j \in [Q]$ (where we let $\mathsf{vk}'_q = \mathsf{vk}'$). So, if there is no abort at all, we have that $\mathsf{vk}^* = \mathsf{vk}'$ and thus $\mathcal{F}'$ returns a valid signature $\sigma$ on a message $m$ (recall that validity follows from $(y, w) \in \mathcal{R}_L$). By definition of $\mathsf{G}_2$ we also have that if it outputs 1, then the message $m^*$ in $y$ constitutes a forgery according to the definition of $\mathbf{Exp}^{\mathsf{FS\text{-}UF}}$. In particular, it holds that for the given $f|\mathsf{vk}'$, there was no signing query $\mathcal{O}'_q(m)$ such that $m^* = f(m)$. Therefore, if $m$ is such that $m^* = f(m)$ (again this follows from $(y, w) \in \mathcal{R}_L$), then $m$ cannot have been queried to $\mathcal{O}'_q$, i.e., $\mathcal{F}'$ never queried $m$ to its signing oracle. From this we have that, as long as $\mathsf{G}_2$ outputs 1, $\mathsf{Ev}_2$ occurs and there is no abort, then $\mathcal{F}'$ outputs a valid forgery. To conclude the proof, we observe that $\mathcal{F}'$ does not abort with probability $1/Q$ which is the probability that the guess of $q$, for which $\mathsf{vk}^* = \mathsf{vk}'$, is correct. Therefore, we have that $\mathbf{Adv}^{\mathsf{UF\text{-}CMA}}_{\mathcal{F}',\Sigma'}(\lambda) = \Pr[\mathsf{G}_2 \Rightarrow 1 \wedge \mathsf{Ev}_2]/Q$.

Finally, we note that the above proof works even if the adversary $\tilde{\mathcal{A}}$ queries $\mathcal{O}'_j(\text{`}sk')$ on all oracles but the $q$-th one. This observation will be useful when we discuss the existence of O-SNARKs for this oracle family. □

Putting together the bounds in equations (1), (2) and (3), with the results of Claims 1 and 2, eventually we obtain:

$$\Pr[\mathbf{Exp}^{\mathsf{FS\text{-}UF}}_{\mathcal{A},\mathsf{FS}}(\lambda) = 1] \leq \Pr[\mathsf{O\text{-}AdPoK}(\lambda, \tilde{\mathcal{A}}, \mathcal{E}_{\tilde{\mathcal{A}}}, \mathbb{O}_{\mathsf{m}\Sigma,Q}) \Rightarrow 1] + \mathbf{Adv}^{\mathsf{UF\text{-}CMA}}_{\mathcal{F}_0,\Sigma_0}(\lambda) + \mathbf{Adv}^{\mathsf{UF\text{-}CMA}}_{\mathcal{F}',\Sigma'}(\lambda)$$

which shows that any efficient adversary has at most negligible probability of breaking the security of scheme $\mathsf{FS}$ under the assumption that $\Pi$ is an O-SNARK for $\mathbb{O}_{\mathsf{m}\Sigma,Q}$ and the schemes $\Sigma_0, \Sigma'$ are unforgeable. □

**Non-adaptive unforgeability.** Similarly to the homomorphic signature case, it is possible to show that the functional signature scheme achieves security against (functional signature) adversaries that make *non-adaptive* signing queries (i.e., all queries are declared at the beginning of the game). This weaker security can be proven assuming that $\Pi$ is a *non-adaptive* O-SNARK (see Definition 8). Combining this change with the result of Theorem 4 we obtain the following:

**Theorem 14.** *If $\Pi$ is a SNARK and $\Sigma_0, \Sigma'$ are* UF-CMA*-secure signature schemes, then* $\mathsf{FS}[\Sigma_0, \Sigma', \Pi]$ *is a functional signature where unforgeability holds against adversaries that make non-adaptive signing queries.*

*Proof.* The proof of the theorem can be obtained via straightforward modifications to the proof of Theorem 13. Having in mind the intuition provided earlier, the main idea is that to work with non-adaptive adversaries, one can define a non-adaptive O-SNARK adversary $\tilde{\mathcal{A}}$ for every non-adaptive functional signature adversary $\mathcal{A}$. In particular, the non-adaptive queries of $\mathcal{A}$ can be used to define the non-adaptive queries of $\tilde{\mathcal{A}}$. The rest of the proof proceeds analogously.

**Function privacy.** We show that the functional signature construction satisfies function privacy provided that the O-SNARK is zero-knowledge.

**Theorem 15.** *If $\Pi$ is a zero-knowledge O-SNARK then* $\mathsf{FS}$ *satisfies function privacy.*

*Proof.* We show that for every adversary $\mathcal{A}_{\mathsf{priv}}$ against the function privacy experiment $\mathbf{Exp}^{\mathsf{FS\text{-}FPri}}_{\mathcal{A}_{\mathsf{priv}},\mathsf{FS}}(\lambda)$, we can construct a distinguisher algorithm $\mathcal{D}$ against the zero knowledge property of $\Pi$.

Consider the following two hybrid experiments:

$\mathsf{G}_0$ is the same as $\mathbf{Exp}^{\mathsf{FS\text{-}FPri}}_{\mathcal{A}_{\mathsf{priv}},\mathsf{FS}}(\lambda)$. In particular, the $\mathsf{crs}$ for $\Pi$ is generated honestly using $\mathsf{Gen}$; and the challenge functional signature $\sigma = (f_b(m_b))$ is generated as $\sigma \leftarrow \mathsf{FS.Sign}(\mathsf{sk}_{f_b}, m_b)$, i.e., by running $\pi \leftarrow \mathsf{Prove}(\mathsf{prs}, (M_L, x, t), w)$ where $x = (\mathsf{mvk}_0, f_b(m_b))$, $t = |x|^{e_L}$, and $w = (m_b, f_b, \mathsf{vk}', \sigma_{\mathsf{vk}'}, \sigma)$.

$\mathsf{G}_1$ is the same as $\mathsf{G}_0$ except that one uses the zero-knowledge simulator algorithm $S$ in order to generate both the $\mathsf{crs}$ and the proof in the challenge. Namely, $(\mathsf{prs}, \mathsf{vst}, \mathsf{tr}) \leftarrow S^{\mathsf{crs}}(1^\lambda)$, and the challenge signature is generated by running $\pi \leftarrow S^{\mathsf{Prove}}(aux, \mathsf{prs}, (M_L, x, t), \mathsf{tr})$ for $x = (\mathsf{mvk}_0, m')$, $t = |x|^{e_L}$, where $m' = f_0(m_0) = f_1(m_1)$.

Denote by $\mathsf{win}_0$ and $\mathsf{win}_1$ the advantage of the adversary $\mathcal{A}_{\mathsf{priv}}$ in guessing the bit $b$ in $\mathsf{G}_0$, and $\mathsf{G}_1$, respectively. Clearly $\mathsf{win}_1 = 1/2$ since the bit $b$ is not used at all, and thus the view of $\mathcal{A}_{\mathsf{priv}}$ is independent of $b$. To complete the proof we show that under the assumption that $\Pi$ is zero-knowledge, the following holds:

**Claim 3** $\mathsf{win}_0 - \mathsf{win}_1 \leq negl(\lambda)$.

To prove this, we show that for any $\mathcal{A}_{\mathsf{priv}}$ such that $\mathsf{win}_0 - \mathsf{win}_1 = \epsilon$ is non-negligible there is a distinguisher $\mathcal{D}$ that succeeds against the zero-knowledge property of $\Pi$ with the same advantage $\epsilon$. $\mathcal{D}$ is defined as follows:

$\mathcal{D}(\mathsf{crs})$ :
Generate a pair $(\mathsf{msk}_0, \mathsf{mvk}_0) \leftarrow \mathsf{kg}_0(1^\lambda)$
Run $\mathcal{A}_{\mathsf{priv}}(\mathsf{mvk} = (\mathsf{crs}, \mathsf{mvk}_0))$
$\mathcal{A}_{\mathsf{priv}}$ adaptively chooses function queries $f_0, f_1$ and message pairs $m_0, m_1$
such that $f_0(m_0) = f_1(m_1)$:
For each $f_b$ asked by $\mathcal{A}_{\mathsf{priv}}$, return the secret key $\mathsf{sk}_{f_b} \leftarrow \mathsf{FS.KeyGen}(\mathsf{msk}, f_b)$
To answer the challenge $\mathcal{D}$ proceeds as follows:
  pick $b \xleftarrow{\$} \{0,1\}$
  set $x = (\mathsf{mvk}_0, f_b(m_b))$, $t = |x|^{e_L}$, and $w = (m_b, f_b, \mathsf{vk}', \sigma_{\mathsf{vk}'}, \sigma)$
  output $(y, w)$ (where $y = (M_L, x, t)$) to its challenger, and get back $\pi$
  return $(f_b(m_b), \pi)$ to $\mathcal{A}_{\mathsf{priv}}$
Let $b'$ be $\mathcal{A}_{\mathsf{priv}}$'s output
If $b' = b$ output 1, else output 0.

Note that when $\mathcal{D}$ receives $\mathsf{crs}$ and $\pi$ that are generated using the real algorithms, then $\mathcal{D}$ is perfectly simulating $\mathsf{G}_0$ to $\mathcal{A}_{\mathsf{priv}}$. Otherwise, if $\mathcal{D}$ receives $\mathsf{crs}$ and $\pi$ that are generated using the simulator, then $\mathcal{D}$ perfectly simulates $\mathsf{G}_1$. Therefore it is easy to see that $\mathcal{D}$'s advantage is $\mathsf{win}_0 - \mathsf{win}_1$.

*Remark 6 (On the applicability of Corollary 1).* For the same reasons discussed in Remark 4, it is not possible to apply the result of Corollary 1 to conclude the that the (adaptive) security of the functional signature scheme holds under classical SNARKs.

*Remark 7.* [On the use of multiple signing oracles] In order to prove the security of the functional signature scheme, one might be tempted to use the notion of O-SNARK with a single signing oracle. Precisely, one might use O-SNARKs for $\mathbb{O}_{\Sigma_0}$ when making a reduction to $\Sigma_0$ and O-SNARKs for $\mathbb{O}_{\Sigma'}$ when making a reduction to $\Sigma'$. Unfortunately, this approach does not work for an intricate technical reason that we explain here. Intuitively, assume that one wants to build an O-SNARK adversary $\tilde{\mathcal{A}}$ that has access to a single signing oracle, say from $\mathbb{O}_{\Sigma_0}$. Then the secret keys needed to simulate all the other oracles have to be given to $\tilde{\mathcal{A}}$ as part of its auxiliary input ($\tilde{\mathcal{A}}$ needs them to simulate $\mathcal{A}_{\mathsf{FS}}$). At this point the issue is that such secret keys in fact give an efficient way to compute a witness for several $y$ in the relation $\mathcal{R}_L$. Therefore, if the extractor gets these secret keys as auxiliary information, we then have no guarantee that, while doing a reduction to the unforgeability of the signature scheme, the extractor will output a witness of the form we expect.

### 5.3 SNARKs on authenticated data

As another application of O-SNARKs we consider the generic construction of SNARKs on authenticated data that is given in [BBFR15]. Since this construction is very similar to the homomorphic signature scheme that we present in Section 5.1, we only provide an informal discussion of this application. In [BBFR15] Backes et al. introduce the notion of SNARKs on authenticated data to capture in an explicit way the possibility of performing (zero-knowledge) proofs about statements that are authenticated by third parties, i.e., to prove that $(x, w) \in \mathcal{R}$ for some $x$ for which there is a valid signature. While the main focus of that work is on a concrete construction based on quadratic arithmetic programs, the authors also show a generic construction based on SNARKs and digital signatures. Roughly speaking, this construction consists in letting the prover use a SNARK to prove a statement of the form "$\exists x, w, \sigma : (x, w) \in \mathcal{R} \wedge \mathsf{vfy}(\mathsf{vk}, \tau | x, \sigma) = 1$", for some public label $\tau$ of the statement. The formalization of their model is rather similar to that of homomorphic signatures in this paper (e.g., they also use labels). Noticeable differences are that their construction uses pre-processing SNARKs for arithmetic circuit satisfiability, and that to handle several functions they use different SNARK instantiations (one per function).

In [BBFR15] the security proof of this generic construction is only sketched, and in particular they use the existence of an extractor for an adversary that interacts with a signing oracle without providing a particular justification on its existence. With a more careful look, it is possible to see that this security proof incurs the same issue of extraction in the presence of oracles. Using the same techniques that we developed in this paper for the homomorphic signature scheme,[14] it is possible to prove the security of that generic construction using O-SNARKs for signing oracles (or non-adaptive security based on classical SNARKs). In conclusion, for this construction one can either conjecture that a specific SNARK scheme (e.g., [PHGR13]) is secure in the presence of oracles, or, more conservatively, argue only the non-adaptive security of the primitive under the existence of classical SNARKs.

[14] The only major difference is that one has to consider a specification of our definitions to the case of pre-processing SNARKs.

### Acknowledgements

We would like to thank Manuel Barbosa for useful suggestions and valuable discussions on this work. We also thank Bogdan Warinschi for helpful discussions in an early stage of this work.

## References

BBFR15. Michael Backes, Manuel Barbosa, Dario Fiore, and Raphael M. Reischuk. ADSNARK: Nearly practical and privacy-preserving proofs on authenticated data. In *2015 IEEE Symposium on Security and Privacy*, pages 271–286. IEEE Computer Society Press, 2015.

BCC88. Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci.*, 37(2):156–189, October 1988.

BCC+14. Nir Bitansky, Ran Canetti, Alessandro Chiesa, Shafi Goldwasser, Huijia Lin, Aviad Rubinstein, and Eran Tromer. The hunting of the SNARK. Cryptology ePrint Archive, Report 2014/580, 2014. `http://eprint.iacr.org/2014/580`.

BCCT12. Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In Shafi Goldwasser, editor, *ITCS 2012*, pages 326–349. ACM, January 2012.

BCCT13. Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for SNARKS and proof-carrying data. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 111–120. ACM Press, June 2013.

BCPR14. Nir Bitansky, Ran Canetti, Omer Paneth, and Alon Rosen. On the existence of extractable one-way functions. In David B. Shmoys, editor, *46th ACM STOC*, pages 505–514. ACM Press, May / June 2014.

BCTV14. Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Scalable zero knowledge via cycles of elliptic curves. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 276–294. Springer, August 2014.

BF11. Dan Boneh and David Mandell Freeman. Homomorphic signatures for polynomial functions. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 149–168. Springer, May 2011.

BG08. Boaz Barak and Oded Goldreich. Universal arguments and their applications. *SIAM Journal on Computing*, 38(5):1661–1694, 2008.

BGI14. Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 501–519. Springer, March 2014.

BHZ87. Ravi B. Boppana, Johan Hastad, and Stathis Zachos. Does co-np have short interactive proofs? *Information Processing Letters*, 25(2):127 – 132, 1987.

BP15. Elette Boyle and Rafael Pass. Limits of extractability assumptions with distributional auxiliary input. ASIACRYPT, 2015.

BSCG+13. Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. SNARKs for C: Verifying program executions succinctly and in zero knowledge. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 90–108. Springer, August 2013.

CF13. Dario Catalano and Dario Fiore. Practical homomorphic MACs for arithmetic circuits. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 336–352. Springer, May 2013.

CFW14. Dario Catalano, Dario Fiore, and Bogdan Warinschi. Homomorphic signatures with efficient verification for polynomial functions. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 371–389. Springer, August 2014.

GGPR13. Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 626–645. Springer, May 2013.

GH98. Oded Goldreich and Johan Håstad. On the complexity of interactive proofs with bounded communication. *Information Processing Letters*, 67(4):205 – 214, 1998.

GMR89. Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

GVW02. Oded Goldreich, Salil Vadhan, and Avi Wigderson. On interactive proofs with a laconic prover. *computational complexity*, 11(1-2):1–53, 2002.

GVW15. Sergey Gorbunov, Vinod Vaikuntanathan, and Daniel Wichs. Leveled fully homomorphic signatures from standard lattices. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 469–477. ACM Press, June 2015.

GW11. Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 99–108. ACM Press, June 2011.

GW13. Rosario Gennaro and Daniel Wichs. Fully homomorphic message authenticators. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 301–320. Springer, December 2013.

HT98. Satoshi Hada and Toshiaki Tanaka. On the existence of 3-round zero-knowledge protocols. In Hugo Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 408–423. Springer, August 1998.

Kil92. Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *24th ACM STOC*, pages 723–732. ACM Press, May 1992.

Mic94. Silvio Micali. CS proofs (extended abstracts). In *35th FOCS*, pages 436–453. IEEE Computer Society Press, November 1994.

Mic00. Silvio Micali. Computationally sound proofs. *SIAM Journal on Computing*, 30(4):1253–1298, 2000.

Nao03. Moni Naor. On cryptographic assumptions and challenges (invited talk). In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 96–109. Springer, August 2003.

PHGR13. Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy*, pages 238–252. IEEE Computer Society Press, May 2013.

Val08. Paul Valiant. Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In Ran Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 1–18. Springer, March 2008.

Wee05. Hoeteck Wee. On round-efficient argument systems. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *ICALP 2005*, volume 3580 of *LNCS*, pages 140–152. Springer, July 2005.

# A Additional Preliminaries

## A.1 Digital Signatures

A digital signature scheme $\Sigma$ consists of a triple of algorithms $\Sigma = (\mathsf{kg}, \mathsf{sign}, \mathsf{vfy})$ working as follows:

$\mathsf{kg}(1^\lambda)$ the key generation takes as input the security parameter $\lambda$ and returns a pair of keys $(\mathsf{sk}, \mathsf{vk})$.

$\mathsf{sign}(\mathsf{sk}, m)$ on input a signing key $\mathsf{sk}$ and a message $m$, the signing algorithm produces a signature $\sigma$.

$\mathsf{vfy}(\mathsf{vk}, m, \sigma)$ given a triple $\mathsf{vk}, m, \sigma$ the verification algorithm tests if $\sigma$ is a valid signature on $m$ with respect to verification key $\mathsf{vk}$.

The standard security notion for digital signatures, unforgeability against chosen-message attacks ($\mathsf{UF\text{-}CMA}$, for short) is defined by the following experiment:

Experiment $\mathbf{Exp}^{\mathsf{UF\text{-}CMA}}_{\mathcal{F},\Sigma}(\lambda)$

$(\mathsf{sk}, \mathsf{vk}) \xleftarrow{\$} \mathsf{kg}(1^\lambda)$

$(m^*, \sigma^*) \xleftarrow{\$} \mathcal{F}^{\mathsf{sign}(\mathsf{sk},\cdot)}(\mathsf{vk})$

If $\mathsf{vfy}(\mathsf{vk}, m^*, \sigma^*) = 1$ and $m^*$ is "new" then output 1

Else output 0

A message ($m^*$ is said "new" if it is different from all the messages $m_i$ that the adversary queried to the signing oracle $\mathsf{sign}(\mathsf{sk}, \cdot)$ during the experiment. The advantage of a forger $\mathcal{F}$ in breaking the unforgeability against chosen-message attacks ($\mathsf{SUF\text{-}CMA}$) of $\Sigma$ is $\mathbf{Adv}^{\mathsf{SUF\text{-}CMA}}_{\mathcal{F},\Sigma}(\lambda) = \Pr[\mathbf{Exp}^{\mathsf{SUF\text{-}CMA}}_{\mathcal{F},\Sigma}(\lambda) = 1]$.

**Definition 13 (SUF-CMA security).** *A digital signature scheme $\Sigma$ is* SUF-CMA*-secure if for any PPT forger $\mathcal{F}$,* $\mathbf{Adv}^{\mathsf{SUF\text{-}CMA}}_{\mathcal{F},\Sigma}(\lambda) \leq \mathsf{negl}(\lambda)$.

A stronger notion of security is *strong unforgeability* against chosen-message attacks (SUF-CMA). This notion is defined by considering a security experiment slightly different than $\mathbf{Exp}^{\mathsf{UF\text{-}CMA}}_{\mathcal{F},\Sigma}(\lambda)$. Instead of checking whether $m^*$ is "new", one checks whether the pair $(m^*, \sigma^*)$ is "new", i.e., if $(m^*, \sigma^*)$ is different from all the pairs $(m_i, \sigma_i)$ obtained by $\mathcal{F}$ from the signing oracle.

## B The SNARK definition used in [BGI14]

Here we recall the definition of SNARK considered in the work of Boyle, Goldwasser and Ivan [BGI14]. This definition is almost the same as our Definition 3 except that adaptive proof of knowledge is formulated as follows:

- **(Strong) Adaptive Proof of Knowledge.** $\Pi$ is a SNARK for a language $L \in \mathsf{NP}$ with witness relation $\mathcal{R}$ if there exists a negligible function $\mu(\cdot)$ such that for all PPT provers $\mathcal{A}$ there exists a PPT algorithm $\mathcal{E}_{\mathcal{A}} = (\mathcal{E}^1_{\mathcal{A}}, \mathcal{E}^2_{\mathcal{A}})$ such that:
  (i) for every $\mathcal{D}$

$$\left|\Pr\left[\mathcal{D}(\mathsf{crs},\mathsf{vst}) = 1 \,\middle|\, (\mathsf{crs},\mathsf{vst}) \leftarrow \mathsf{Gen}(1^\lambda)\right] - \Pr\left[\mathcal{D}(\mathsf{crs},\mathsf{vst}) = 1 \,\middle|\, (\mathsf{crs},\mathsf{vst},\mathsf{td}) \leftarrow \mathcal{E}^1_{\mathcal{A}}(1^\lambda)\right]\right| = \mu(\lambda)$$

  and (ii)

$$\Pr\left[\begin{array}{c|c} \mathsf{Ver}(\mathsf{crs}, y, \pi) = 1 & (\mathsf{crs},\mathsf{td}) \leftarrow \mathcal{E}^1_{\mathcal{A}}(1^\lambda) \\ \wedge & (y,\pi) \leftarrow \mathcal{A}(\mathsf{crs}) \\ (y,w) \notin \mathcal{R} & w \leftarrow \mathcal{E}^2_{\mathcal{A}}(\mathsf{td},\mathsf{crs},y,\pi) \end{array}\right] = \mu(\lambda)$$

  where the probabilities are taken over $(\mathsf{crs},\mathsf{td}) \leftarrow \mathcal{E}^1_{\mathcal{A}}(1^\lambda, T)$ and the random coins of $\mathcal{E}^2_{\mathcal{A}}$.

Besides the fact that the extractor can take as additional input the trapdoor $\mathsf{td}$, a major difference between this definition and the more standard one seems that in the above definition the extractor $\mathcal{E}^2_{\mathcal{A}}$ does not necessarily take the random tape of $\mathcal{A}$ as part of its input. To the best of our knowledge, we are not aware of constructions under this definition, in the standard model.