Nearly Practical and Privacy-Preserving Proofs on Authenticated Data

Michael Backes[1,2], Dario Fiore[3], and Raphael M. Reischuk[2]

[1] Max Planck Institute for Software Systems (MPI-SWS)
Saarbrücken, Germany
backes@mpi-sws.org
[2] Saarland University
Saarbrücken, Germany
reischuk@cs.uni-saarland.de
[3] IMDEA Software Institute
Madrid, Spain
dario.fiore@imdea.org

Abstract. We study the problem of privacy-preserving proofs on authenticated data in which a party receives data from a trusted source and is requested to prove statements over the data to third parties in a correct and private way, i.e., the third party learns no information on the data but is still assured that the claimed proof is valid. Our work particularly focuses on the challenging requirement that the third party should be able to verify the validity with respect to the specific data authenticated by the source — even without having access to that source. This problem is motivated by various scenarios emerging from several application areas such as wearable computing, smart metering, or general business-to-business interactions. Furthermore, these applications also demand any meaningful solution to satisfy additional properties related to usability and scalability. First, third parties should be able to check proofs very efficiently. Second, the trusted source should be independent of the data processor: it simply (and possibly continuously) provides data, e.g., without knowing which statements will be proven.
This paper formalizes the above three-party model, discusses concrete application scenarios, and introduces a new cryptographic primitive for proving NP relations where statements are authenticated by trusted sources. After discussing a generic approach to construct this primitive, we present a more direct and efficient realization that supports general-purpose NP relations. Our realization significantly improves over state-of-the-art solutions for this model, such as those based on Pinocchio (Oakland'13), by at least three orders of magnitude.

Table of Contents

1 Introduction

With the emergence of modern IT services, a growing number of applications relies on confidential data for various purposes such as billing, legal compliance, etc. For instance, in the emerging area of wearable computing [36,3], smart devices collect measurable human conditions, and subsequently aggregate them for doctors or health insurances. Likewise, in the area of smart metering [35], energy companies intend to collect energy consumption measurements in order to compute the user bills. Or, in the realm of B2B, a company would like to perform efficient computations on business-sensitive data. In these scenarios, the result of the computation is typically used further in interaction with other third parties, be it other humans or companies (doctors, health insurances, energy companies, business collaborators).

This consideration of disseminating the results of a computation to third parties imposes security requirements for both the data owner and the data recipient: On the one hand, the computation inputs might contain sensitive data (such as patient data, energy consumptions, business plans) that the data owner would like to keep confidential. On the other hand, the data recipient would like to be able to verify the correctness of the computation results – even though it is not granted access to the computation input!

To illustrate the problem more formally, we consider a scenario in which a prover $\mathcal{P}$ is requested to prove certain statements $R(D)$ about data D to third parties $\mathcal{V}$, which we call the verifiers. Since the two parties $\mathcal{P}$ and $\mathcal{V}$ may not trust each other, we are interested in the simultaneous achievement of two main security properties: (1) *integrity*, in the sense that $\mathcal{V}$ should be convinced about the validity of $R(D)$. In particular, in order to verify that this statement holds for some specific D, the data is assumed to be generated and authenticated by some *trusted source* $\mathcal{S}$; and (2) *privacy*, in the sense that $\mathcal{V}$ should not learn any information about D beyond what is trivially revealed by $R(D)$.

In addition to the security requirements above, any meaningful solution has to meet the following properties that have been identified as key for practical scalability in previous work: (3) *efficiency*, meaning that $\mathcal{V}$'s verification cost should be much less than the cost of computing the proven statement $R(D)$; and (4) *data independence*, in the sense that the data source $\mathcal{S}$ should be independent of $\mathcal{P}$, i.e., it should be able to provide D without knowing in advance what statements will be proven about D (e.g., the billing function may change over time). In particular, also D's size should not be fixed in advance, i.e., $\mathcal{S}$ can continuously provide data to $\mathcal{P}$, even after some proofs have been generated.

The simultaneous achievement of integrity and privacy is a fundamental goal that has a long research history starting with the seminal work on zero-knowledge proofs [23]. The main goal of this work is to study solutions aiming to achieve *all* of the four properties above, with a particular focus on the setting in which the data is *authenticated* by some trusted source. We believe that such a setting is relevant to many practical scenarios (such as the ones sketched above) and observe that no much prior work addressed the problem of proofs on authenticated data in a systematic and general way. Most work focused on specific computations (e.g., credentials or electronic cash [13,16,29,30]), but very little work addressed the case of proving the integrity of *arbitrary* computations involving authenticated data. An exception is the recent work ZQL [17], which provides an expression language for (privacy-preserving) processing of data that can also be originated by trusted data sources. Inspired by the goals of ZQL, our work is rather focused on the study and realization of efficient cryptographic primitives that can yield suitable solutions for this setting.

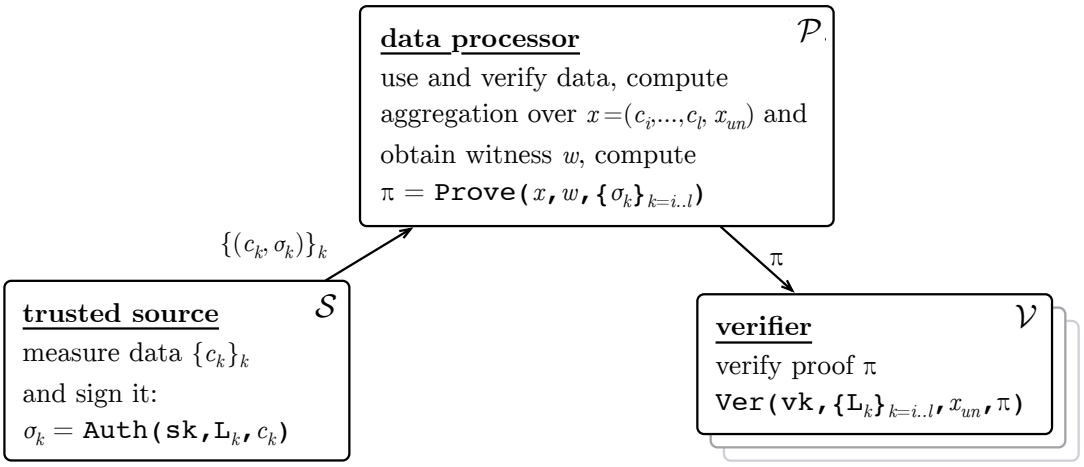

Fig. 1. Scenario of authenticating data D and proving properties R over D.

1.1 Contributions

Our contribution is twofold. First, we fully formalize a model for the above problem by defining a new cryptographic primitive that we call *Succinct Non-Interactive Arguments on Authenticated Data* (or AD-SNARG, for short). SNARGs, first introduced by Micali under the name of "CS proofs" [31], are proof systems that provide *succinct verification*, i.e., the verifier is able to check a long poly-time computation in much less time than that required to run the computation, given the witness. Our new notion of AD-SNARGs extends SNARGs so as to explicitly capture proofs of $\mathcal{NP}$ relations $R(x, w)$ in which the statement (or part of it) is *authenticated*. More precisely, the main difference between SNARGs and AD-SNARGs is that in the former the verifier always knows the statement, whereas in the latter, the authenticated statements are not disclosed to the verifier, yet the verifier can be assured about the existence of w such that $R(x, w)$ holds for the specific x authenticated by the trusted source. Moreover, to model privacy (and looking ahead to our applications) we define the zero-knowledge property so as to hold not only for the witnesses of the relation, but also for the authenticated statements. In particular, our zero-knowledge definition holds also against adversaries who generate the authentication keys.

Turning our attention to concrete realizations, we show that AD-SNARGs can be constructed in a generic fashion by embedding digital signatures into SNARKs (i.e., SNARGs of Knowledge [5]). However, motivated by the fact that this "generic construction" is not very efficient in practice, our second contribution is a *direct and more efficient realization* of AD-SNARGs, that from now on we refer to as the "direct construction". Interestingly, compared to instantiating the generic construction with state-of-the-art SNARKs schemes, our direct construction performs roughly three orders of magnitude better on the prover side. In what follows we give more details on this efficiency aspect: We first discuss the efficiency of instantiating the generic construction, and then we describe our solution.

On the (in)efficiency of the generic construction. The idea of the generic (not very practical) construction of AD-SNARGs for an $\mathcal{NP}$ relation $R(x, w)$ is to let the prover $\mathcal{P}$ prove an extended $\mathcal{NP}$ relation R' which contains the set of tuples (x', w') with $x' = (|x|, \mathsf{pk})$, $w' = (w, x, \sigma)$,

and $\sigma = (\sigma_1, \ldots, \sigma_{|x|})$, such that there is a valid signature σ_i for every statement value x_i at position i under public key pk. The problem with this generic construction is that, in practice, a proof for such extended relation R' is much more expensive than a proof for R. The issue is that R' needs to "embed" the verification algorithm of a signature scheme. If we consider very efficient SNARKs, such as the recent Pinocchio system [33], embedding the verification algorithm means encoding the verification algorithm of the signature with an arithmetic circuit over a specific finite field $\mathbb{F}_p$ (where p is a large prime, the order of some bilinear groups), and then creating a Quadratic Arithmetic Program [19], a QAP for short, out of this circuit. Without going into the details of QAPs (we will review them later in Section 2), we note that the efficiency of the prover in Pinocchio depends on the size of the QAP, which in turn depends on the number of multiplication gates in the relation satisfiability circuit.

Our main observation is that the circuit resulting from expressing the verification algorithm of a digital signature scheme is very likely to be quite inefficient (from a QAP perspective), especially for the prover. Such inefficiency stems from the fact that the circuit would contain a huge number of multiplication gates. In what follows, we discuss why this is the case for various examples of signatures in both the random oracle and the standard model, and based on different algebraic problems. If one considers signature schemes in the random oracle model (which include virtually all the schemes used in practice), any such scheme uses a collision-resistant hash function (e.g., SHA-1) which is thus part of the verification algorithm computation. Unfortunately, as shown also in [33], a QAP (just) for a SHA-1 computation is terribly inefficient due to the high number of multiplication gates (roughly 24 000, for inputs of 416 bits). On the other hand, if we focus on standard model signature schemes, it does not get any better: These schemes involve specific algebraic computations, and encoding these computations into an arithmetic circuit over a field $\mathbb{F}_p$ is costly. For instance, signatures based on pairings [7,37] require pairing computations that amount to, roughly, 10 000 multiplications. RSA-based standard-model signatures (e.g., Cramer-Shoup [15]) require exponentiations over rings of large order (e.g., 3 000 bits), and simulating such computations over $\mathbb{F}_p$ ends up with thousands of multiplication gates as well. Lattice-based signatures (in the standard model), e.g., [10] can be cheaper in terms of the number of multiplications. However, such multiplications typically work over $\mathbb{Z}_q$ for a q much smaller than our p. An option would be to implement mod-q-reductions in $\mathbb{F}_p$ circuits, which is costly. Another option would be to let these schemes work over $\mathbb{Z}_p$, but then one has to work with higher dimensional lattices or (polynomial rings) for security reasons, again incurring a large number of multiplications.

This state of affairs essentially suggests that a QAP encoding a signature verification circuit is likely to have at least one thousand multiplications for *every* signature that must be checked. If, for instance, we consider smart metering, in which the prover wants to certify about 1 000 (signed) meter readings (amounting to approximately 3 weeks of electricity measurements – almost a monthly bill), the costs can become prohibitive!

Our Solution. In contrast, we propose a new, *direct*, AD-SNARG construction that achieves the same efficiency as state-of-the-art SNARGs (e.g., Pinocchio [33]), yet it additionally allows for proofs on authenticated statements. Our scheme builds upon a *corrected* version of Pinocchio[4], and our key technical contribution is a technique (that we illustrate in Section 1.3) for embedding the authentication verification mechanism directly in the proof system, without having to resort to extended relations that would incur the efficiency loss discussed earlier. As a result, the per-

[4] The corrected version of Pinocchio – we emphasize – is available via ePrint and differs from the initially published version [33].

formance of our scheme is almost the same as that of running Pinocchio without any proof about authenticated values.

When comparing our direct construction with the instantiation of the generic scheme in Pinocchio, it is interesting to note that the improvement of our solution lies in the generation of setup keys (for the relation) and proofs, which is currently the main bottleneck of Pinocchio (and other QAP-based schemes [4]). Namely, while these schemes perform excellently in terms of verification time and proof size, the performances get much worse when it comes to generating keys and proofs, especially for relations that have "unfriendly" arithmetic circuit representations, such as signature verification algorithms, as discussed earlier. This is why our technique for avoiding the encoding of signature verification in QAPs allows us to use much smaller QAPs, thus saving at least one thousand multiplication gates per signature involved in the proof.

1.2 Further Related Work

As we mentioned earlier, our work extends the notion of succinct non-interactive arguments (SNARGs) [31,5], which in turn builds on (succinct) interactive proofs [23] and interactive arguments [26,27]. In particular, we focus on the so-called *preprocessing model* where the verifier is required to run an expensive but re-usable key generation phase. In this preprocessing model, several works [24,28,19,6] proposed efficient realizations of SNARGs, and two notable recent works [33,4] have shown efficient, highly-optimized, implementations that support general-purpose computations. These schemes can also support zero-knowledge proofs. It is worth mentioning that all known SNARGs are either in the random oracle model or rely on non-standard non-falsifiable assumptions [32]. Assumptions from this class have been shown [21] likely to be inherent for SNARGs for $\mathcal{NP}$.

The notion of SNARGs is also related to *verifiable computation* [18], in which a (computationally weak) client delegates the computation of a function to a powerful server and wants to verify the result efficiently. As noted in previous work, by using SNARGs for $\mathcal{NP}$, it is possible to construct a verifiable computation scheme, and several works [19,33,4] indeed follow this approach. However, alternative approaches to realizing verifiable computation have been proposed, notably based on fully homomorphic encryption [18,14,1] or attribute-based encryption [34].

Another line of work which is closely related to ours is the one on *homomorphic authentication* (comprising both homomorphic signatures [25,9] and MACs [20,12,2]). The main idea of this primitive is that, given a set of messages $(\sigma_1, \ldots, \sigma_n)$ authenticated using a secret key sk, anyone can evaluate a program P on such authenticated messages in a way that the result $\sigma \leftarrow P(\{\sigma_i\})$ is again authenticated with respect to the same key sk (or some public key vk in the case of signatures). Compared to AD-SNARGs, homomorphic MACs/signatures satisfy a similar notion of soundness, and they have an additional nice property of composability, i.e., one can run a program on results authenticated by other programs. On the other hand, they do not provide efficient verification (with the only exception of [2]) and do not satisfy the zero-knowledge notion of AD-SNARGs that is important for the applications of our interest. It is worth noting that a notion of privacy, called context-hiding, has been considered for homomorphic signatures [9]. However, this notion is weaker than our zero-knowledge as, for instance, it does not allow to hide additional, non-authenticated, witnesses of a computation.

1.3 An Intuitive Description of Our Techniques

The key idea for the construction of our AD-SNARG scheme is to build upon Pinocchio (in particular, its SNARG version) [33] and to modify it by embedding a linearly-homomorphic MAC that

enforces the prover to run Pinocchio's Prove algorithm on correctly authenticated statements. At a more technical level, in Pinocchio the verifier, given a statement $x = (x_1, \ldots, x_a)$, has to compute the linear combination $v_{in} = \sum_{k=1}^{a} x_k \cdot v_k(x)$ (where the $v_k(x)$ are the QAP polynomials)[5]. Since in AD-SNARGs the verifier does not know the statement, our idea is to let the prover compute this linear combination v_{in} on the verifier's behalf. Then, to enforce a cheating prover to provide the correct v_{in}, we ask the prover to additionally show that v_{in} was obtained by using authenticated values x_k. To this end, we employ a linearly-homomorphic MAC.

However, a further complication to applying this technique arises from the fact that v_{in} may be randomized (by adding a random multiple of the target polynomial $t(x)$) in the case the proof is zero-knowledge, while homomorphic MACs typically authenticate only deterministic computations. We solve this issue using the following ideas. First, we provide a way to publicly re-randomize the homomorphic MACs: roughly speaking, by publicly revealing a MAC of $t(x)$. Second, we enforce the prover to use the same random coefficient for $t(x)$ in both v_{in} and its MAC. Very intuitively, this is achieved by asking the prover to provide this linear combination in two different subspaces.

2 Background

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

Notation. We will denote with $\lambda \in \mathbb{N}$ a security parameter. We say that a function ϵ is *negligible* if it vanishes faster than the inverse of any polynomial. If not explicitly specified otherwise, negligible functions are negligible with respect to λ. If S is a set, $x \leftarrow_{\mathcal{R}} S$ denotes the process of selecting x uniformly at random in S. If $\mathcal{A}$ is a probabilistic algorithm, $x \leftarrow_{\mathcal{R}} \mathcal{A}(\cdot)$ denotes the process of running $\mathcal{A}$ on some appropriate input and assigning its output to x. Moreover, for a positive integer n, we denote by $[n]$ the set $\{1, \ldots, n\}$.

Algebraic Tools. Let $\mathcal{G}(1^\lambda)$ be an algorithm that upon input of the security parameter 1^λ, outputs the description of bilinear groups $\mathsf{bgpp} = (p, \mathbb{G}, \mathbb{G}_T, e, g)$ where $\mathbb{G}$ and $\mathbb{G}_T$ are groups of the same prime order $p > 2^\lambda$, $g \in \mathbb{G}$ is a generator and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is an efficiently computable bilinear map. We call such an algorithm $\mathcal{G}$ a *bilinear group generator*. In this work we make use of bilinear groups and in particular we rely on the q-DHE [11], q-BDHE [8] and q-PKE [24] assumptions over these groups.

Arithmetic Circuits and Quadratic Arithmetic Programs. An *arithmetic circuit* C over a finite field $\mathbb{F}$ consists of addition and multiplication *gates* and of a set of *wires* between the gates. The wires carry values over $\mathbb{F}$. A *Quadratic Arithmetic Program* (QAP) [19] encodes the wires of an arithmetic circuit C into three sets of polynomials $\mathcal{V}, \mathcal{W}, \mathcal{Y}$ in such a way that for every multiplication gate $g_\times$ of C, any valid assignment of the circuit wires yields that the left input wires $\mathcal{V}$ of $g_\times$ multiplied by the right input wires $\mathcal{W}$ of $g_\times$ equals the values of the output wires $\mathcal{Y}$ of $g_\times$. More precisely, each polynomial set contains $m + 1$ polynomials of the form

$$\mathcal{V} = \{v_k(x)\}_{k=0\ldots m}, \qquad \mathcal{W} = \{w_k(x)\}_{k=0\ldots m}, \qquad \mathcal{Y} = \{y_k(x)\}_{k=0\ldots m}$$

such that $v_k(r_{g_\times}) = 1$ iff the k-th wire of C is a *left* input to multiplication gate $g_\times$. Each multiplication gate $g_\times$ is thereby represented as an arbitrary number $r_{g_\times} \in \mathbb{F}$, its "root". Dually,

[5] Precisely, the verifier also computes $w_{in} = \sum_{k=1}^{a} x_k \cdot w_k(x)$ and $y_{in} = \sum_{k=1}^{a} x_k \cdot y_k(x)$. In this intuitive description, we simplify and describe our technique only for v_{in}.

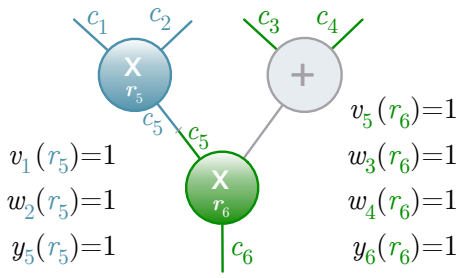

Fig. 2. Two multiplication gates g_5 and g_6 with roots r_5 and r_6.

the polynomials w_k and y_k represent right inputs and outputs, respectively.[6] Figure 2 shows two multiplication gates with corresponding polynomials. The arithmetic constraints for all multiplication gates of C are enforced by virtue of a divisibility check with a specific target polynomial $t(x) = \prod_{g_\times}(x - r_{g_\times})$. More precisely, Q is said to compute C if, whenever $(c_1, \ldots, c_N) \in \mathbb{F}^N$ is a valid assignment of C's input and output wires, then there exists coefficients $(c_{N+1}, \ldots, c_m)$ such that $t(x)$ divides $p(x)$ where

$$p(x) \;=\; \big(v_0(x) + \sum_{k=1}^{m} c_k\, v_k(x)\big) \cdot \big(w_0(x) + \sum_{k=1}^{m} c_k\, w_k(x)\big) - \big(y_0(x) + \sum_{k=1}^{m} c_k\, y_k(x)\big)$$

The divisibility hence implies that all wire assignments are consistent, in particular the output wires of C carry the correct evaluation result of C for the given input wires.

3 Zero-Knowledge SNARGs over Authenticated Data

We define the notion of SNARGs [31,5] on authenticated data (AD-SNARGs, for short). Let $R = \{(x, w)_i\}$ be a relation over $\mathbb{F}^{a+b}$ where $\mathbb{F}$ is a finite field, $x \in \mathbb{F}^a$ is called the *statement*, and $w \in \mathbb{F}^b$ is the *witness*. Proof systems typically consider the problem in which a prover $\mathcal{P}$ proves to a verifier $\mathcal{V}$ the existence of a witness w such that $(x, w) \in R$. In this scenario, the statement x is supposed to be public, i.e., it is known to both the prover and the verifier. For example, $\mathcal{V}$ could be convinced by $\mathcal{P}$ that 3 colors are sufficient to color a public graph x such that no two adjacent vertices are assigned the same color. The coloring serves as witness w.

In this work, we consider a variation of the above problem in which (1) the statement x (or part of it) is provided to the prover by a trusted source $\mathcal{S}$, and (2) the portion of x provided by $\mathcal{S}$ is not known to $\mathcal{V}$ (see Figure 1 for illustration). Yet, $\mathcal{V}$ wants to be convinced by $\mathcal{P}$ that $(x, w) \in R$ holds for the specific x provided by $\mathcal{S}$, and not for some other x' (which can still satisfy R) of $\mathcal{P}$'s choice. For example, $\mathcal{S}$ might have provided a graph x – not known to $\mathcal{V}$ – for which $\mathcal{P}$ proves to $\mathcal{V}$ that x is 3-colorable. A proof for any other graph x' is meaningless.

To formalize the idea that $\mathcal{V}$ checks that some values unknown to $\mathcal{V}$ have been authenticated by $\mathcal{S}$, we adopt the concept of labeling used for homomorphic authentication [20]. Namely, we assume that the source $\mathcal{S}$ authenticates a set of values $X_{auth} = \{c_i, \ldots, c_\ell\}$ against a set of (public) labels $\mathsf{L} = \{\mathsf{L}_i, \ldots, \mathsf{L}_\ell\}$ by using a secret authentication key (e.g., a signing key). $\mathcal{S}$ then sends the

[6] The precise construction is slightly more complex, since it also handles addition and multiplication by constants.

authenticated X_{auth} to $\mathcal{P}$. Later, $\mathcal{P}$'s goal is to prove to $\mathcal{V}$ that $(x, w) \in R$ for a statement x in which some positions have been correctly authenticated by $\mathcal{S}$, i.e., $x_i \in X_{auth}$ for some $i \in [a]$.

For such a proof system we define the usual properties of completeness and soundness, and in addition, to model privacy, zero-knowledge. Moreover, since we are interested in efficient, scalable, protocols, we define succinctness to model that the size of the proofs should be independent of the witness $|w|$.

Finally, we consider AD-SNARGs that can have either public or secret verifiability, the difference being in whether the adversary knows or not the verification key for the authentication tags produced by the data source $\mathcal{S}$.

3.1 SNARGs over Authenticated Data

Here we provide the formal definition for SNARGs over authenticated data.

Definition 1 (AD/SNARG). *A scheme for* Succinct Non-interactive Arguments over Authenticated Data *with respect to a family of relations $\mathcal{R}$ consists of a tuple of algorithms* $(\mathsf{Setup}, \mathsf{AuthKeyGen}, \mathsf{Auth}, \mathsf{AuthVer}, \mathsf{Gen}, \mathsf{Prove}, \mathsf{Ver})$ *satisfying* authentication correctness, completeness, adaptive soundness, *and* succinctness *(as defined below):*

$\mathsf{Setup}(1^\lambda)$**:** *On input the security parameter λ, output some common public parameters* pp.

$\mathsf{AuthKeyGen}(\mathsf{pp})$**:** *given the public parameters* pp*, the key generation algorithm outputs a secret authentication key* sk*, a verification key* vk*, and public authentication parameters* pap.

$\mathsf{Auth}(\mathsf{sk}, \mathsf{L}, c)$**:** *the authentication algorithm takes as input the secret authentication key* sk*, a label $\mathsf{L} \in \mathcal{L}$, and a message $c \in \mathbb{F}$, and it outputs an authentication tag σ.*

$\mathsf{AuthVer}(\mathsf{vk}, \sigma, \mathsf{L}, c)$**:** *the authentication verification algorithm takes as input a verification key* vk*, a tag σ, a label $\mathsf{L} \in \mathcal{L}$, and a message $c \in \mathbb{F}$. It outputs $\perp$ (reject) or $\top$ (accept).*

$\mathsf{Gen}(\mathsf{pap}, R)$**:** *given the public authentication parameters* pap *and a relation $R \subseteq \mathbb{F}^a \times \mathbb{F}^b \in \mathcal{R}$, the algorithm outputs an evaluation key EK_R and a verification key VK_R for R.* Gen *can hence be seen as a relation encoding algorithm.*

$\mathsf{Prove}(\mathsf{EK}_R, x, w, \sigma)$**:** *on input a relation evaluation key EK_R, a statement $x = (x_1, \ldots, x_a)$, a witness $w = (w_1, \ldots, w_b)$, and authentication tags for the statement $\sigma = (\sigma_1, \ldots, \sigma_a)$, the proof algorithm outputs a proof of membership π for $(x, w) \in R$. We stress that σ does not need to contain authentication tags for all positions: in case a value at position i is not authenticated, the empty tag $\sigma_i = \star$ is used instead.*

$\mathsf{Ver}(\mathsf{vk}, \mathsf{VK}_R, \mathsf{L}, \{x_i\}_{\mathsf{L}_i=\star}, \pi)$**:** *given the verification key* vk*, a relation verification key VK_R, labels for the statement $\mathsf{L} = (\mathsf{L}_1, \ldots, \mathsf{L}_a)$, unauthenticated statement components x_i, and a proof π, the verification algorithm outputs $\perp$ (reject) or $\top$ (accept).*

Example (Graph Coloring using AD/SNARG**).** To prove that x is a particular graph with valid 3-coloring, the prover $\mathcal{P}$ uses the Prove algorithm of an AD/SNARG to produce a proof $\pi \leftarrow \mathsf{Prove}(\mathsf{EK}_R, x, w, \sigma)$, where $\mathsf{EK}_R \leftarrow_{\mathcal{R}} \mathsf{Gen}(\mathsf{pap}, R)$, and $\sigma = (\sigma_1, \ldots, \sigma_a)$ are the signatures to authenticate the particular graph x under the labels L. The verifier runs $\mathsf{Ver}(\mathsf{vk}, \mathsf{VK}_R, \mathsf{L}, (), \pi)$ to decide whether the coloring is valid.

Example (Example: Verifiable Computation using AD/SNARG). Let $F : \mathbb{F}^r \to \mathbb{F}^s$ be a function to be executed over authenticated data $x \in \mathbb{F}^r$ with authentication tags σ and corresponding labels L. The worker computes $y = f(x)$ and obtains w as witness of the computation. The relation to be proven using AD/SNARG is $R : \mathbb{F}^{r+s} \times \mathbb{F}^b$ such that $(x||y, w) \in R$ whenever $y = f(x)$. The worker adaptively "extends" the statement by appending the result y of the computation to the input x.
More precisely, in the case of delegated computations over authenticated data, the worker first receives labels $\mathsf{L} = (\mathsf{L}_1, \ldots, \mathsf{L}_r)$, and then fetches the corresponding input $x = (x_1, \ldots, x_r)$ authenticated through $\sigma = (\sigma_1, \ldots, \sigma_r)$. The worker computes $y = f(x)$, obtains witness w and uses $\mathsf{Prove}(\mathsf{EK}_R, x||y, w, \sigma)$ to obtain a proof π. The authentication information in this case is $\sigma = (\sigma_1, \ldots, \sigma_r, \star_1, \ldots, \star_s)$, since there is no authentication for y.
The verifier runs $\mathsf{Ver}(\mathsf{vk}, \mathsf{VK}_R, (\mathsf{L}_1, \ldots, \mathsf{L}_r, \star_1, \ldots, \star_s), y, \pi)$ to convince himself that $y = f(x)$ and that x is indeed the right input, hence the one authenticated via L.

Authentication Correctness. Intuitively, an AD/SNARG scheme has authentication correctness if any tag σ generated by $\mathsf{Auth}(\mathsf{sk}, \mathsf{L}, c)$ authenticates c with respect to L. More formally, we say that an AD/SNARG scheme satisfies authentication correctness if for any message $c \in \mathbb{F}$, all keys $(\mathsf{sk}, \mathsf{vk}, \mathsf{pap}) \leftarrow_{\mathcal{R}} \mathsf{AuthKeyGen}(1^\lambda)$, any label $\mathsf{L} \in \mathcal{L}$, and any authentication tag $\sigma \leftarrow_{\mathcal{R}} \mathsf{Auth}(\mathsf{sk}, \mathsf{L}, c)$, we have that $\mathsf{AuthVer}(\mathsf{vk}, \sigma, \mathsf{L}, c) = \top$ with probability 1. Moreover, we assume $\mathsf{Auth}(\mathsf{sk}, \star, c) = \star$.

Completeness. This property aims at capturing that if the Prove algorithm produces π when run on (x, w, σ) for some $(x, w) \in R$, then verification $\mathsf{Ver}(\mathsf{vk}, \mathsf{VK}_R, \mathsf{L}, \{x_i\}_{\mathsf{L}_i = \star}, \pi)$ must output $\top$ with probability 1 whenever $\mathsf{AuthVer}(\mathsf{vk}, \sigma_i, \mathsf{L}_i, x_i) = \top$. More formally, let us fix $(\mathsf{sk}, \mathsf{vk}, \mathsf{pap}) \leftarrow_{\mathcal{R}} \mathsf{AuthKeyGen}(\mathsf{pp})$, and a relation $R : \mathbb{F}^a \times \mathbb{F}^b$ with keys $(\mathsf{EK}_R, \mathsf{VK}_R) \leftarrow_{\mathcal{R}} \mathsf{Gen}(\mathsf{pap}, R)$. Let $(x, w) \in R$ be given with $x = (x_1, \ldots, x_a)$, $w = (w_1, \ldots, w_b)$. Let $\mathsf{L} = (\mathsf{L}_1, \ldots, \mathsf{L}_a) \in (\mathcal{L} \cup \{\star\})^a$, be a set of labels, and let $\sigma = (\sigma_1, \ldots, \sigma_a)$ be tags for the statement such that $\mathsf{AuthVer}(\mathsf{vk}, \sigma_i, \mathsf{L}_i, x_i) = \top$. Then if $\pi \leftarrow_{\mathcal{R}} \mathsf{Prove}(\mathsf{EK}_R, x, w, \sigma)$, we have that $\mathsf{Ver}(\mathsf{vk}, \mathsf{VK}_R, \mathsf{L}, \{x_i\}_{\mathsf{L}_i = \star}, \pi) = \top$ with probability 1.

Adaptive Soundness. Intuitively, the soundness property captures that no malicious party can produce proofs that verify correctly for tuples not contained in the relation. More formally, we formalize our definition via an experiment, called $\mathbf{Exp}_{\mathcal{A}}^{\mathsf{AD/Soundness}}$, which is described in Figure 3, using the notation of code-based games. The game is defined by a number of procedures that can be run by an adversary $\mathcal{A}$ as follows. As usual, the game starts by once executing `Initialize` and terminates with once executing `Finalize`. In between, $\mathcal{A}$ can (concurrently) run the procedures `Gen`, `Auth`, and `Ver`. We define the output of the game to be the output of the `Finalize` procedure. The three procedures `Gen`, `Auth`, and `Ver` essentially give to the adversary oracle access to the algorithms Gen, Auth, and Ver, respectively, with some additional bookkeeping information. In particular, it is worth noting that `Ver` returns the output of Ver, and additionally, checks whether a proof accepted by Ver (i.e., $v = \top$) proves a false statement according to R. In this case, `Ver` sets $\texttt{GameOutput} \leftarrow 1$.

We say that an adversary $\mathcal{A}$ *wins the game* if it manages to make the experiment $\mathbf{Exp}_{\mathcal{A}}^{\mathsf{AD/Soundness}}$ output 1, i.e., if it ever asks a verification query that sets $\texttt{GameOutput} \leftarrow 1$. More formally, let $\mathcal{R}$ be a class of relations. Then for any $\lambda \in \mathbb{N}$, we define the advantage of an adversary $\mathcal{A}$ in the experiment $\mathbf{Exp}_{\mathcal{A}}^{\mathsf{AD/Soundness}}(\mathcal{R}, \lambda)$ against AD/Soundness for $\mathcal{R}$ as

$$\mathbf{Adv}_{\mathcal{A}}^{\mathsf{AD/Soundness}}(\mathcal{R}, \lambda) \;=\; \Pr[\mathbf{Exp}_{\mathcal{A}}^{\mathsf{AD/Soundness}}(\mathcal{R}, \lambda) = 1].$$

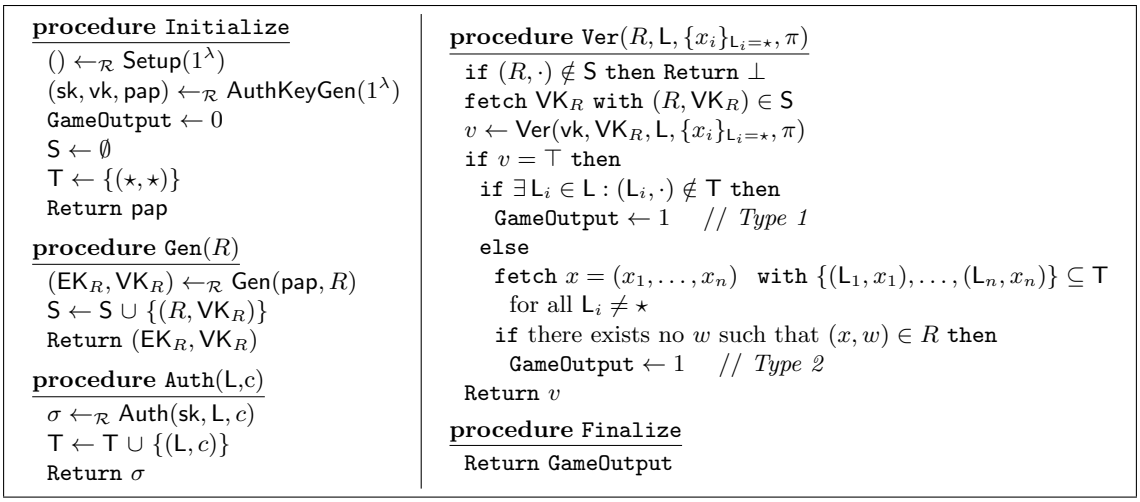

Fig. 3. Game AD/Soundness.

An AD/SNARG over authenticated data with respect to a class of relations $\mathcal{R}$ is *computationally sound* if for any PPT $\mathcal{A}$, it holds that $\mathbf{Adv}^{\mathsf{AD/Soundness}}_{\mathcal{A}}(\mathcal{R}, \lambda)$ is negligible in λ.

Our soundness definition is inspired by the security definition for homomorphic MACs [20,12,2]. The catch here is that there are essentially two ways to create a "cheating proof", and thus to break the soundness of an AD/SNARG. The first way, Type 1, is to produce an accepting proof without having ever queried an authentication tag for a label L_i. This basically captures that, in order to create a valid proof, one needs to have all authenticated parts of the statement, each with a valid authentication tag. The second way to break the security, Type 2, is the more "classical" one, i.e., generating a proof that accepts for a tuple (x, w) which is not the correct one, i.e., $(x, \cdot) \notin R$.

Second, we note that the above game definition captures the setting in which the verification key vk is kept secret. The definition for the publicly verifiable setting is easily obtained by having `Initialize` return vk to the adversary.

Succinctness. Given a relation $R : \mathbb{F}^a \times \mathbb{F}^b$, the length of π is bound by $|\pi| = \mathrm{poly}(\lambda)\mathrm{polylog}(a, b)$.

Zero-Knowledge. Loosely speaking, an AD/SNARG is *zero-knowledge* if the Prove algorithm generates proofs π that reveal no information: neither about the *witness* of the relation, nor about the authenticated statements. In other words, the proofs do not reveal anything beyond what is known by the verifiers when checking a proof. A formal definition follows:

Definition 2 (Zero-Knowledge AD/SNARG). *A scheme* AD/SNARG *is a* zero-knowledge *SNARG over authenticated data if the following additional property "*Zero-Knowledge*" holds. Let $R \in \mathcal{R}$ be any relation. Then there exists a simulator $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2)$, such that for all PPT distinguishers $\mathcal{D}$, the following difference is negligible*

$$|\Pr[\mathbf{Exp}^{\mathcal{D},R}_{Real}(\lambda) = 1] - \Pr[\mathbf{Exp}^{\mathcal{D},R}_{Sim}(\lambda) = 1]|$$

where the experiments Real and Sim are defined as as follows:

$\mathbf{Exp}^{\mathcal{D},R}_{Real}(\lambda)$:
$\mathsf{pp} \leftarrow_{\mathcal{R}} \mathsf{Setup}(1^\lambda)$
$(\mathsf{sk}, \mathsf{vk}, \mathsf{pap}) \leftarrow_{\mathcal{R}} \mathcal{D}(1^\lambda, \mathsf{pp})$
$(\mathsf{EK}_R, \mathsf{VK}_R) \leftarrow_{\mathcal{R}} \mathsf{Gen}(\mathsf{pap}, R)$
$(x, \mathsf{L}, \sigma) = \{(x_i, \mathsf{L}_i, \sigma_i)\}_{i=1}^{a} \leftarrow_{\mathcal{R}} \mathcal{D}(\mathsf{EK}_R, \mathsf{VK}_R)$
$w \leftarrow_{\mathcal{R}} \mathcal{D}(\mathsf{EK}_R, \mathsf{VK}_R)$
$\pi \leftarrow_{\mathcal{R}} \mathsf{Prove}(\mathsf{EK}_R, x, w, \sigma)$
if $\mathcal{D}(\pi) = 1 \wedge \{\mathsf{AuthVer}(\mathsf{vk}, \sigma_i, \mathsf{L}_i, x_i) = \top\}_{i=1}^{a}$
$\wedge\ (x, w) \in R$
output 1

$\mathbf{Exp}^{\mathcal{D},R}_{Sim}(\lambda)$:
$\mathsf{pp} \leftarrow_{\mathcal{R}} \mathsf{Setup}(1^\lambda)$
$(\mathsf{sk}, \mathsf{vk}, \mathsf{pap}) \leftarrow_{\mathcal{R}} \mathcal{D}(1^\lambda, \mathsf{pp})$
$(\mathsf{EK}_R, \mathsf{VK}_R, \mathsf{td}) \leftarrow_{\mathcal{R}} \mathsf{Sim}_1(\mathsf{sk}, \mathsf{vk}, \mathsf{pp}, \mathsf{pap}, R)$
$(x, \mathsf{L}, \sigma) = \{(x_i, \mathsf{L}_i, \sigma_i)\}_{i=1}^{a} \leftarrow_{\mathcal{R}} \mathcal{D}(\mathsf{EK}_R, \mathsf{VK}_R)$

$\pi \leftarrow_{\mathcal{R}} \mathsf{Sim}_2(\mathsf{td}, \mathsf{L}, \{x_i\}_{\mathsf{L}_i = \star})$
if $\mathcal{D}(\pi) = 1 \wedge \{\mathsf{AuthVer}(\mathsf{vk}, \sigma_i, \mathsf{L}_i, x_i) = \top\}_{i=1}^{a}$
$\wedge\ (x, \cdot) \in R$
output 1

Note that the distinguisher $\mathcal{D}$ in the above game has a shared state that is persistent over all invocations of $\mathcal{D}$ during an experiment.

We stress that the above zero-knowledge notion aims at capturing, in the strongest possible sense, that the verifier cannot learn any useful information on the inputs, *even if it knows (or chooses) the secret authentication key*. Indeed, as one can see, our definition allows the distinguisher to choose the authentication key pair as well as the authentication tags.

Interestingly, we note that the notion of AD-SNARGs immediately implies a corresponding notion of *verifiable computation on authenticated data* (similar to [2]). In [5] it is discussed how to construct a verifiable computation scheme from SNARGs for $\mathcal{NP}$ with adaptive soundness. This is simply based on the fact that the correctness of a computation can be described with an $\mathcal{NP}$ statement. It is not hard to see that, in a very similar way, one can construct verifiable computation on authenticated data from AD-SNARGs.

3.2 A Generic Construction of AD-SNARGs

We show how to construct an AD-SNARG scheme from SNARKs and digital signatures. A similar construction was informally sketched in [5][Appendix 10.1.2 of the full version]. Here we make it more formal with the main purpose of offering a comparison with our direct AD-SNARG constructions proposed in the next sections.

Let $\Pi' = (\mathsf{Gen}', \mathsf{Prove}', \mathsf{Ver}')$ be a SNARK scheme, and $\Sigma = (\Sigma.\mathsf{KG}, \Sigma.\mathsf{Sign}, \Sigma.\mathsf{Ver})$ be a signature scheme. We will use the signature scheme to sign pairs consisting of a label L and an actual message m. Although, labels and messages can be arbitrary binary strings, for ease of description we assume that labels can take a special value $\star$. Also, we modify the signature scheme in such a way that $\Sigma.\mathsf{Sign}(\mathsf{sk}, \star|m) = \star$ and $\Sigma.\mathsf{Ver}(\mathsf{vk}, \star|m', \star) = 1$. Basically, we let everyone (trivially) generate a valid signature on a message with label $\star$.

We define an AD⁄SNARG $\Pi = (\mathsf{Setup}, \mathsf{AuthKeyGen}, \mathsf{Auth}, \mathsf{AuthVer}, \mathsf{Prove}, \mathsf{Ver})$ as follows.

$\mathsf{Setup}(1^\lambda)$**:** Output $\mathsf{pp} = 1^\lambda$.

$\mathsf{AuthKeyGen}(\mathsf{pp})$**:** run $(\mathsf{sk}', \mathsf{vk}') \leftarrow_{\mathcal{R}} \Sigma.\mathsf{KG}(1^\lambda)$ to generate the key pair of the signature scheme and return $\mathsf{sk} = \mathsf{sk}'$ and $\mathsf{vk} = \mathsf{pap} = \mathsf{vk}'$.

$\mathsf{Auth}(\mathsf{sk}, \mathsf{L}, c)$**:** compute a signature on the concatenation of the label L and the value c, i.e., $\sigma' \leftarrow_{\mathcal{R}} \Sigma.\mathsf{Sign}(\mathsf{sk}', \mathsf{L}|c)$. Finally, output $\sigma = (\sigma', \mathsf{L})$.

$\mathsf{AuthVer}(\mathsf{vk}, \sigma, \mathsf{L}, c)$**:** let $\sigma = (\sigma', L')$, output the result of the signature verification algorithm $\mathsf{Ver}'(\mathsf{vk}', \mathsf{L}|c, \sigma')$.

$\mathsf{Gen}(\mathsf{pap}, R)$**:** informally, we define R' as the relation that contains all the $(x, w) \in R$ such that x is correctly signed with respect to a set of labels and a public key. More formally, define R' as the relation that contains all the tuples (x', w') with $x' = (y_1, \mathsf{L}_1, \ldots, y_a, \mathsf{L}_a, \mathsf{vk})$ and $w' = (w, z_1, \sigma_1, \ldots, z_a, \sigma_a)$ such that, by setting $x_i = y_i$ if $\mathsf{L}_i = \star$ and $x_i = z_i$ otherwise, for all $i \in [a]$, it holds: (i) $((x_1, \ldots, x_a), w) \in R$, and (ii) $\Sigma.\mathsf{Ver}(\mathsf{vk}, \mathsf{L}_i | x_i, \sigma_i) = 1$.
Then, run $\mathsf{Gen}'(1^\lambda, R')$ to generate $(\mathsf{EK}'_{R'}, \mathsf{VK}'_{R'})$ and output $\mathsf{EK}_R = \mathsf{EK}'_{R'}$, $\mathsf{VK}_R = \mathsf{VK}'_{R'}$.

$\mathsf{Prove}(\mathsf{EK}_R, x, w, \sigma)$**:** Let EK_R the be evaluation key as defined above, (x, w) be a statement-witness pair for R, and $\sigma = (\sigma_1, \ldots, \sigma_a)$ be a tuple of authentication tags for $x = (x_1, \ldots, x_a)$.
If all the tags verify correctly, define $x' = (y_1, \mathsf{L}_1, \ldots, y_a, \mathsf{L}_a, \mathsf{vk})$, $w' = (w, z_1, \sigma_1, \ldots, z_a, \sigma_a)$ so that for all $i \in [a]$: $z_i = x_i$, $y_i = x_i$ if $\sigma_i = \star$ and $y_i = 0$ otherwise. Next, run $\pi \leftarrow_{\mathcal{R}} \mathsf{Prove}(\mathsf{EK}'_{R'}, x', w')$ to generate a proof for $(x', w') \in R'$ and return π.

$\mathsf{Ver}(\mathsf{vk}, \mathsf{VK}_R, \mathsf{L}, \{x_i\}_{\mathsf{L}_i = \star}, \pi)$**:** given the verification key vk, a relation verification key VK_R, labels for the statement $\mathsf{L} = (\mathsf{L}_1, \ldots, \mathsf{L}_a)$, unauthenticated statement components x_i, and a proof π, the verification algorithm defines $x' = (y_1, \mathsf{L}_1, \ldots, y_a, \mathsf{L}_a, \mathsf{vk})$ with $y_i = x_i$ if $\mathsf{L}_i = \star$ and $y_i = 0$ otherwise. Finally, it returns the output of $\mathsf{Ver}'(\mathsf{VK}'_{R'}, x', \pi)$.

Theorem 1. *If Π' is a zero-knowledge SNARK and Σ is a secure digital signature, then the scheme described above is a zero-knowledge* AD/SNARG.

Proof (Sketch). We provide a proof sketch to show that the above construction satisfies all the properties. First, it is easy to see that if the SNARK is succinct, then the AD/SNARG proofs are succinct as well. Moreover, authentication correctness and completeness immediately follows from the correctness of the signature scheme and the completeness of the SNARK respectively.

Second, to see adaptive soundness note that for every accepting proof produced by the adversary we can extract the corresponding witness (since Π' is an argument of knowledge). Such proof, by definition, will contain a set of valid signatures. Then, if any of these signatures was not obtained from a query to the `Auth` oracle, then it is easy that it can be used as a forgery to break the unforgeability of the signature scheme. In the case all the signatures are valid, then one can extract the full statement $(x_1, \ldots, x_a)$ from the witness w'. Hence, any adversary who outputs an invalid proof for the AD/SNARG can be immediately turned into an adversary against the adaptive soundness of Π'.

Third, the zero-knowledge of the AD/SNARG follows from the one of the SNARK in a straightforward way.

4 Our Construction of Zero-Knowledge AD-SNARGs

In this section we describe our construction of an AD-SNARG scheme for arbitrary $\mathcal{NP}$ relations. The presented scheme can be used with either secret or public verifiability. The main difference between the two verification modes is that the size of the proof in the secretly verifiable case is a fixed constant, whereas in the publicly verifiable case the proof grows linearly with the number of authenticated statement values. Although we loose constant-size proofs for public verifiability, we stress that proofs become linear only in the number N of authenticated values, and that the verification algorithm runs linearly in N in any case (even in the generic construction). Furthermore, for verifiers that know the secret authentication key (as it may be the case for smart metering where companies install the keys in the meters) the proofs can be maintained of constant size, and – importantly – revealing such secret key does not compromise privacy. We prove our scheme adaptive

sound under two computational assumptions in bilinear groups, the q-Diffie-Hellman Exponent assumption (q-DHE) [11] and the q-Power Knowledge of Exponent assumption (q-PKE) [24]. We note that the latter one is a non-falsifiable assumption. As discussed in the Introduction this kind of assumptions is likely to be inherent for SNARGs for $\mathcal{NP}$. For privacy, we show that the scheme has statistical zero-knowledge and we stress that this property holds even against adversaries who know (and even generate) the authentication keys.

A detailed description of our scheme follows.

$\mathsf{Setup}(1^\lambda)$**:** On input the security parameter 1^λ, run $\mathsf{pp} = (p, \mathbb{G}, \mathbb{G}_T, e, g) \leftarrow_{\mathcal{R}} \mathcal{G}(1^\lambda)$ to generate a bilinear group description, where $\mathbb{G}$ and $\mathbb{G}_T$ are groups of the same prime order $p > 2^\lambda$, $g \in \mathbb{G}$ is a generator and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is an efficiently computable bilinear map.

$\mathsf{AuthKeyGen}(\mathsf{pp})$**:** Generate a key pair $(\mathsf{sk}', \mathsf{vk}') \leftarrow_{\mathcal{R}} \Sigma.\mathsf{KG}(1^\lambda)$ for a regular signature scheme. Run $(S, \mathsf{prfpp}) \leftarrow_{\mathcal{R}} \mathsf{F}.\mathsf{KG}(1^\lambda)$ to obtain the seed S and the public parameters prfpp of a pseudorandom function $\mathsf{F}_S : \{0,1\}^* \to \mathbb{F}$. Choose a random value $z \leftarrow_{\mathcal{R}} \mathbb{F}$. Compute $Z = g^z \in \mathbb{G}$. Return the secret key $\mathsf{sk} = (\mathsf{sk}', S, z)$, the public verification key $\mathsf{vk} = (\mathsf{vk}', Z)$ and the public authentication parameters $\mathsf{pap} = (\mathsf{pp}, \mathsf{prfpp}, Z)$.

$\mathsf{Auth}(\mathsf{sk}, \mathsf{L}, c)$**:** To authenticate a value $c \in \mathbb{F}$ with label L, generate $\lambda \leftarrow \mathsf{F}_S(\mathsf{L})$ using the PRF, compute $\mu = \lambda + z \cdot c$ and $\Lambda = g^\lambda$. Then compute a signature $\sigma' \leftarrow_{\mathcal{R}} \Sigma.\mathsf{Sign}(\mathsf{sk}', \Lambda|\mathsf{L})$, and output the tag $\sigma = (\mu, \Lambda, \sigma')$.

$\mathsf{AuthVer}(\mathsf{vk}, \sigma, \mathsf{L}, c)$**:** Let $\mathsf{vk} = (\mathsf{vk}', Z)$ be the verification key. To verify that $\sigma = (\mu, \Lambda, \sigma')$ is a valid authentication tag for a value $c \in \mathbb{F}$ with respect to label L, output $\top$ if $g^\mu = \Lambda \cdot Z^c$ and $\Sigma.\mathsf{Ver}(\mathsf{vk}', \Lambda|\mathsf{L}, \sigma') = 1$. Output $\perp$ otherwise. In the secret key setting (i.e., if vk is replaced by sk), the tag can be verified by checking whether $\mu = \mathsf{F}_S(\mathsf{L}) + zc$.

$\mathsf{Gen}(\mathsf{pap}, R)$**:** Let $R : \mathbb{F}^a \times \mathbb{F}^b$ be an $\mathcal{NP}$ relation with statements of length a and witnesses of length b. Let C_R be R's characteristic circuit, i.e., $C_R(x, w) = 1$ iff $(x, w) \in R$. Build a QAP $Q_R = (t(x), \mathcal{V}, \mathcal{W}, \mathcal{Y})$ of size m and degree d for C_R. We denote by I_{st}, I_{mid}, I_{out} the following partitions of $\{1, \ldots, m\}$: $I_{st} = \{1, \ldots, a\}$, $I_{mid} = \{a+1, \ldots, m-1\}$ and $I_{out} = \{m\}$.[7] In other words, we partition all the circuit wires into: statement wires I_{st}, internal wires I_{mid} (including the witness wires), and the output wire I_{out}.
Next, pick $r_v, r_w \leftarrow_{\mathcal{R}} \mathbb{F}$ uniformly at random and set $r_y = r_v\, r_w$. Then pick $s, \alpha_v, \alpha_w, \alpha_y, \beta, \gamma \leftarrow_{\mathcal{R}} \mathbb{F}$ uniformly at random and compute the following values:

$$
\begin{array}{lllll}
 & T = g^{r_y t(s)} & & & \\
\forall k \in [m] \cup \{0\} : & V_k = g^{r_v v_k(s)}, & W_k = g^{r_w w_k(s)}, & Y_k = g^{r_y y_k(s)}, & \\
\forall k \in [m] : & V'_k = (V_k)^{\alpha_v}, & W'_k = (W_k)^{\alpha_w}, & Y'_k = (Y_k)^{\alpha_y}, & B_k = (V_k\, W_k\, Y_k)^\beta.
\end{array}
$$

Additionally, compute the following values:

$$
\begin{array}{lll}
\rho_v = Z^{r_v\, t(s)}, & \rho_w = Z^{r_w\, t(s)}, & \rho_y = Z^{r_y\, t(s)}, \\
V_t = g^{r_v\, t(s)}, & W_t = g^{r_w\, t(s)}, & Y_t = g^{r_y\, t(s)}, \\
V'_t = (V_t)^{\alpha_v}, & W'_t = (W_t)^{\alpha_w}, & Y'_t = (Y_t)^{\alpha_y}, \\
B_v = (V_t)^\beta, & B_w = (W_t)^\beta, & B_y = (Y_t)^\beta.
\end{array}
$$

[7] For a reader familiar with Pinocchio, we point out our change of notation: we will use v_{st} instead of v_{in} to refer to the statement-related inputs.

Output the *evaluation key* EK_R and the *verification key* VK_R defined as follows:

$$\mathsf{EK}_R = \Big(\{V_k, V'_k, W_k, W'_k, Y_k, Y'_k, B_k\}_{k\in I_{st}\cup I_{mid}},\ \{g^{s^i}\}_{i\in[d]} \\ V_t, V'_t,\ W_t, W'_t,\ Y_t, Y'_t,\ B_v, B_w, B_y,\ \rho_v, \rho_w, \rho_y,\ Q_R \Big)$$

$$\mathsf{VK}_R = \Big(g,\ g^{\alpha_v}, g^{\alpha_w}, g^{\alpha_y}, g^{\gamma},\ g^{\beta\gamma},\ T,\ \{V_k, W_k, Y_k\}_{k\in I_{st}\cup\{0,m\}} \Big)$$

$\mathsf{Prove}(\mathsf{EK}_R, x, w, \sigma)$**:** Let EK_R the be evaluation key as defined above, $(x, w) \in \mathbb{F}^a \times \mathbb{F}^b$ be a statement-witness pair, and $\sigma = (\sigma_1, \ldots, \sigma_a)$ be a tuple of authentication tags for $x = (x_1, \ldots, x_a)$ such that for any $i \in [a]$ either $\sigma_i = (\mu_i, \Lambda_i, \sigma'_i)$ or $\sigma_i = \star$. We define $I_{at} = \{i \in I_{st} : \sigma_i \neq \star\} \subseteq I_{st}$ as the set of indices for which there is an authenticated statement value, and let $I_{un} = I_{st} \setminus I_{at}$ be its complement.

To produce a proof for $(x, w) \in R$ proceed as follows. First, evaluate the circuit $C_R(x, w)$ and learn the values of all internal wires: $\{c_k\}_{k\in I_{mid}}$. For ease of description, we assume $c_i = x_i$ for $i \in [a]$, and $c_{a+i} = w_i$ for $i \in [b]$. The first b indices of I_{mid} hence represent the witness values w.

Next, proceed as follows to compute the proof:

$$\begin{aligned}
V_{at} &= \prod_{k\in I_{at}} (V_k)^{c_k}, \quad W_{at} = \prod_{k\in I_{at}} (W_k)^{c_k}, \quad Y_{at} = \prod_{k\in I_{at}} (Y_k)^{c_k}, \\
V'_{at} &= \prod_{k\in I_{at}} (V'_k)^{c_k}, \quad W'_{at} = \prod_{k\in I_{at}} (W'_k)^{c_k}, \quad Y'_{at} = \prod_{k\in I_{at}} (Y'_k)^{c_k}, \\
V_{mid} &= \prod_{k\in I_{mid}} (V_k)^{c_k}, \quad W_{mid} = \prod_{k\in I_{mid}} (W_k)^{c_k}, \quad Y_{mid} = \prod_{k\in I_{mid}} (Y_k)^{c_k}, \\
V'_{mid} &= \prod_{k\in I_{mid}} (V'_k)^{c_k}, \quad W'_{mid} = \prod_{k\in I_{mid}} (W'_k)^{c_k}, \quad Y'_{mid} = \prod_{k\in I_{mid}} (Y'_k)^{c_k}, \\
B_{mid} &= \prod_{k\in I_{mid}} (B_k)^{c_k}.
\end{aligned}$$

Authenticate the values V_{at}, W_{at}, and Y_{at} by computing $\hat{\mu}_v = \prod_{k\in I_{at}} (V_k)^{\mu_k}$, $\hat{\mu}_w = \prod_{k\in I_{at}} (W_k)^{\mu_k}$, and $\hat{\mu}_y = \prod_{k\in I_{at}} (Y_k)^{\mu_k}$, respectively.

To make the proof zero-knowledge, pick random values $\delta^{(v)}_{at}, \delta^{(v)}_{mid}, \delta^{(w)}_{at}, \delta^{(w)}_{mid}, \delta^{(y)}_{at}, \delta^{(y)}_{mid} \leftarrow_{\mathcal{R}} \mathbb{F}$, and compute:

$$\begin{aligned}
\tilde{V}_{at} &= V_{at} \cdot (V_t)^{\delta^{(v)}_{at}}, \quad \tilde{W}_{at} = W_{at} \cdot (W_t)^{\delta^{(w)}_{at}}, \quad \tilde{Y}_{at} = Y_{at} \cdot (Y_t)^{\delta^{(y)}_{at}}, \\
\tilde{V}'_{at} &= V'_{at} \cdot (V'_t)^{\delta^{(v)}_{at}}, \quad \tilde{W}'_{at} = W'_{at} \cdot (W'_t)^{\delta^{(w)}_{at}}, \quad \tilde{Y}'_{at} = Y'_{at} \cdot (Y'_t)^{\delta^{(y)}_{at}}, \\
\tilde{V}_{mid} &= V_{mid} \cdot (V_t)^{\delta^{(v)}_{mid}}, \quad \tilde{W}_{mid} = W_{mid} \cdot (W_t)^{\delta^{(w)}_{mid}}, \quad \tilde{Y}_{mid} = Y_{mid} \cdot (Y_t)^{\delta^{(y)}_{mid}}, \\
\tilde{V}'_{mid} &= V'_{mid} \cdot (V'_t)^{\delta^{(v)}_{mid}}, \quad \tilde{W}'_{mid} = W'_{mid} \cdot (W'_t)^{\delta^{(w)}_{mid}}, \quad \tilde{Y}'_{mid} = Y'_{mid} \cdot (Y'_t)^{\delta^{(y)}_{mid}}, \\
\tilde{B}_{mid} &= B_{mid} \cdot (B_v)^{\delta^{(v)}_{mid}} \cdot (B_w)^{\delta^{(w)}_{mid}} \cdot (B_y)^{\delta^{(y)}_{mid}}
\end{aligned}$$

To authenticate the new values $\tilde{V}_{at}$, $\tilde{W}_{at}$, and $\tilde{Y}_{at}$, compute $\tilde{\mu}_v = \hat{\mu}_v \cdot (\rho_v)^{\delta^{(v)}_{at}}$, $\tilde{\mu}_w = \hat{\mu}_w \cdot (\rho_w)^{\delta^{(w)}_{at}}$, and $\tilde{\mu}_y = \hat{\mu}_y \cdot (\rho_y)^{\delta^{(y)}_{at}}$, respectively. Note that our technique preserves the re-randomization property of Pinocchio.

Next, solve the QAP Q_R by finding a polynomial $\tilde{h}(x)$ such that $\tilde{p}(x) = \tilde{h}(x) \cdot t(x)$ where the polynomial $\tilde{p}(x)$ includes the "perturbed versions" of the polynomials $v(x)$, $w(x)$, and $y(x)$ with $\delta^{(v)} = \delta^{(v)}_{at} + \delta^{(v)}_{mid}$, $\delta^{(w)} = \delta^{(w)}_{at} + \delta^{(w)}_{mid}$, and $\delta^{(y)} = \delta^{(y)}_{at} + \delta^{(y)}_{mid}$, respectively:

$$\tilde{p}(x) = \Big(v_0(x) + \sum_{k\in[m]} c_k v_k(x) \;+\; \delta^{(v)} t(x)\Big)\Big(w_0(x) + \sum_{k\in[m]} c_k w_k(x) \;+\; \delta^{(w)} t(x)\Big) - \Big(y_0(x) + \sum_{k\in[m]} c_k y_k(x) \;+\; \delta^{(y)} t(x)\Big)$$

Finally, compute $\tilde{H} = g^{\tilde{h}(s)}$ using the values g^{s^i} contained in the evaluation key EK_R. Output $\tilde{\pi} = (\tilde{\mu}_v, \tilde{\mu}_w, \tilde{\mu}_y,\ \tilde{V}_{at}, \tilde{V}'_{at}, \tilde{V}_{mid}, \tilde{V}'_{mid},\ \tilde{W}_{at}, \tilde{W}'_{at}, \tilde{W}_{mid}, \tilde{W}'_{mid},\ \tilde{Y}_{at}, \tilde{Y}'_{at}, \tilde{Y}_{mid}, \tilde{Y}'_{mid},\quad \tilde{B}_{mid}, \tilde{H})$. To make the proof publicly verifiable, include also $\{\Lambda_k, \sigma'_k\}_{\mathsf{L}_k \neq \star}$ in $\tilde{\pi}$.

$\mathsf{Ver}(\mathsf{vk}, \mathsf{VK}_R, \mathsf{L}, \{x_i\}_{\mathsf{L}_i=\star}, \tilde{\pi})$**:** Let VK_R be the verification key for relation R, $\mathsf{L} = (\mathsf{L}_1, \ldots, \mathsf{L}_a)$ be a vector of labels, and let $\tilde{\pi}$ be a proof as defined above. In a similar way as in Prove, we define $I_{at} = \{i \in I_{st} : \mathsf{L}_i \neq \star\} \subseteq I_{st}$ and $I_{un} = I_{st} \setminus I_{at}$. The verification algorithm proceeds as follows:

($A.1^{secret}$) If verification is done using the secret key $\mathsf{sk} = (S, z)$, check the authenticity of $\tilde{V}_{at}$, $\tilde{W}_{at}$ and $\tilde{Y}_{at}$ against the labels L:

$$\tilde{\mu}_v = \Big[\prod_{k\in I_{at}} (V_k)^{\mathsf{F}_S(\mathsf{L}_k)}\Big] \cdot (\tilde{V}_{at})^z$$
$$\wedge\ \tilde{\mu}_w = \Big[\prod_{k\in I_{at}} (W_k)^{\mathsf{F}_S(\mathsf{L}_k)}\Big] \cdot (\tilde{W}_{at})^z$$
$$\wedge\ \tilde{\mu}_y = \Big[\prod_{k\in I_{at}} (Y_k)^{\mathsf{F}_S(\mathsf{L}_k)}\Big] \cdot (\tilde{Y}_{at})^z$$

($A.1^{pub}$) If the verification is performed using the public verification key $\mathsf{vk} = (Z, \mathsf{vk}')$, first check the validity of all Λ_k by checking that $\Sigma.\mathsf{Ver}(\mathsf{vk}', \Lambda_k | \mathsf{L}_k, \sigma'_k) = 1$ for all $k \in I_{at}$. Then check the authenticity of $\tilde{V}_{at}$, $\tilde{W}_{at}$, and $\tilde{Y}_{at}$:

$$e(\tilde{\mu}_v, g) = \Big[\prod_{k\in I_{at}} e(V_k, \Lambda_k)\Big] \cdot e(\tilde{V}_{at}, Z)$$
$$\wedge\ e(\tilde{\mu}_w, g) = \Big[\prod_{k\in I_{at}} e(W_k, \Lambda_k)\Big] \cdot e(\tilde{W}_{at}, Z)$$
$$\wedge\ e(\tilde{\mu}_y, g) = \Big[\prod_{k\in I_{at}} e(Y_k, \Lambda_k)\Big] \cdot e(\tilde{Y}_{at}, Z)$$

($A.2$) Check that $\tilde{V}_{at}$, $\tilde{V}'_{at}$, $\tilde{W}_{at}$, $\tilde{W}'_{at}$, and $\tilde{Y}_{at}$, $\tilde{Y}'_{at}$ were computed using the same linear combination:

$$e(\tilde{V}'_{at}, g) = e(\tilde{V}_{at}, g^{\alpha_v}) \ \wedge\ e(\tilde{W}'_{at}, g) = e(\tilde{W}_{at}, g^{\alpha_w}) \ \wedge\ e(\tilde{Y}'_{at}, g) = e(\tilde{Y}_{at}, g^{\alpha_y})$$

($P.1$) Check the satisfiability of the QAP by setting $V_{out} = (V_m)^{c_m} = V_m$ (similarly $W_{out} = W_m$ and $Y_{out} = Y_m$), where we assume that $c_m = 1 = C_R(x, w)$ since $(x, w) \in R$, then computing $V_{un} = \prod_{k\in I_{un}} (V_k)^{x_k}$ (and similarly W_{un}, Y_{un}), and finally checking:

$$e(V_0\, \tilde{V}_{at}\ V_{un}\ \tilde{V}_{mid}\, V_{out},\ W_0\, \tilde{W}_{at}\ W_{un}\ \tilde{W}_{mid}\, W_{out}) = e(T, \tilde{H}) \cdot e(Y_0\, \tilde{Y}_{at}\ Y_{un}\ \tilde{Y}_{mid}\, Y_{out},\ g)$$

($P.2$) Check that all linear combinations are in the appropriate spans:

$$e(\tilde{V}'_{mid}, g) = e(\tilde{V}_{mid}, g^{\alpha_v}) \ \wedge \ e(\tilde{W}'_{mid}, g) = e(\tilde{W}_{mid}, g^{\alpha_w}) \ \wedge \ e(\tilde{Y}'_{mid}, g) = e(\tilde{Y}_{mid}, g^{\alpha_y})$$

($P.3$) Check that all the QAP linear combinations use the same coefficients:

$$e(\tilde{B}_{mid}, g^{\gamma}) = e(\tilde{V}_{mid}\ \tilde{W}_{mid}\ \tilde{Y}_{mid}, g^{\beta\gamma})$$

If all the checks above are satisfied, then return $\top$; otherwise return $\bot$.

Performance and Comparison. Before proving correctness, soundness, and zero-knowledge, we compare the performance of our construction to Pinocchio [33] (more precisely, to its SNARG version, which for convenience is recalled in Appendix B). First, we note that the generation of the keys is essentially the same except for the three exponentiations for creating ρ_v, ρ_w, ρ_y. Second, in Prove our scheme additionally computes the proof values $\tilde{V}_{at}$, $\tilde{V}'_{at}$, and $\tilde{\mu}_v$ (and the similar ones for W and Y), whose generation cost amounts to 9 multi-exponentiations with $N = |I_{at}|$ terms. Third, in Ver, the difference lies in the realm of authenticated statements: equation (P1) in Pinocchio computes $\prod_{k\in I_{st}}(V_k)^{c_k}$, $\prod_{k\in I_{st}}(W_k)^{c_k}$ and $\prod_{k\in I_{st}}(Y_k)^{c_k}$ for *all* the $a = |I_{st}|$ statement values, whereas in our scheme we only compute those multi-exponentiations over I_{un} (of size $a - N$) and – in the secretly verifiable case – move the checks for the *authenticated* statements, three multi-exponentiations (of size N), to equation $(\text{A.1})^{secret}$. Hence, the total cost of running ($P.1$) and $(\text{A.1})^{secret}$ in our scheme is essentially the same as ($P.1$) in Pinocchio. In the publicly verifiable case of equation $(\text{A.1})^{public}$, the verifier in our scheme has to perform one signature check for $\{\sigma'_k\}$ per authenticated statement, and the computation of $\prod_{k\in I_{at}} e(V_k, \Lambda_k)$ (and similarly for W_k, Y_k). If we assume to use, for instance Schnorr's signatures for σ'_k, all the signatures can be verified in batch with a work roughly the same as that of computing a single multi-exponentiation like $\prod_{k\in I_{at}}(V_k)^{c_k}$. Also, by considering the micro-benchmarks in [33], the cost of $3N$ pairings is about the cost of 30 multi-exponentiations with N terms.[8] Finally, in our scheme we additionally compute six pairings for equation ($A.2$).

Given such cost evaluation for our scheme against Pinocchio, for a fair comparison, we compare our scheme against the best possible instantiation of the generic construction of Section 3.2, i.e., Pinocchio with the extended relation R'. If we assume that each signature verification costs c multiplication gates in the arithmetic circuits, and if we assume that this is the only additional cost for the design of R', then this means that: if R yields a QAP of size m and degree d, then R' yields a QAP of, at least, size $m' = m + cN$ and degree $d' = d + cN$. When running on R', Pinocchio's performance in *verification* remains the same as the one discussed above, whereas Pinocchio's performance in *proof generation* depends on the larger m' and d'. Precisely, it performs multi-exponentiations with m' and d' terms, and a polynomial division operation whose cost is $O(d' \log^2 d')$. In other words, if we compare the two schemes we obtain:

For secret verification both schemes perform almost the same, the only difference being that we need to perform six more pairings; for public verification our scheme has an additional (concrete) cost of about 30 multi-exponentiations with N terms over Pinocchio. For proof generation Pinocchio (with R') has to perform additional operations that involve a factor at least linear in $c \cdot N$. We recall from the discussion in the Introduction that such c is likely to be as large as 1 000.

[8] Overall, if we take e.g., $N = 100$, the cost of such 30 multi-exponentiations is not that terrible: about 0.5ms, considering costs in [33].

Therefore, one can see that while our solution charges a little more to the verifier (only in the public verification case), the costs of our scheme on the prover side can be much cheaper, at least by a factor cN.

	Our scheme for relation R	Generic scheme for R with Pinocchio
QAP $(size, degree)$	(m, d)	$(m', d') = (m + cN, d + cN)$
Proof generation		
QAP evaluation	$Q(m, d)$	$Q(m', d')$
$V_{mid}, V'_{mid}, W_{mid}, W'_{mid}$ etc.	$7ME(m - a - 1)$	$7ME(m' - a - 1)$
$V_{at}, V'_{at}, \hat{\mu}_v$ etc	$9ME(N)$ terms	—
$h(x)$	$Div(d)$	$Div(d')$
$\tilde{H}$	$1ME(d)$	$1ME(d')$
Verification		
$(A.1)^{\ secret}$	$3ME(N)$	—
$(A.1)^{\ pub}$	$1ME(N) + 3N \cdot P$	—
$(A.2)$	6 P	—
$(P.1)$	$3P + 3ME(a - N)$	$3P + 3ME(a)$
$(P.2)$	$6P$	$6P$
$(P.3)$	$2P$	$2P$

Table 1. Cost of generating and verifying a proof. N = number of authenticated values. c =number of multiplications for one signature verification. P is the cost of a pairing, and $ME(n)$ is the cost of a multi-exponentiation with n terms. $Div(d)$ is the cost of performing a polynomial division for computing $h(x)$ with polynomials of degree d.

4.1 Completeness

Theorem 2. *The above scheme satisfies authentication correctness and completeness.*

Proof. It is straightforward to see that the scheme has authentication correctness by the correctness of the regular signature scheme and by construction. To show the completeness, we prove all verification equations in the order they appear in the verification procedure.

($A.1^{secret}$) We only show the case for $\tilde{\mu}_v$. The cases for $\tilde{\mu}_w$ and $\tilde{\mu}_y$ are analogous.

$$
\begin{aligned}
\tilde{\mu}_v \overset{\mathsf{Prove}}{=} \hat{\mu}_v \cdot (\rho_v)^{\delta_{at}^{(v)}} &\overset{\mathsf{Prove}}{=} \prod_{k \in I_{at}} (V_k)^{\mu_k} \cdot \big(g^{z\, r_v\, t(s)}\big)^{\delta_{at}^{(v)}} \\
&\overset{\mathsf{Auth}}{=} \prod_{k \in I_{at}} (V_k)^{\mathsf{F}_S(\mathsf{L}_k) + z\, c_k} \cdot g^{r_v\, t(s)\, z\, \delta_{at}^{(v)}} \\
&\overset{\mathsf{Gen}}{=} \Big[\prod_{k \in I_{at}} (V_k)^{\mathsf{F}_S(\mathsf{L}_k)} \cdot (V_k)^{z\, c_k} \Big] \cdot (V_t)^{z\, \delta_{at}^{(v)}} \\
&= \Big[\prod_{k \in I_{at}} (V_k)^{\mathsf{F}_S(\mathsf{L}_k)} \Big] \cdot \prod_{k \in I_{at}} (V_k)^{z\, c_k} \cdot (V_t)^{z\, \delta_{at}^{(v)}} \\
&= \Big[\prod_{k \in I_{at}} (V_k)^{\mathsf{F}_S(\mathsf{L}_k)} \Big] \cdot \Big(\prod_{k \in I_{at}} (V_k)^{c_k} \cdot (V_t)^{\delta_{at}^{(v)}} \Big)^{z} \\
&\overset{\mathsf{Prove}}{=} \Big[\prod_{k \in I_{at}} (V_k)^{\mathsf{F}_S(\mathsf{L}_k)} \Big] \cdot \big(V_{at} \cdot (V_t)^{\delta_{at}^{(v)}} \big)^{z} \\
&\overset{\mathsf{Prove}}{=} \Big[\prod_{k \in I_{at}} (V_k)^{\mathsf{F}_S(\mathsf{L}_k)} \Big] \cdot \big(\tilde{V}_{at} \big)^{z}
\end{aligned}
$$

($A.1^{pub}$) We only show the case for $\tilde{\mu}_v$. The cases for $\tilde{\mu}_w$ and $\tilde{\mu}_y$ are analogous.

$$
\begin{aligned}
e(\tilde{\mu}_v, g) \overset{\mathsf{Prove}}{=} e\big(\hat{\mu}_v \cdot (\rho_v)^{\delta_{at}^{(v)}},\, g\big) &\overset{\mathsf{Prove}}{=} e\Big(\prod_{k \in I_{at}} (V_k)^{\mu_k} \cdot \big(g^{z\, r_v\, t(s)}\big)^{\delta_{at}^{(v)}},\, g \Big) \\
&\overset{\mathsf{Auth}}{=} e\Big(\prod_{k \in I_{at}} (V_k)^{\mathsf{F}_S(\mathsf{L}_k) + z\, c_k} \cdot g^{r_v\, t(s)\, z\, \delta_{at}^{(v)}},\, g \Big) \\
&= e\Big(\prod_{k \in I_{at}} (V_k),\, g^{\mathsf{F}_S(\mathsf{L}_k)} \Big) \cdot e\Big(\prod_{k \in I_{at}} (V_k)^{c_k},\, g^{z} \Big) \cdot e\big(g^{r_v\, t(s)\, \delta_{at}^{(v)}},\, g^{z} \big) \\
&\overset{\mathsf{Auth,Gen}}{=} e\Big(\prod_{k \in I_{at}} V_k,\, \Lambda_k \Big) \cdot e\Big(\prod_{k \in I_{at}} (V_k)^{c_k},\, Z \Big) \cdot e\big((V_t)^{\delta_{at}^{(v)}},\, Z \big) \\
&\overset{\mathsf{Prove}}{=} e\Big(\prod_{k \in I_{at}} V_k,\, \Lambda_k \Big) \cdot e\big(V_{at} \cdot (V_t)^{\delta_{at}^{(v)}},\, Z \big) \\
&\overset{\mathsf{Prove}}{=} e\Big(\prod_{k \in I_{at}} V_k,\, \Lambda_k \Big) \cdot e\big(\tilde{V}_{at},\, Z \big)
\end{aligned}
$$

(A.2) We only show the case for $\tilde{V}_{at}$. The cases for $\tilde{W}_{at}$ and $\tilde{Y}_{at}$ are analogous.

$$\begin{aligned}
e(\tilde{V}'_{at},\, g) &= e\big(V'_{at} \cdot (V'_t)^{\delta^{(v)}_{at}},\, g\big) \\
&= e\big(\prod_{k\in I_{at}} (V'_k)^{c_k} \cdot (V'_t)^{\delta^{(v)}_{at}},\, g\big) \\
&= e\big(\prod_{k\in I_{at}} (V_k)^{\alpha_v c_k} \cdot (V_t)^{\alpha_v \delta^{(v)}_{at}},\, g\big) \\
&= e\big(\prod_{k\in I_{at}} (V_k)^{c_k} \cdot (V_t)^{\delta^{(v)}_{at}},\, g^{\alpha_v}\big) \\
&= e\big(V_{at} \cdot (V_t)^{\delta^{(v)}_{at}},\, g^{\alpha_v}\big) \\
&= e(\tilde{V}_{at},\, g^{\alpha_v})
\end{aligned}$$

(P.1)

$$\begin{aligned}
& e\Big(V_0\, \tilde{V}_{at} V_{un} \tilde{V}_{mid} V_{out}, W_0\, \tilde{W}_{at} W_{un} \tilde{W}_{mid} W_{out}\Big) \\
= \;& e\Big(g^{r_v v_0(s)}\, V_{at} (V_t)^{\delta^{(v)}_{at}}\, V_{un} V_{mid} (V_t)^{\delta^{(v)}_{mid}}\, V_m,\; g^{r_w w_0(s)}\, W_{at} (W_t)^{\delta^{(w)}_{at}}\, W_{un} W_{mid} (W_t)^{\delta^{(w)}_{mid}}\, W_m\Big) \\
= \;& e\Big(g^{r_v v_0(s)}\, V_{at} V_{un} V_{mid} V_m\, (V_t)^{\delta^{(v)}_{at}+\delta^{(v)}_{mid}},\; g^{r_w w_0(s)}\, W_{at} W_{un} W_{mid} W_m\, (W_t)^{\delta^{(w)}_{at}+\delta^{(w)}_{mid}}\Big) \\
\overset{c_0\leftarrow 1}{=} \;& e\Big(\Big[\prod_{i\in[0..m]} g^{r_v\, v_i(s)\, c_i}\Big]\, (V_t)^{\delta^{(v)}},\; \Big[\prod_{j\in[0..m]} g^{r_w\, w_j(s)\, c_j}\Big]\, (W_t)^{\delta^{(w)}}\Big) \\
= \;& e\Big(g^{\sum_{i\in[0..m]} r_v\, v_i(s)\, c_i}\; g^{r_v\, t(s)\, \delta^{(v)}},\; g^{\sum_{j\in[0..m]} r_w\, w_j(s)\, c_j}\; g^{r_w\, t(s)\, \delta^{(w)}}\Big) \\
= \;& e\Big(g^{\left[\sum_{i\in[0..m]} r_v\, v_i(s)\, c_i\right] + r_v\, t(s)\, \delta^{(v)}},\; g^{\left[\sum_{j\in[0..m]} r_w\, w_j(s)\, c_j\right] + r_w\, t(s)\, \delta^{(w)}}\Big) \\
= \;& e\Big(g^{r_v\left(\left[\sum_{i\in[0..m]} v_i(s)\, c_i\right] + t(s)\, \delta^{(v)}\right)},\; g^{r_w\left(\left[\sum_{j\in[0..m]} w_j(s)\, c_j\right] + t(s)\, \delta^{(w)}\right)}\Big) \\
= \;& e\Big(g^{\left(\left[\sum_{i\in[0..m]} v_i(s)\, c_i\right] + t(s)\, \delta^{(v)}\right)\cdot\left(\left[\sum_{j\in[0..m]} w_j(s)\, c_j\right] + t(s)\, \delta^{(w)}\right)},\; g\Big)^{r_v\, r_w} \\
\overset{\mathsf{Prove}}{=} \;& e\Big(g^{\left(\tilde{p}(s)+\sum_{k\in[0..m]} y_k(s)\, c_k\right] + t(s)\, \delta^{(y)}\right)},\; g\Big)^{r_y} \\
= \;& e\Big(g^{\tilde{p}(s)} \cdot \Big[\prod_{k\in[0..m]} g^{y_k(s)\, c_k}\Big] \cdot g^{t(s)\, \delta^{(y)}},\; g\Big)^{r_y} \\
= \;& e\Big(g^{r_y\, t(s)\, \tilde{h}(s)} \cdot \Big[\prod_{k\in[0..m]} g^{r_y\, y_k(s)\, c_k}\Big] \cdot g^{r_y\, t(s)\, \delta^{(y)}},\; g\Big) \\
= \;& e\big(g^{r_y\, t(s)\, \tilde{h}(s)}, g\big) \cdot e\Big(\Big[\prod_{k\in[0..m]} g^{r_y\, y_k(s)\, c_k}\Big] \cdot (Y_t)^{\delta^{(y)}},\; g\Big) \\
\overset{c_0=1}{=} \;& e\big(g^{r_y\, t(s)}, g^{\tilde{h}(s)}\big) \cdot e\big(Y_0\; Y_{at} Y_{un} Y_{mid} Y_m \cdot (Y_t)^{\delta^{(y)}_{at}}\, (Y_t)^{\delta^{(y)}_{mid}},\; g\big) \\
= \;& e\big(T, \tilde{H}\big) \cdot e\big(Y_0\, \tilde{Y}_{at} Y_{un} \tilde{Y}_{mid} Y_{out},\; g\big)
\end{aligned}$$

(P.2) We refer to the proof of (A.2), which is very similar to the cases of $\tilde{V}_{mid}$, $\tilde{W}_{mid}$, and $\tilde{Y}_{mid}$.

$(P.3)$

$$
\begin{aligned}
e(\tilde{B}_{mid},\, g^{\gamma}) \overset{\mathsf{Prove}}{=}\; & e(B_{mid}\,(B_v)^{\delta^{(v)}_{mid}}\,(B_w)^{\delta^{(w)}_{mid}}\,(B_y)^{\delta^{(y)}_{mid}},\, g^{\gamma}) \\
=\; & e\Big(\Big[\prod_{k\in I_{mid}} (B_k)^{c_k}\Big]\cdot (V_t)^{\beta\delta^{(v)}_{mid}}\,(W_t)^{\beta\delta^{(w)}_{mid}}\,(Y_t)^{\beta\delta^{(y)}_{mid}},\, g^{\gamma}\Big) \\
=\; & e\Big(\Big[\prod_{k\in I_{mid}} \big((V_k\,W_k\,Y_k)^{\beta}\big)^{c_k}\Big]\cdot (V_t)^{\beta\delta^{(v)}_{mid}}\,(W_t)^{\beta\delta^{(w)}_{mid}}\,(Y_t)^{\beta\delta^{(y)}_{mid}},\, g^{\gamma}\Big) \\
=\; & e\Big(\Big[\prod_{k\in I_{mid}} (V_k\,W_k\,Y_k)^{c_k}\Big]\cdot (V_t)^{\delta^{(v)}_{mid}}\,(W_t)^{\delta^{(w)}_{mid}}\,(Y_t)^{\delta^{(y)}_{mid}},\, g^{\beta\gamma}\Big) \\
=\; & e\Big(\prod_{k\in I_{mid}} (V_k)^{c_k}\prod_{k\in I_{mid}} (W_k)^{c_k}\prod_{k\in I_{mid}} (Y_k)^{c_k}\cdot (V_t)^{\delta^{(v)}_{mid}}\,(W_t)^{\delta^{(w)}_{mid}}\,(Y_t)^{\delta^{(y)}_{mid}},\, g^{\beta\gamma}\Big) \\
=\; & e\big(V_{mid}\,(V_t)^{\delta^{(v)}_{mid}}\,W_{mid}\,(W_t)^{\delta^{(w)}_{mid}}\,Y_{mid}\,(Y_t)^{\delta^{(y)}_{mid}},\, g^{\beta\gamma}\big) \\
=\; & e\big(\tilde{V}_{mid}\,\tilde{W}_{mid}\,\tilde{Y}_{mid},\, g^{\beta\gamma}\big)
\end{aligned}
$$

□

4.2 Proof of Security

In the following theorem we prove the adaptive soundness of our AD-SNARG construction. Note that we can base (part of) its security directly on the soundness of Pinocchio, which is also based on the q-PKE and the q-DHE assumptions.

Theorem 3. *If Pinocchio is a sound SNARG,* F *is a pseudorandom function, the q-PKE [24] and the q-DHE [11] assumptions hold, then the scheme described above is a secretly-verifiable* AD$\mathit{/}$SNARG *with adaptive soundness.*

Before giving the proof, we first recall the q-DHE and the q-PKE assumptions.

Definition 3 (q-Diffie-Hellman Exponent assumption [11]). *The q-DHE problem in a group* $\mathbb{G}$ *of prime order p is defined as follows. Let* $\mathcal{G}$ *be a bilinear group generator, and let* $\mathsf{bgpp} = (p, \mathbb{G}, \mathbb{G}_T, e, g) \leftarrow_{\mathcal{R}} \mathcal{G}(1^{\lambda})$*. Let* $a \leftarrow_{\mathcal{R}} \mathbb{Z}_p$ *be chosen uniformly at random. We define the advantage of an adversary* $\mathcal{A}$ *in solving the q-DHE problem as*

$$\mathbf{Adv}^{q-\mathsf{DHE}}_{\mathcal{A}}(\lambda) = \Pr[\mathcal{A}(\mathsf{bgpp}, g^{a}, \ldots, g^{a^q}, g^{a^{q+2}}, \ldots, g^{a^{2q}}) = g^{a^{q+1}}].$$

We say that the q-DHE assumption holds for $\mathcal{G}$ *if for every PPT algorithm* $\mathcal{A}$ *and any polynomially-bounded* $q = \mathsf{poly}(\lambda)$ *we have that* $\mathbf{Adv}^{q-\mathsf{DHE}}_{\mathcal{A}}(\lambda)$ *is negligible in* λ*.*

Definition 4 (q-Power Knowledge of Exponent assumption [24]). *Let* $\mathcal{G}$ *be a bilinear group generator,* λ *be a security parameter and* $q = \mathsf{poly}(\lambda)$*. The q-PKE assumption holds for* $\mathcal{G}$ *if for every non-uniform PPT adversary* $\mathcal{A}$ *there exists a non-uniform PPT extractor* $\mathcal{E}_{\mathcal{A}}$ *such that:*

$$
\begin{aligned}
&\Pr[h^{\alpha} = \hat{h} \;\wedge\; h \neq g^{\sum_{i=0}^{q} \tilde{v}_i a^i} : \\
&\qquad (h, \hat{h};\, \tilde{v}_0, \ldots, \tilde{v}_q) \leftarrow (\mathcal{A}|\mathcal{E}_{\mathcal{A}})(\mathsf{bgpp}, g^{a}, \ldots, g^{a^q}, g^{\alpha}, g^{\alpha a}, \ldots, g^{\alpha a^q}, aux)] = \mathsf{negl}(\lambda)
\end{aligned}
$$

where $\mathsf{bgpp} = (p, \mathbb{G}, \mathbb{G}_T, e, g) \leftarrow_{\mathcal{R}} \mathcal{G}(1^{\lambda})$*,* $a, \alpha \leftarrow_{\mathcal{R}} \mathbb{Z}_p$ *are chosen uniformly at random, and aux is any auxiliary information that is generated independently of* α*. The notation* $(h, \hat{h};\, \tilde{v}_i) \leftarrow (\mathcal{A}|\mathcal{E}_{\mathcal{A}})(inp)$ *means that* $\mathcal{A}$ *upon input of inp returns* $(h, \hat{h})$ *and* $\mathcal{E}_{\mathcal{A}}$ *on the same input returns* $\tilde{v}_i$*. In this case,* $\mathcal{E}_{\mathcal{A}}$ *has access to* $\mathcal{A}$*'s random tape.*

In order to prove Theorem 3, we describe a series of hybrid experiments $\mathsf{G}_0 - \mathsf{G}_4$, where experiment G_0 is identical to the real adaptive soundness experiment and the remaining experiments $\mathsf{G}_1 - \mathsf{G}_4$ are progressively modified in such a way that each consecutive pair is proven to be (computationally) indistinguishable. Some of the games use some flag values bad_i that are initially set to false. If at the end of a game any of these values is set to true, the `Finalize` procedure always overwrites the outcome of the game to 0. For notation, we denote with G_i the event that a run of G_i with the adversary outputs 1, and we call Bad_i the event that bad_i is set to true during a run of G_i. Essentially, whenever an event Bad_i occurs, the corresponding game may deviate its outcome.

Game G_0: This is the adaptive soundness experiment described in Section 3.1 and Figure 3.

Game G_1: This is the same as G_0 except that the PRF $\mathsf{F}_S(\cdot)$ is replaced by a truly random function $\mathcal{R} : \{0,1\}^* \to \mathbb{F}$. By the security of the PRF, G_1 is computationally indistinguishable from G_0, i.e.,

$$|\Pr[\mathsf{G}_0] - \Pr[\mathsf{G}_1]| \leq \mathbf{Adv}^{PRF}_{\mathcal{D},\mathsf{F}}(\lambda)$$

Game G_2: This is the same as G_1 except that the procedure `Ver` sets $\mathsf{bad}_2 \leftarrow \mathsf{true}$ if the adversary makes verification queries that (a) verify correctly with respect to the equations $(\text{A.1})^{secret}$, and in which (b) there is a label $\mathsf{L} \notin T$ (i.e., $\mathcal{A}$ never asked to authenticate a value under label L). Clearly, G_1 and G_2 are identical until Bad_2, i.e.,

$$|\Pr[\mathsf{G}_1] - \Pr[\mathsf{G}_2]| \leq \Pr[\mathsf{Bad}_2]$$

We show that G_2 is statistically close to G_1, by proving in Lemma 1 that $\Pr[\mathsf{Bad}_2]$ is (unconditionally) negligible. Intuitively, this follows from the fact that when $\mathsf{L} \notin \mathsf{T}$ the first verification check is an equation with an almost-freshly sampled element $\lambda_{\mathsf{L}} = \mathcal{R}(\mathsf{L}) \in \mathbb{F}$, i.e., the equation will be satisfied only with negligible probability, which is at most $1/(p-Q)$.

Game G_3: This is the same as G_2 except for the following change when answering Type 2 verification queries, i.e., we assume every label L was previously used to authenticate a value. Let $\tilde{\mu}_v, \tilde{V}_{at}$, $\tilde{\mu}_w, \tilde{W}_{at}$, and $\tilde{\mu}_y, \tilde{Y}_{at}$ be the elements in the proof $\tilde{\pi}$ queried by the adversary. In G_3 we compute $V^*_{at} = \prod_{k \in I_{at}} (V_k)^{c_k}$ (and W^*_{at}, Y^*_{at} in the similar way), as well as their corresponding authentication tag $\mu^*_v = \prod_{k \in I_{at}} (V_k)^{\mu_k}$ (and μ^*_w, μ^*_y), where each μ_k is the tag previously generated for (L_k, c_k) upon the respective authentication query. Next, we replace the check of equations $(\text{A.1})^{secret}$ with checking whether

$$\begin{array}{rc} & e(\tilde{\mu}_v/\mu^*_v, g) = e(\tilde{V}_{at}/V^*_{at}, g^z) \\ \wedge & e(\tilde{\mu}_w/\mu^*_w, g) = e(\tilde{W}_{at}/W^*_{at}, g^z) \\ \wedge & e(\tilde{\mu}_y/\mu^*_y, g) = e(\tilde{Y}_{at}/Y^*_{at}, g^z) \end{array} \tag{1}$$

is satisfied. Then, if the equations in (A.2) are satisfied, (hence $\tilde{V}'_{at} = (\tilde{V}_{at})^{\alpha_v}$, $\tilde{W}'_{at} = (\tilde{W}_{at})^{\alpha_w}$, $\tilde{Y}'_{at} = (\tilde{Y}_{at})^{\alpha_y}$), we can run an extractor $\mathcal{E}_{\mathcal{A}}$ to obtain polynomials $\tilde{v}_{at}(x)$, $\tilde{w}_{at}(x)$, and $\tilde{y}_{at}(x)$ of degree at most d. If $\tilde{V}_{at} \neq (g^{r_v})^{\tilde{v}_{at}(s)}$ or $\tilde{W}_{at} \neq (g^{r_w})^{\tilde{w}_{at}(s)}$ or $\tilde{Y}_{at} \neq (g^{r_y})^{\tilde{y}_{at}(s)}$, then we set $\mathsf{bad}_3 \leftarrow \mathsf{true}$.

First, we observe that by correctness, checking equation (1) is equivalent to checking the verification equation $(\text{A.1})^{secret}$.

Second, to see that we can run the extractor $\mathcal{E}_{\mathcal{A}}$, we observe that the input received by the adversary $\mathcal{A}$ can indeed be expressed as a pair (S, aux), where $S = \{g^{s^i}, g^{\alpha s^i}\}_{i \in [0,d]}$ and aux

is some auxiliary information independent of α – exactly as in the definition of the d-PKE assumption.

Hence, G_2 and G_3 are identical up to Bad_3, i.e.,

$$|\Pr[\mathsf{G}_2] - \Pr[\mathsf{G}_3]| \leq \Pr[\mathsf{Bad}_3]$$

and it is easy to see that the d-PKE assumption immediately implies that the probability of Bad_3 (i.e., that the extractor outputs a polynomial which is not a correct one) is negligible.

Game G_4: This game proceeds as G_3 except for the following change in the `Ver` procedure. Assume that the equations (1) are satisfied and that $\mathsf{bad}_3 \leftarrow \mathsf{true}$ is not set (i.e., $\tilde{V}_{at} = (g^{r_v})^{\tilde{v}_{at}(s)}$ holds, and similar the corresponding cases of $\tilde{W}_{at}$ and $\tilde{Y}_{at}$).

Then, compute the polynomials $v^*_{at}(x) = \sum_{k \in I_{at}} c_k v_k(x)$ and $\delta_v(x) = \tilde{v}_{at}(x) - v^*_{at}(x)$, where $\tilde{v}_{at}(x)$ is the polynomial obtained from the extractor. Similarly, compute $w^*_{at}(x), \delta_w(x), y^*_{at}(x), \delta_y(x)$.

If any among $\delta_v(x), \delta_w(x), \delta_y(x)$ is *not* divisible by $t(x)$ then set $\mathsf{bad}_4 \leftarrow \mathsf{true}$.

Clearly, G_3 and G_4 are identical up to Bad_4, i.e.,

$$|\Pr[\mathsf{G}_3] - \Pr[\mathsf{G}_4]| \leq \Pr[\mathsf{Bad}_4]$$

To show that the two games are negligibly close, we prove in Lemma 2 that $\Pr[\mathsf{Bad}_4]$ is negligible under the q-DHE assumption, for some $q = 2d + 1$.

Finally, we observe that at this point, if Bad_4 does not occur, we have verified that $\tilde{V}_{at}$, $\tilde{W}_{at}$, and $\tilde{Y}_{at}$ were computed by using the correct (i.e., authenticated) statement values. Namely, except for having randomized elements $\tilde{V}_{at}$ (resp. $\tilde{W}_{at}$, $\tilde{Y}_{at}$), we are almost in the same conditions as in proof of security of Pinocchio. In fact, in Lemma 3 we show that if any adversary has advantage at most ϵ in breaking the security of Pinocchio (in the zero-knowledge SNARG version of the scheme), then $\Pr[\mathsf{G}_4] \leq Q \cdot \epsilon$, where Q is the number of `Gen` queries made by the adversary.

Lemma 1. $\Pr[\mathsf{Bad}_2] \leq \frac{3Q}{p-3Q}$.

Proof. Let Q be the number of verification queries made by the adversary in G_2, and let B_i be the event that bad_2 was set from false to true in the i-th verification query. Clearly, we have:

$$\Pr[\mathsf{Bad}_2] = \Pr\Big[\bigvee_{i=1}^{Q} \mathsf{B}_i\Big] \leq \sum_{i=1}^{Q} \Pr[\mathsf{B}_i]$$

To prove the lemma we will bound the probability $\Pr[\mathsf{B}_i]$ for any $1 \leq i \leq Q$, where the probability is taken over the random choices of the function $\mathcal{R}(\cdot)$.

By definition of B_i we have $\Pr[\mathsf{B}_i] = \Pr[\mathsf{B}_i | \overline{\mathsf{B}}_1 \wedge \cdots \wedge \overline{\mathsf{B}}_{i-1}]$. Also, observe that bad_2 is set to true if $\exists k \in I_{at}$ such that $(\mathsf{L}_k, \cdot) \notin \mathsf{T}$ and at least one of the equations

$$\tilde{\mu}_v = \big[\prod_{k \in I_{at}} (V_k)^{\mathcal{R}(\mathsf{L}_k)}\big] \cdot (\tilde{V}_{at})^z, \quad \tilde{\mu}_w = \big[\prod_{k \in I_{at}} (W_k)^{\mathcal{R}(\mathsf{L}_k)}\big] \cdot (\tilde{W}_{at})^z, \quad \tilde{\mu}_y = \big[\prod_{k \in I_{at}} (Y_k)^{\mathcal{R}(\mathsf{L}_k)}\big] \cdot (\tilde{Y}_{at})^z \qquad (2)$$

is satisfied.

Let us fix one such index $\bar{k} \in I_{at}$ such that $(\mathsf{L}_{\bar{k}}, \cdot) \notin \mathsf{T}$. If $\lambda_{\bar{k}} = \mathcal{R}(\mathsf{L}_{\bar{k}})$ is sampled uniformly at random in the i-th query, then an equation as the ones above will be satisfied with probability

$1/p$, which by union bound sums up to $3/p$. However, the adversary might have asked $\mathsf{L}_{\bar{k}}$ in some previous verification query, and this might have leaked some information about $\lambda_{\bar{k}} = \mathcal{R}(\mathsf{L}_{\bar{k}})$. Yet, since it holds $\overline{\mathsf{B}}_1 \wedge \cdots \wedge \overline{\mathsf{B}}_{i-1}$, the only information leaked to the adversary is that a bunch of equations involving $\lambda_{\bar{k}}$ were not satisfied. For every such equation, one can exclude at most three possible values of $\lambda_{\bar{k}}$. In conclusion, we have that in the i-th query, one of the equations (2) is satisfied with probability at most $\frac{3}{p-3(i-1)}$. Hence,

$$\Pr[\mathsf{Bad}_2] \leq \sum_{i=1}^{Q} \frac{3}{p-3(i-1)} \leq \frac{3Q}{p-3Q}.$$

□

Lemma 2. *If the q-DHE assumption [11] holds for $\mathcal{G}$, then for any PPT adversary $\mathcal{A}$ we have that* $\Pr[\mathsf{Bad}_4]$ *is negligible.*

Proof. Assume that there is an adversary $\mathcal{A}$ such that $\Pr[\mathsf{Bad}_4] \geq \epsilon$ is non-negligible. We show how to build an adversary $\mathcal{B}$ that breaks the q-DHE assumption with probability $\epsilon/DQ - 1/|\mathbb{F}|$ such that: (a) $D = \mathsf{poly}(\lambda)$ is an upper bound on the number of multiplication gates (and thus the degree of the corresponding QAP) in the Q relations $R_1, \ldots, R_Q$ queried by $\mathcal{A}$ to $\mathtt{Gen}$, and (b) $q = 2d^* + 1$ for some $d^* \leq D$, which is the degree of the QAP in the relation R^* for which Bad_4 occurs.

$\mathcal{B}$ takes as input an instance of the q-DHE assumption $(\mathsf{bgpp}, g^a, g^{a^2}, \ldots, g^{a^q}, g^{a^{q+2}}, \ldots, g^{a^{2q}})$ and its goal is to compute the missing element $g^{a^{q+1}}$. To do so, $\mathcal{B}$ simulates G_4 to $\mathcal{A}$ as described in the following. Assume that Bad_4 occurs for the relation R^* which is the j-th relation queried to $\mathtt{Gen}$.

$\underline{\mathtt{Initialize}()}$

- $\mathcal{B}$ runs $\mathtt{Initialize}$ as in G_4 with the following modifications.
- It picks random $j^* \leftarrow_{\mathcal{R}} \{1, \ldots, Q\}$, $d^* \leftarrow_{\mathcal{R}} \{1, \ldots, D\}$ to guess the query's index of R^* and its QAP's degree respectively.
- It picks a random $\nu \leftarrow_{\mathcal{R}} \{0, 1\}$ as a guess on whether Bad_4 will occur for either $\delta_v(x)$ or $\delta_y(x)$ ($\nu = 0$), or $\delta_w(x)$ or $\delta_y(x)$ ($\nu = 1$).
- $\mathcal{B}$ sets $q \leftarrow 2d^* + 1$, and takes as input an instance $(\mathsf{bgpp}, g^a, g^{a^2}, \ldots, g^{a^q}, g^{a^{q+2}}, \ldots, g^{a^{2q}})$ of the q-DHE assumption.
- It defines the degree-d^* polynomial $t^*(x) = \prod_{k=1}^{d^*}(x - r_k)$ where $\{r_k\}$ is a set of canonical roots used to build the QAP.[9]
- $\mathcal{B}$ chooses $z^*(x)$ as a random polynomial in $\mathbb{F}[x]$ of degree $d^* + 1$ such that the polynomial $z^*(x)\, t^*(x)$ of degree $2d^* + 1$ has a zero coefficient in front of x^{d^*+1}.
- $\mathcal{B}$ simulates the secret z with $z^*(a)$ by computing $Z = g^{z^*(a)}$. Observe that $g^{z^*(a)}$ can be computed efficiently using $\{g^{a^i}\}_{i=1}^{d^*+1}$ from the q-DHE instance.
- $\mathcal{B}$ generates a key pair $(\mathsf{sk}', \mathsf{vk}') \leftarrow_{\mathcal{R}} \Sigma.\mathsf{KG}(1^\lambda)$ for the regular signature scheme and outputs $\mathsf{pap} = (\mathsf{pp}, \mathsf{prfpp}, Z)$ and $\mathsf{vk} = (\mathsf{vk}', Z)$.

$\underline{\mathtt{Gen}(R)}$

$\mathcal{B}$ proceeds as follows to simulate the i-th query.

- [Case $i \neq j^*$] $\mathcal{B}$ runs the real $\mathsf{Gen}(\mathsf{pap}, R)$ algorithm and returns its output.

[9] The roots of Pinocchio's QAP target polynomial can be chosen arbitrarily.

– [Case $i = j^*$] Let us call R^* the queried relation. $\mathcal{B}$ simulates the answer to this query as follows. First, it builds the QAP for R^* and if its degree d is not d^*, then $\mathcal{B}$ aborts the simulation. Otherwise, we have $d = d^*$ and hence $t(x) = t^*(x)$ and $\mathcal{B}$ proceeds as follows.
For the value s, instead of randomly choosing it, $\mathcal{B}$ implicitly uses the value a from the q-DHE assumption as follows.
If $\nu = 0$, $\mathcal{B}$ implicitly sets $r_v = r'_v\, a^{d+1}$ and $r_y = r'_v\, r_w\, a^{d+1}$, where $r_w, r'_v \leftarrow_{\mathcal{R}} \mathbb{F}$, by computing

$$V_k = g^{r'_v\, a^{d+1}\, v_k(a)} \qquad Y_k = g^{r'_v\, r_w\, a^{d+1}\, v_k(a)} \qquad V_t = g^{r'_v\, a^{d+1}\, t(a)} \qquad Y_t = g^{r'_v\, r_w\, a^{d+1}\, t(a)}.$$

Notice that these values can be computed efficiently since all the polynomials $a^{d+1}\, v_k(a)$ and $a^{d+1}\, t(a)$ have degree at most $2d^* + 1 = q$. Similarly, all the remaining values $\{W_k, Y_k\}_{k\in[m]}$ can be simulated as the degree of the polynomials encoded in the exponent is at most $d^* < q$.
If $\nu = 1$, $\mathcal{B}$ proceeds in the dual way by setting $r_w = r'_w\, a^{d+1}$ and $r_y = r_v\, r'_w\, a^{d+1}$ for randomly chosen $r_v, r'_w \leftarrow_{\mathcal{R}} \mathbb{F}$. From now on, we describe the simulation for the case $\nu = 0$ only. The other case can easily be adapted.
Finally, $\rho_v = (V_t)^z$ is simulated by computing $(g^{a^{d+1}\, z^*(a)\, t(a)})^{r'_v}$. Notice that $g^{a^{d+1}\, z^*(a) t(a)}$ can be computed since $a^{d+1}\, z^*(a)\, t(a)$ has degree $3d + 2$ and has a zero coefficient in front of $a^{2d+2} = a^{q+1}$. The same holds for the computation of ρ_y, whereas computing $\rho_w = g^{r_w\, z^*(a)\, t(a)}$ can be simulated as $z^*(a)\, t(a)$ has degree $2d + 1 = q$.

$\underline{\texttt{Auth}(\mathsf{L}, c)}$
To simulate authentication queries, $\mathcal{B}$ samples a random $\mu \leftarrow_{\mathcal{R}} \mathbb{F}$, computes $\Lambda = g^{\mu} Z^{-c}$, generates $\sigma' \leftarrow_{\mathcal{R}} \Sigma.\mathsf{Sign}(\mathsf{sk}', \Lambda|\mathsf{L})$, updates $\mathsf{T} \leftarrow \mathsf{T} \cup \{(\mathsf{L}, c)\}$, and returns $\sigma = (\mu, \Lambda, \sigma')$. Observe that such σ is identically distributed as an authentication tag returned by $\texttt{Auth}$ in G_4. Also, although $\mathcal{B}$ is not explicitly generating $\lambda \leftarrow \mathcal{R}(\mathsf{L})$, as one can notice, these values are no longer used to answer the verification queries.

$\underline{\texttt{Ver}(R, \mathsf{L}, \{x_i\}_{\mathsf{L}_i \neq \star}, \tilde{\pi})}$
Finally, we describe how $\mathcal{B}$ handles verification queries. First, note that for those queries that fall in the Type 1 branch, $\mathcal{B}$ can directly answer $\perp$ (reject), and it does not have to use the values $\mathcal{R}(\mathsf{L})$. Clearly, due to definition of game G_4 and since Bad_2 does not occur, answers to these queries are correctly distributed. Second, for queries in the Type 2 branch, we distinguish two cases according to whether the queried relation R is R^* or not.
– If $R \neq R^*$, then $\mathcal{B}$ can answer as in game G_4. In particular, note that equation $(\text{A.1})^{secret}$ has been replaced by equation (1) that requires only public values to be checked.
– If $R = R^*$, then $\mathcal{B}$ proceeds as in G_4. Set $\delta_v(x) \leftarrow \tilde{v}_{at}(x) - v^*_{at}(x)$, $\delta_w(x) \leftarrow \tilde{w}_{at}(x) - w^*_{at}(x)$, and $\delta_y(x) \leftarrow \tilde{y}_{at}(x) - y^*_{at}(x)$.
  • If both $\delta_v(x)$ and $\delta_y(x)$ are divisible by $t^*(x)$, i.e., $\delta_v(x) \in \mathit{Span}(t^*(x))$ and $\delta_y(x) \in \mathit{Span}(t^*(x))$, i.e., Bad_4 did not occur for them, but instead for $\delta_w(x)$ and $\delta_y(x)$, then $\mathcal{B}$ aborts (since here, we detail the case of $\nu = 0$ only).
  • Otherwise, assume that either $\delta_v(x)$ or $\delta_y(x)$ is *not* in $\mathit{Span}(t^*(x))$, and without loss of generality assume this holds for $\delta_v(x)$ (the other case is analogous). Then $\mathcal{B}$ checks whether $\omega(x) = \delta_v(x) z^*(x)$ is such that its coefficient ω_{d+1} is zero. If so, $\mathcal{B}$ aborts the simulation (however, by Lemma 10 [19], this happens with probability at most $1/|\mathbb{F}|$). Otherwise, if $\omega_{d+1} \neq 0$, $\mathcal{B}$ returns

$$\Omega = \left[\frac{\tilde{\mu}_v}{\mu^*_v \prod_{k=0, k\neq d+1}^{2d+1} (g^{a^{k+d+1}})^{r'_v \omega_k}} \right]^{1/(\omega_{d+1}\, r'_v)}$$

Notice that $\mathcal{B}$'s simulation to $\mathcal{A}$ is perfect except if $\mathcal{B}$ aborts. However, $\mathcal{B}$ can abort only in four cases: if its guess on j^* is wrong, i.e., if $j \neq j^*$ (which happens with probability $1-1/Q$); if its guess on d^* is wrong, i.e., if $d \neq d^*$ (which happens with probability $1-1/D$); if its guess on ν is wrong (which happens with probability $1/2$); if $\omega_{d+1} = 0$ (which holds unconditionally with probability at most $1/|\mathbb{F}|$). Also, it is not hard to see that if Bad_4 occurs, then $\mathcal{B}$ returns $\Omega = g^{a^{2d+2}} = g^{a^{q+1}}$ and breaks the q-DHE assumption, as desired.

Therefore, by putting together the probability that $\mathcal{B}$ does not abort, with our assumption that $\Pr[\mathsf{Bad}_4] \geq \epsilon$, then we obtain that $\mathcal{B}$ breaks the q-DHE assumption with probability $\geq \epsilon/2DQ - 1/|\mathbb{F}|$. □

Lemma 3. *If Pinocchio is a sound SNARG scheme, and the q-PKE assumption holds, then for any PPT adversary $\mathcal{A}$ we have that* $\Pr[\mathsf{G}_4]$ *is negligible.*

Proof. We make our reduction by considering a slightly modified version of the Pinocchio scheme in which the evaluation key also includes the values $V'_k = \{g^{r_v \alpha_v v_k(s)}\}_{k \in I_{st}}$ (as well as the corresponding W'_k, Y'_k, and B_k). It is trivial to check that the same proof of security in [33] carries through when these additional values are included in the evaluation key.

Assume by contradiction that there exists an adversary $\mathcal{A}$ such that $\Pr[\mathsf{G}_4] \geq \epsilon$ is non-negligible. We show how to build an adversary $\mathcal{B}$ that breaks the security of Pinocchio with probability at least ϵ/Q_1Q_2, where Q_1 is the number of relations $R_1, \ldots, R_{Q_1}$ queried by $\mathcal{A}$ to $\mathtt{Gen}$ during game G_4, and Q_2 is the number of verification queries. Without loss of generality, assume that $\mathcal{B}$ receives the parameters bgpp of the bilinear groups before choosing the relation R^* to attack.[10]

$\underline{\mathtt{Initialize}()}$

- $\mathcal{B}$ picks a random $j^* \leftarrow_{\mathcal{R}} \{1, \ldots, Q_1\}$ to guess the query's index of R^*, the relation for which $\mathcal{A}$ will break the security of our $\mathsf{AD/SNARG}$ scheme.
- $\mathcal{B}$ generates a key pair $(\mathsf{sk}', \mathsf{vk}') \leftarrow_{\mathcal{R}} \Sigma.\mathsf{KG}(1^\lambda)$ for the regular signature scheme, and then samples a random $z \leftarrow_{\mathcal{R}} \mathbb{F}$. It outputs $\mathsf{pap} = (\mathsf{bgpp}, \mathsf{prfpp}, Z = g^z)$ and $\mathsf{vk} = (\mathsf{vk}', Z)$.

$\underline{\mathtt{Gen}(R)}$

$\mathcal{B}$ proceeds as follows to simulate the i-th generation query.

- [Case $i \neq j^*$] $\mathcal{B}$ runs the real $\mathsf{Gen}(\mathsf{pap}, R)$ algorithm and returns its output.
- [Case $i = j^*$] Let us call R^* the queried relation. $\mathcal{B}$ forwards R^* to its challenger and receives a pair of keys $(\mathsf{VK}^*_P, \mathsf{EK}^*_P)$ of the Pinocchio scheme. $\mathcal{B}$ then uses z to compute $\rho_v = (V_t)^z$, $\rho_w = (W_t)^z$, and $\rho_y = (Y_t)^z$, sets the key pair of the $\mathsf{AD/SNARG}$ scheme to $(\mathsf{VK}^*, \mathsf{EK}^*)$, where $\mathsf{VK}^* = \mathsf{VK}^*_P$ and EK^* consists of EK^*_P and the additional values ρ_v, ρ_w, ρ_y, and the elements $\{V_k, W_k, Y_k\}_{k \in \mathcal{I}_{st}}$ of VK^*_P.

$\underline{\mathtt{Auth}(L, c)}$

$\mathcal{B}$ runs $\mathtt{Auth}$ as in G_4, i.e., $\mathcal{B}$ outputs $\sigma = (\mu = \mathcal{R}(\mathsf{L}) + zc, \Lambda = g^{\mathcal{R}(\mathsf{L})}, \sigma' = \Sigma.\mathsf{Sign}(\mathsf{sk}', \Lambda|\mathsf{L}))$.

$\underline{\mathtt{Ver}(R, \mathsf{L}, \{x_i\}_{\mathsf{L}_i \neq \star}, \tilde{\pi})}$

Finally, we describe how $\mathcal{B}$ simulates verification queries to $\mathcal{A}$. Notice that all the equation checks require only public values. Also, observe that in G_4 the adversary $\mathcal{A}$ can win only by returning a Type 2 forgery, and by returning a proof $\tilde{\pi}$ containing values $\tilde{V}_{at}, \tilde{V}'_{at}$ of the "correct form", i.e.,

[10] We note that this reduction to the security of Pinocchio is done for ease of exposition. Indeed, we could have included in our simulator $\mathcal{B}$ the same code of the simulator in the security proof of the Pinocchio scheme, where the parameters of the bilinear groups are received at the very beginning.

$\tilde{V}_{at} = (g^{r_v})^{v^*_{at}(s)+\delta^{(v)}_{at}t(s)}$ and $\tilde{V}'_{at} = (g^{r_v\alpha_v})^{v^*_{at}(s)+\delta^{(v)}_{at}t(s)}$ respectively, for some $\delta^{(v)}_{at} \in \mathbb{F}$. Similarly it holds the correctness of $\tilde{W}_{at}, \tilde{W}'_{at}$, $\tilde{Y}_{at}$ and $\tilde{Y}'_{at}$ for some coefficients $\delta^{(w)}_{at}, \delta^{(y)}_{at} \in \mathbb{F}$.

So, for every verification query that passes the verification checks and that involves the relation R^*, $\mathcal{B}$ translates the given proof $\tilde{\pi}$ into a proof π_P as described below.

Translation of $\tilde{\pi}$ to π_P. Let $\tilde{\pi} = (\tilde{\mu}_v, \tilde{\mu}_w, , \tilde{\mu}_y, \tilde{V}_{at}, \tilde{V}'_{at}, \tilde{W}_{at}, \tilde{W}'_{at}, \tilde{Y}_{at}, \tilde{Y}'_{at}, \tilde{V}_{mid}, \tilde{V}'_{mid}, \tilde{W}_{mid}, \tilde{W}'_{mid}, \tilde{Y}_{mid}, \tilde{Y}'_{mid}, \tilde{B}_{mid}, \tilde{H})$. First, $\mathcal{B}$ computes $V_{mid} = \tilde{V}_{mid} \cdot \tilde{V}_{at}/V^*_{at}$ and $V'_{mid} = \tilde{V}'_{mid} \cdot \tilde{V}'_{at}/V'^*_{at}$, where $V^*_{at} = \prod_{k\in I_{at}}(V_k)^{c_k}$ and $V'^*_{at} = \prod_{k\in I_{at}}(V'_k)^{c_k}$. Similarly, $\mathcal{B}$ computes W_{mid}, W'_{mid}, Y_{mid} and Y'_{mid}. Then, $\mathcal{B}$ computes $B_{mid} = \tilde{B}_{mid} \cdot (B_v)^{\delta^{(at)}_v} \cdot (B_w)^{\delta^{(at)}_w} \cdot (B_y)^{\delta^{(at)}_y}$, where $\delta^{(at)}_v = (\tilde{v}_{at}(x) - v^*_{at}(x))/t(x)$. The values $\delta^{(at)}_w$ and $\delta^{(at)}_y$ are computed accordingly. Next, $\mathcal{B}$ changes the (accepting) proof $\tilde{\pi}$ produced by $\mathcal{A}$ by replacing $\tilde{V}_{mid}$ (resp. $\tilde{V}'_{mid}$, $\tilde{W}_{mid}$, $\tilde{W}'_{mid}$, $\tilde{Y}_{mid}$, $\tilde{Y}'_{mid}$, $\tilde{B}_{mid}$) with the value V_{mid} (resp. V'_{mid}, W_{mid}, W'_{mid}, Y_{mid}, Y'_{mid}, B_{mid}) computed above, and by removing $\tilde{V}_{at}$, $\tilde{V}'_{at}$, $\tilde{W}_{at}$, $\tilde{W}'_{at}$, $\tilde{Y}_{at}$, $\tilde{Y}'_{at}$, $\tilde{\mu}_v$, $\tilde{\mu}_w$ and $\tilde{\mu}_y$. Let π_P be such modified proof. $\mathcal{B}$ stores the tuple $(\{c_k\}_{k\in\mathcal{I}_{st}}, \pi_P)$ into a list Ω.

First, observe that the proof π_P is identical to a proof in the Pinocchio scheme, and in particular it has the same distribution. Second, we claim that if $\tilde{\pi}$ is accepted in G_4 for relation R^* and labels $\{\mathsf{L}_k\}_{k\in\mathcal{I}_{at}}$ (used to authenticate $\{c_k\}_{k\in\mathcal{I}_{at}}$), then π_P is accepted for statement $\{c_k\}_{k\in\mathcal{I}_{st}}$ in the given instance of the Pinocchio scheme for relation R^*.

The first claim follows by inspection and by observing that since Bad_4 does not occur, the value $(\tilde{V}_{at}/V^*_{at})$ contains a multiple of $t(s)$ in the exponent, i.e., the honest form of V_{mid} is preserved. In particular, the value $\delta^{(at)}_v$ is a scalar value since $(\tilde{v}_{at}(x) - v^*_{at}(x))$ is divisible by $t(x)$ and $\deg(\tilde{v}_{at}(x)) = \deg(v^*_{at}(x))$.

The second claim follows from the fact that the value $\tilde{V} = \tilde{V}_{at} \cdot V_{un} \cdot \tilde{V}_{mid} \cdot V_{out}$ computed to verify the proof in the $\mathsf{AD/SNARG}$ scheme, and the value $V = (\prod_{k\in\mathcal{I}_{st}}(V_k)^{c_k}) \cdot V_{mid} \cdot V_{out}$ computed to verify the proof in Pinocchio are identical (because of $V_{mid} = \tilde{V}_{mid} \cdot \tilde{V}_{at}/V^*_{at}$). Also, note that similar arguments apply for the corresponding W and Y values. Since Bad_4 does not occur, the value $\delta^{(at)}_v$ is exactly the value used by $\mathcal{A}$ for the randomization of $\tilde{V}_{at}$.

After $\mathcal{A}$ stops running, $\mathcal{B}$ picks a random tuple $(\{c_k\}_{k\in\mathcal{I}_{st}}, \pi_P)$ from the list Ω (which contains at most Q_2 elements) and returns this tuple to its challenger.

To complete the proof we analyze $\mathcal{B}$'s success probability. We claim that if $\mathcal{A}$ breaks the security of the $\mathsf{AD/SNARG}$ scheme in game G_4, then $\mathcal{B}$ breaks the security of Pinocchio with probability at least $1/Q_1Q_2$. It is not hard to see that $\mathcal{B}$'s simulation has a distribution which is statistically close to the distribution of game G_4. Also, if $\mathcal{A}$ breaks the scheme it means that for at least one of its verification queries that accepts, say the ℓ-th query, we have that $x \notin R$. Assume that R was the j-th relation queried to $\mathtt{Gen}$, and that $\mathcal{B}$ returns the ℓ^*-th tuple in the list Ω. Since the simulation does not leak any information on j^* and ℓ^*, we have that $\Pr[j^* = j \wedge \ell^* = \ell] \geq 1/Q_1Q_2$. Therefore, if $\mathcal{A}$ breaks the security of the $\mathsf{AD/SNARG}$ scheme in game G_4 with probability at least ϵ, then $\mathcal{B}$ breaks the security of Pinocchio with probability $\geq \epsilon/Q_1Q_2$. □

Security with public verifiability It is easy to adapt the proof of Theorem 3 in order to show that our scheme is sound also in the case where the proof is made publicly verifiable. Hence, it is possible to prove the following theorem:

Theorem 4. *If Pinocchio is a sound SNARG,* F *is a pseudorandom function,* Σ *is a secure signature scheme, the d-PKE [24] and the q-DHE [11] assumptions hold, then the scheme described above is a publicly-verifiable* AD/SNARG *with adaptive soundness.*

In the publicly verifiable case, since the adversary can verify the proofs on its own, we can assume that it makes a single verification query to `Ver`. To obtain the proof of Theorem 4, we use the same games as those for Theorem 3. The only difference is that the probability $\Pr[\mathsf{Bad}_2]$ is now shown to be negligible under the assumption that the regular signature scheme is secure. Such is rather straightforward: an adversary which returns a proof involving a statement value with label L_k that had not been queried to the `Auth` oracle, has to show at least one signature σ'_k for some non-queried label L.

4.3 Proof of the Zero-Knowledge Property

Theorem 5. *The* AD/SNARG *scheme described in Section 4 is statistically zero-knowledge in the sense of Definition 2.*

Proof. To see that our scheme satisfies zero-knowledge, our first observation is that the group elements $\tilde{V}_{at}$, $\tilde{V}_{mid}$, $\tilde{W}_{at}$, $\tilde{W}_{mid}$, $\tilde{Y}_{at}$, and $\tilde{Y}_{mid}$ are statistically uniform over $\mathbb{G}$. Indeed, as long as $t(s) \neq 0$, each of these elements is uniformly randomized.

Second, we notice that once the elements $\tilde{V}_{at}$, $\tilde{V}_{mid}$, $\tilde{W}_{at}$, $\tilde{W}_{mid}$, $\tilde{Y}_{at}$, and $\tilde{Y}_{mid}$ are fixed, the values of all the remaining elements in $\tilde{\pi}$, i.e., $\tilde{\mu}_v$, $\tilde{V}'_{at}$, $\tilde{V}'_{mid}$, $\tilde{\mu}_w$, $\tilde{W}'_{at}$, $\tilde{W}'_{mid}$, $\tilde{\mu}_y$, $\tilde{Y}'_{at}$, $\tilde{Y}'_{mid}$, $\tilde{B}_{mid}$, and $\tilde{H}$ get determined according to the constraints of the verification equations $(A.1)$, $(A.2)$, $(P.1)$, $(P.2)$, $(P.3)$.

Finally, we show that there is a simulator $(\mathsf{Sim}_1, \mathsf{Sim}_2)$, formally described in Figure 4, that satisfies Definition 2. It is trivial to see that the simulated keys generated by Sim_1 are distributed as in the real experiment. Regarding Sim_2, it is not hard to see that the simulated values $\tilde{V}_{at}, \tilde{V}_{mid}, \tilde{W}_{at}, \tilde{W}_{mid}, \tilde{Y}_{at}$ and $\tilde{Y}_{mid}$ are statistically uniform. Also, given the trapdoor, Sim_2 can generate (without knowing the inputs $\{c_k\}_{k \in I_{at}}$) all the remaining elements of $\tilde{\pi}$ with the correct distribution, i.e., such that verification equations $(A.1)$, $(A.2)$, $(P.1)$, $(P.2)$, $(P.3)$ are satisfied. □

5 Our Construction of Secretly-Verifiable Zero-Knowledge AD-SNARGs

In this section, we show a variant of the scheme proposed in Section 4 which allows for a verification algorithm whose efficiency does *not* depend on the number of authenticated values. In order to achieve this appealing property, we trade efficiency for usability in making the previous scheme only secretly verifiable.

$\mathsf{Setup}(1^\lambda)$**:** On input the security parameter 1^λ, run $\mathsf{pp} = (p, \mathbb{G}, \mathbb{G}_T, e, g) \leftarrow_{\mathcal{R}} \mathcal{G}(1^\lambda)$ to generate a bilinear group description, where $\mathbb{G}$ and $\mathbb{G}_T$ are groups of the same prime order $p > 2^\lambda$, $g \in \mathbb{G}$ is a generator and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is an efficiently computable bilinear map.

$\mathsf{AuthKeyGen}(\mathsf{pp})$**:** Run $(S, \mathsf{prfpp}) \leftarrow_{\mathcal{R}} \mathsf{F.KG}(1^\lambda)$ to obtain the seed S and the public parameters prfpp of a pseudorandom function $\mathsf{F}_S : \{0,1\}^* \to \mathbb{G}$. Choose a random value $z \leftarrow_{\mathcal{R}} \mathbb{F}$. Compute $Z = e(g,g)^z \in \mathbb{G}_T$. Return the secret key $\mathsf{sk} = \mathsf{vk} = (S, z)$, and the public authentication parameters $\mathsf{pap} = (\mathsf{pp}, \mathsf{prfpp}, Z)$.

$\underline{\mathsf{Sim}_1(\mathsf{pp}, R, \mathsf{sk}, \mathsf{vk}, \mathsf{pap})}$
Run $\mathsf{Gen}(\mathsf{pap}, R)$ to obtain
$(\mathsf{EK}_R, \mathsf{VK}_R)$ and also store
$\mathsf{sk}, s, \beta, \alpha_v, \alpha_w, \alpha_y, r_v, r_w, r_y$ in td
$\mathtt{Return}\ (\mathsf{EK}_R, \mathsf{VK}_R, \mathsf{td})$

$\underline{\mathsf{Sim}_2(\mathsf{td}, \mathsf{L}, \{x_i\}_{\mathsf{L}_i=\star})}$
$\mathtt{let}\ c_m = 1, v_{out}(x) = c_m v_m(x), v_{un}(x) = \sum_{k\in I_{un}} c_k v_k(x)$
$w_{out}(x) = c_m w_m(x), w_{un}(x) = \sum_{k\in I_{un}} c_k w_k(x)$
$y_{out}(x) = c_m y_m(x), y_{un}(x) = \sum_{k\in I_{un}} c_k y_k(x)$
$\{\lambda_k \leftarrow \mathsf{F}_S(\mathsf{L}_k)\}_{k\in I_{at}}$
$\tilde{v}_{at}(x), \tilde{v}_{mid}(x) \leftarrow_{\mathcal{R}} \mathbb{F}[x]$
$\tilde{v}(x) \leftarrow v_0(x) + \tilde{v}_{at}(x) + v_{un}(x) + \tilde{v}_{mid}(x) + v_{out}(x)$
Choose random $\tilde{w}_{mid}(x), \tilde{w}_{at}(x), \tilde{y}_{mid}(x), \tilde{y}_{at}(x)$, such that $t(x)|\tilde{p}(x)$
where $\tilde{p}(x) \leftarrow \tilde{v}(x)\,\tilde{w}(x) - \tilde{y}(x)$ and
$\tilde{w}(x) \leftarrow w_0(x) + \tilde{w}_{at}(x) + w_{un}(x) + \tilde{w}_{mid}(x) + w_{out}(x)$
$\tilde{y}(x) \leftarrow y_0(x) + \tilde{y}_{at}(x) + y_{un}(x) + \tilde{y}_{mid}(x) + y_{out}(x)$
$\tilde{h}(x) \leftarrow \tilde{p}(x)\,/\,t(x)$
$\tilde{\mu}_v \leftarrow \prod_{k\in I_{at}} (V_k)^{\lambda_k} \cdot Z^{r_v \tilde{v}_{at}(s)}$
$\tilde{\mu}_w \leftarrow \prod_{k\in I_{at}} (W_k)^{\lambda_k} \cdot Z^{r_w \tilde{w}_{at}(s)}, \tilde{\mu}_y \leftarrow \prod_{k\in I_{at}} (Y_k)^{\lambda_k} \cdot Z^{r_y \tilde{y}_{at}(s)}$
$\tilde{V}_{at} \leftarrow g^{r_v \tilde{v}_{at}(s)}, \tilde{V}'_{at} \leftarrow (\tilde{V}_{at})^{\alpha_v}, \tilde{V}_{mid} \leftarrow g^{r_v \tilde{v}_{mid}(s)}, \tilde{V}'_{mid} \leftarrow (\tilde{V}_{mid})^{\alpha_v}$
$\tilde{W}_{at} \leftarrow g^{r_w \tilde{w}_{at}(s)}, \tilde{W}'_{at} \leftarrow (\tilde{W}_{at})^{\alpha_w}, \tilde{W}_{mid} \leftarrow g^{r_w \tilde{w}_{mid}(s)}, \tilde{W}'_{mid} \leftarrow (\tilde{W}_{mid})^{\alpha_w}$
$\tilde{Y}_{at} \leftarrow g^{r_y \tilde{y}_{at}(s)}, \tilde{Y}'_{at} \leftarrow (\tilde{Y}_{at})^{\alpha_y}, \tilde{Y}_{mid} \leftarrow g^{r_y \tilde{y}_{mid}(s)}, \tilde{Y}'_{mid} \leftarrow (\tilde{Y}_{mid})^{\alpha_y}$
$\tilde{B}_{mid} \leftarrow (\tilde{V}_{mid} \cdot \tilde{W}_{mid} \cdot \tilde{Y}_{mid})^{\beta}$
$\tilde{H} \leftarrow g^{\tilde{h}(s)}$
$\mathtt{Return}\ \tilde{\pi} = (\tilde{\mu}_v, \tilde{\mu}_w, \tilde{\mu}_y, \tilde{V}_{at}, \tilde{V}'_{at}, \tilde{V}_{mid}, \tilde{V}'_{mid}, \tilde{W}_{at}, \tilde{W}'_{at}, \tilde{W}_{mid}, \tilde{W}'_{mid}, \tilde{Y}_{at}, \tilde{Y}'_{at}, \tilde{Y}_{mid}, \tilde{Y}'_{mid}, \tilde{B}_{mid}, \tilde{H})$

Fig. 4. Simulator Sim.

$\mathsf{Auth}(\mathsf{sk}, \mathsf{L}, c)$**:** Let $\mathsf{sk} = (S, z)$. To authenticate a value $c \in \mathbb{F}$ with label L, use the PRF to compute $\hat{R} \leftarrow \mathsf{F}_S(\mathsf{L})$, then compute $\sigma = \hat{R} \cdot (g^z)^c$ and output σ.

$\mathsf{AuthVer}(\mathsf{vk}, \sigma, \mathsf{L}, c)$**:** Let $\mathsf{vk} = (S, z)$ be the (secret) verification key. To verify that σ is a valid authentication tag for a value $c \in \mathbb{F}$ with respect to label L, output $\top$ if $\sigma = \mathsf{F}_S(\mathsf{L}) \cdot (g^z)^c$ and $\bot$ otherwise.

$\mathsf{Gen}(\mathsf{pap}, R)$**:** Let $R : \mathbb{F}^a \times \mathbb{F}^b$ be an $\mathcal{NP}$ relation with statements of length a and witnesses of length b. Let C_R be R's characteristic circuit, i.e., $C_R(x, w) = 1$ iff $(x, w) \in R$. Build a QAP $Q_R = (t(x), \mathcal{V}, \mathcal{W}, \mathcal{Y})$ of size m and degree d for C_R. We denote by I_{st}, I_{mid}, I_{out} the following partitions of $\{1, \ldots, m\}$: $I_{st} = \{1, \ldots, a\}$, $I_{mid} = \{a+1, \ldots, m-1\}$ and $I_{out} = \{m\}$.[11] In other words, we partition all the circuit wires into: statement wires I_{st}, internal wires I_{mid} (including the witness wires), and the output wire I_{out}.

Next, pick $r_v, r_w \leftarrow_{\mathcal{R}} \mathbb{F}$ uniformly at random and set $r_y = r_v\, r_w$. Then pick $s, \alpha_v, \alpha_w, \alpha_y, \beta, \gamma \leftarrow_{\mathcal{R}} \mathbb{F}$ uniformly at random and compute the following values:

$$
\begin{aligned}
& & T &= g^{r_y t(s)} \\
\forall k \in [m] \cup \{0\}: & & V_k &= g^{r_v v_k(s)}, \quad W_k = g^{r_w w_k(s)}, \quad Y_k = g^{r_y y_k(s)}, \\
\forall k \in [m]: & & V'_k &= (V_k)^{\alpha_v}, \quad W'_k = (W_k)^{\alpha_w}, \quad Y'_k = (Y_k)^{\alpha_y}, \quad B_k = (V_k\, W_k\, Y_k)^{\beta}.
\end{aligned}
$$

[11] For a reader familiar with Pinocchio, we point out our change of notation: we will use v_{st} instead of v_{in} to refer to the statement-related inputs.

Additionally, compute the following values:

$$\begin{array}{lll} \rho_v = Z^{r_v\, t(s)}, & \rho_w = Z^{r_w\, t(s)}, & \rho_y = Z^{r_y\, t(s)}, \\ V_t = g^{r_v\, t(s)}, & W_t = g^{r_w\, t(s)}, & Y_t = g^{r_y\, t(s)}, \\ V'_t = (V_t)^{\alpha_v}, & W'_t = (W_t)^{\alpha_w}, & Y'_t = (Y_t)^{\alpha_y}, \\ B_v = (V_t)^{\beta}, & B_w = (W_t)^{\beta}, & B_y = (Y_t)^{\beta}. \end{array}$$

Output the *evaluation key* EK_R and the *verification key* VK_R defined as follows:

$$\begin{aligned} \mathsf{EK}_R = \Big(& \{V_k, V'_k, W_k, W'_k, Y_k, Y'_k, B_k\}_{k\in I_{st}\cup I_{mid}},\ \{g^{s^i}\}_{i\in[d]} \\ & V_t, V'_t,\ W_t, W'_t,\ Y_t, Y'_t,\ B_v, B_w, B_y,\ \rho_v, \rho_w, \rho_y,\ Q_R \Big) \\ \mathsf{VK}_R = \Big(& g,\ g^{\alpha_v}, g^{\alpha_w}, g^{\alpha_y}, g^{\gamma},\ g^{\beta\gamma},\ T,\ \{V_k, W_k, Y_k\}_{k\in I_{st}\cup\{0,m\}} \Big) \end{aligned}$$

$\mathsf{Prove}(\mathsf{EK}_R, x, w, \sigma)$**:** Let EK_R the be evaluation key as defined above, $(x, w) \in \mathbb{F}^a \times \mathbb{F}^b$ be a statement-witness pair, and $\sigma = (\sigma_1, \ldots, \sigma_a)$ be a tuple of authentication tags for $x = (x_1, \ldots, x_a)$ such that for any $i \in [a]$ either $\sigma_i = \hat{R}_i \cdot (g^z)^{x_i}$ or $\sigma_i = \star$. We define $I_{at} = \{i \in I_{st} : \sigma_i \neq \star\} \subseteq I_{st}$ as the set of indices for which there is an authenticated statement value, and let $I_{un} = I_{st} \setminus I_{at}$ be its complement.

To produce a proof for $(x, w) \in R$ proceed as follows. First, evaluate the circuit $C_R(x, w)$ and learn the values of all internal wires: $\{c_k\}_{k\in I_{mid}}$. For ease of description, we assume $c_i = x_i$ for $i \in [a]$, and $c_{a+i} = w_i$ for $i \in [b]$. The first b indices of I_{mid} hence represent the witness values w.

Next, proceed as follows to compute the proof:

$$\begin{array}{llll} V_{at} = \prod_{k\in I_{at}} (V_k)^{c_k}, & W_{at} = \prod_{k\in I_{at}} (W_k)^{c_k}, & Y_{at} = \prod_{k\in I_{at}} (Y_k)^{c_k}, \\ V'_{at} = \prod_{k\in I_{at}} (V'_k)^{c_k}, & W'_{at} = \prod_{k\in I_{at}} (W'_k)^{c_k}, & Y'_{at} = \prod_{k\in I_{at}} (Y'_k)^{c_k}, \\ V_{mid} = \prod_{k\in I_{mid}} (V_k)^{c_k}, & W_{mid} = \prod_{k\in I_{mid}} (W_k)^{c_k}, & Y_{mid} = \prod_{k\in I_{mid}} (Y_k)^{c_k}, \\ V'_{mid} = \prod_{k\in I_{mid}} (V'_k)^{c_k}, & W'_{mid} = \prod_{k\in I_{mid}} (W'_k)^{c_k}, & Y'_{mid} = \prod_{k\in I_{mid}} (Y'_k)^{c_k}, \\ B_{mid} = \prod_{k\in I_{mid}} (B_k)^{c_k}. & & \end{array}$$

Authenticate the values V_{at}, W_{at}, and Y_{at} by computing $\hat{\sigma}_v = \prod_{k\in I_{at}} e(V_k, \sigma_k)$, $\hat{\sigma}_w = \prod_{k\in I_{at}} e(W_k, \sigma_k)$, and $\hat{\sigma}_y = \prod_{k\in I_{at}} e(Y_k, \sigma_k)$, respectively.

To make the proof zero-knowledge, pick random values $\delta_{at}^{(v)}, \delta_{mid}^{(v)}, \delta_{at}^{(w)}, \delta_{mid}^{(w)}, \delta_{at}^{(y)}, \delta_{mid}^{(y)} \leftarrow_{\mathcal{R}} \mathbb{F}$, and compute:

$$\begin{aligned}
\tilde{V}_{at} &= V_{at} \cdot (V_t)^{\delta_{at}^{(v)}}, \quad & \tilde{W}_{at} &= W_{at} \cdot (W_t)^{\delta_{at}^{(w)}}, \quad & \tilde{Y}_{at} &= Y_{at} \cdot (Y_t)^{\delta_{at}^{(y)}}, \\
\tilde{V}'_{at} &= V'_{at} \cdot (V'_t)^{\delta_{at}^{(v)}}, \quad & \tilde{W}'_{at} &= W'_{at} \cdot (W'_t)^{\delta_{at}^{(w)}}, \quad & \tilde{Y}'_{at} &= Y'_{at} \cdot (Y'_t)^{\delta_{at}^{(y)}}, \\
\tilde{V}_{mid} &= V_{mid} \cdot (V_t)^{\delta_{mid}^{(v)}}, \quad & \tilde{W}_{mid} &= W_{mid} \cdot (W_t)^{\delta_{mid}^{(w)}}, \quad & \tilde{Y}_{mid} &= Y_{mid} \cdot (Y_t)^{\delta_{mid}^{(y)}}, \\
\tilde{V}'_{mid} &= V'_{mid} \cdot (V'_t)^{\delta_{mid}^{(v)}}, \quad & \tilde{W}'_{mid} &= W'_{mid} \cdot (W'_t)^{\delta_{mid}^{(w)}}, \quad & \tilde{Y}'_{mid} &= Y'_{mid} \cdot (Y'_t)^{\delta_{mid}^{(y)}}, \\
\tilde{B}_{mid} &= B_{mid} \cdot (B_v)^{\delta_{mid}^{(v)}} \cdot (B_w)^{\delta_{mid}^{(w)}} \cdot (B_y)^{\delta_{mid}^{(y)}}
\end{aligned}$$

To authenticate the new values $\tilde{V}_{at}$, $\tilde{W}_{at}$, and $\tilde{Y}_{at}$, compute $\tilde{\sigma}_v = \hat{\sigma}_v \cdot (\rho_v)^{\delta_{at}^{(v)}}$, $\tilde{\sigma}_w = \hat{\sigma}_w \cdot (\rho_w)^{\delta_{at}^{(w)}}$, and $\tilde{\sigma}_y = \hat{\sigma}_y \cdot (\rho_y)^{\delta_{at}^{(y)}}$, respectively. Note that our technique preserves the re-randomization property of Pinocchio.

Next, solve the QAP Q_R by finding a polynomial $\tilde{h}(x)$ such that $\tilde{p}(x) = \tilde{h}(x) \cdot t(x)$ where the polynomial $\tilde{p}(x)$ includes the "perturbed versions" of the polynomials $v(x)$, $w(x)$, and $y(x)$ with $\delta^{(v)} = \delta_{at}^{(v)} + \delta_{mid}^{(v)}$, $\delta^{(w)} = \delta_{at}^{(w)} + \delta_{mid}^{(w)}$, and $\delta^{(y)} = \delta_{at}^{(y)} + \delta_{mid}^{(y)}$, respectively:

$$\begin{aligned}
\tilde{p}(x) = \Big(v_0(x) + \sum_{k\in[m]} c_k v_k(x) \; + \; \delta^{(v)} t(x)\Big)\Big(w_0(x) + \sum_{k\in[m]} c_k w_k(x) \; + \; \delta^{(w)} t(x)\Big) \\
- \Big(y_0(x) + \sum_{k\in[m]} c_k y_k(x) \; + \; \delta^{(y)} t(x)\Big)
\end{aligned}$$

Finally, compute $\tilde{H} = g^{\tilde{h}(s)}$ using the values g^{s^i} contained in the evaluation key EK_R. Output $\tilde{\pi} = (\tilde{\sigma}_v, \tilde{\sigma}_w, \tilde{\sigma}_y, \tilde{V}_{at}, \tilde{V}'_{at}, \tilde{V}_{mid}, \tilde{V}'_{mid}, \tilde{W}_{at}, \tilde{W}'_{at}, \tilde{W}_{mid}, \tilde{W}'_{mid}, \tilde{Y}_{at}, \tilde{Y}'_{at}, \tilde{Y}_{mid}, \tilde{Y}'_{mid}, \tilde{B}_{mid}, \tilde{H})$.

$\mathsf{Ver}(\mathsf{vk}, \mathsf{VK}_R, \mathsf{L}, \{x_i\}_{\mathsf{L}_i=\star}, \tilde{\pi})$**:** Let $\mathsf{vk} = (S, z)$ be the authentication verification key, VK_R be the verification key for relation R, $\mathsf{L} = (\mathsf{L}_1, \ldots, \mathsf{L}_a)$ be a vector of labels, and let $\tilde{\pi}$ be a proof as defined above. In a similar way as in Prove, we define $I_{at} = \{i \in I_{st} : \mathsf{L}_i \neq \star\} \subseteq I_{st}$ and $I_{un} = I_{st} \setminus I_{at}$. The verification algorithm proceeds as follows:

(*A*.1) Check the authenticity of $\tilde{V}_{at}$, $\tilde{W}_{at}$, and $\tilde{Y}_{at}$ against the labels L:

$$\begin{aligned}
\tilde{\sigma}_v &= \Big[\prod_{k\in I_{at}} e(V_k, \mathsf{F}_S(\mathsf{L}_k))\Big] \cdot e(\tilde{V}_{at}, g^z) \\
\wedge \; \tilde{\sigma}_w &= \Big[\prod_{k\in I_{at}} e(W_k, \mathsf{F}_S(\mathsf{L}_k))\Big] \cdot e(\tilde{W}_{at}, g^z) \\
\wedge \; \tilde{\sigma}_y &= \Big[\prod_{k\in I_{at}} e(Y_k, \mathsf{F}_S(\mathsf{L}_k))\Big] \cdot e(\tilde{Y}_{at}, g^z)
\end{aligned}$$

(*A*.2) Check that $\tilde{V}_{at}$, $\tilde{V}'_{at}$, $\tilde{W}_{at}$, $\tilde{W}'_{at}$, and $\tilde{Y}_{at}$, $\tilde{Y}'_{at}$ were computed using the same linear combination:

$$e(\tilde{V}'_{at}, g) = e(\tilde{V}_{at}, g^{\alpha_v}) \;\wedge\; e(\tilde{W}'_{at}, g) = e(\tilde{W}_{at}, g^{\alpha_w}) \;\wedge\; e(\tilde{Y}'_{at}, g) = e(\tilde{Y}_{at}, g^{\alpha_y})$$

($P.1$) Check the satisfiability of the QAP by setting $V_{out} = (V_m)^{c_m} = V_m$ (similarly $W_{out} = W_m$ and $Y_{out} = Y_m$), where we assume that $c_m = 1 = C_R(x, w)$ since $(x, w) \in R$, then computing $V_{un} = \prod_{k \in I_{un}} (V_k)^{x_k}$ (and similarly W_{un}, Y_{un}), and finally checking:

$$e(V_0 \, \tilde{V}_{at} \; V_{un} \; \tilde{V}_{mid} \, V_{out}, \; W_0 \, \tilde{W}_{at} \; W_{un} \; \tilde{W}_{mid} \, W_{out}) = e(T, \, \tilde{H}) \cdot e(Y_0 \, \tilde{Y}_{at} \; Y_{un} \; \tilde{Y}_{mid} \, Y_{out}, \; g)$$

($P.2$) Check that all linear combinations are in the appropriate spans:

$$e(\tilde{V}'_{mid}, g) = e(\tilde{V}_{mid}, g^{\alpha_v}) \;\; \wedge \;\; e(\tilde{W}'_{mid}, g) = e(\tilde{W}_{mid}, g^{\alpha_w}) \;\; \wedge \;\; e(\tilde{Y}'_{mid}, g) = e(\tilde{Y}_{mid}, g^{\alpha_y})$$

($P.3$) Check that all the QAP linear combinations use the same coefficients:

$$e(\tilde{B}_{mid}, g^{\gamma}) = e(\tilde{V}_{mid} \; \tilde{W}_{mid} \; \tilde{Y}_{mid}, g^{\beta\gamma})$$

If all the checks above are satisfied, then return $\top$; otherwise return $\bot$.

Efficient Verification. By assuming a proper labeling of the data and a suitable pseudorandom function F, the scheme described above is adapted to allow for an improved verification algorithm whose running time does not depend on the number $|I_{at}|$ of authenticated values. Following the ideas in [2], we assume that every input c is labeled by using a multi-label $\mathsf{L} = (\Delta, \tau)$, where Δ is a data set identifier, and τ is an input identifier. As an example, the input identifiers $\tau_1, \ldots \tau_n$ can be *specific* canonical information like date and time (e.g., day 05, 11:12:42), and the data set identifier Δ can be more *general* information describing the category (e.g., energy consumption for March 2014).

As for the pseudorandom function, we can instantiate F_S by using the specific ACF-efficient PRF of [2] $\mathsf{F}_S : \{0,1\}^* \times \{0,1\}^* \to \mathbb{G}$ such that: $\mathsf{F}_S(\Delta, \tau) = g^{a_\Delta \lambda_\tau + b_\Delta \mu_\tau}$, where the values (a_Δ, b_Δ) and (λ_τ, μ_τ) are derived by applying two standard PRFs (mapping into $\mathbb{F}$) to Δ and τ, respectively. This function is pseudorandom under the Decision Linear assumption [2]. To achieve efficient verification one proceeds as follows (we describe only the case for V, i.e., ω_v and Ω_v, the computations for W and Y are similar):

- Offline phase: precompute $\omega_v^{(\lambda)} = e(\prod_{k \in I_{at}} (V_k)^{\lambda_k}, g)$ and $\omega_v^{(\mu)} = e(\prod_{k \in I_{at}} (V_k)^{\mu_k}, g)$ where (λ_k, μ_k) are derived from τ_k for all $k \in I_{at}$. Store $(\omega_v^{(\lambda)}, \omega_v^{(\mu)})$.
- Online phase: given Δ, derive (a_Δ, b_Δ) from Δ, and compute $\Omega_v = (\omega_v^{(\lambda)})^{a_\Delta} \cdot (\omega_v^{(\mu)})^{b_\Delta}$. Finally, use Ω_v to check the verification equation ($A.1$), i.e., check that $\tilde{\sigma}_v = \Omega_v \cdot e(\tilde{V}_{at}, g^z)$.

The correctness of this efficient verification follows from $\Omega_v = \left[\prod_{k \in I_{at}} e(V_k, \mathsf{F}_S(\Delta, \tau_k))\right]$.

5.1 Correctness

Theorem 6. *The above scheme satisfies authentication correctness and completeness.*

Proof. It is straightforward to see that the scheme has authentication correctness by construction: $\sigma = \mathsf{F}_S(\mathsf{L}) \cdot (g^z)^c$. In order to show the completeness, we prove the correctness of equation ($A.1$). The remaining equations are the same as those of the scheme in Section 4.

($A.1$) We only prove the case for σ_v, the cases for σ_w and σ_y are equal.

$$
\begin{aligned}
\tilde{\sigma}_v \overset{\mathsf{Prove}}{=} \hat{\sigma}_v \cdot (\rho_v)^{\delta_{at}^{(v)}} &\overset{\mathsf{Prove}}{=} \prod_{k \in I_{at}} e(V_k,\, \sigma_k) \cdot (Z^{\,r_v t(s)})^{\delta_{at}^{(v)}} \\
&\overset{\mathsf{Auth}}{=} \prod_{k \in I_{at}} e(V_k,\, \mathsf{F}_S(\mathsf{L}_k)\; g^{z\, c_k}) \cdot (e(g,g)^{z\, r_v\, t(s)})^{\delta_{at}^{(v)}} \\
&= \Big[\prod_{k \in I_{at}} e(V_k,\, \mathsf{F}_S(\mathsf{L}_k)) \cdot e(V_k,\, g^{z\, c_k}) \Big] \cdot e(g,g)^{z\, r_v\, t(s)\, \delta_{at}^{(v)}} \\
&= \Big[\prod_{k \in I_{at}} e(V_k,\, \mathsf{F}_S(\mathsf{L}_k)) \Big] \cdot \Big[\prod_{k \in I_{at}} e(V_k,\, g^{z\, c_k}) \Big] \cdot e(g^{r_v\, t(s)\, \delta_{at}^{(v)}}, g^z) \\
&\overset{\mathsf{Gen}}{=} \Big[\prod_{k \in I_{at}} e(V_k,\, \mathsf{F}_S(\mathsf{L}_k)) \Big] \cdot e\big(\prod_{k \in I_{at}} (V_k)^{c_k},\, g^z \big) \cdot e((V_t)^{\delta_{at}^{(v)}}, g^z) \\
&\overset{\mathsf{Prove}}{=} \Big[\prod_{k \in I_{at}} e(V_k,\, \mathsf{F}_S(\mathsf{L}_k)) \Big] \cdot e\big(V_{at},\, g^z\big) \cdot e((V_t)^{\delta_{at}^{(v)}}, g^z) \\
&= \Big[\prod_{k \in I_{at}} e(V_k,\, \mathsf{F}_S(\mathsf{L}_k)) \Big] \cdot e(V_{at}\, (V_t)^{\delta_{at}^{(v)}},\, g^z) \\
&\overset{\mathsf{Prove}}{=} \Big[\prod_{k \in I_{at}} e(V_k,\, \mathsf{F}_S(\mathsf{L}_k)) \Big] \cdot e(\tilde{V}_{at},\, g^z)
\end{aligned}
$$

□

5.2 Proof of Security

Theorem 7. *If Pinocchio is a sound SNARG scheme,* F *is a pseudorandom function, and the d-PKE [24] and q-BDHE [8] assumptions hold, then the scheme described above is an* AD/SNARG *with adaptive soundness.*

Before giving the proof, we first recall the q-BDHE assumption, which is an easy extension of the q-DHE assumption (Definition 3).

Definition 5 (q-Bilinear Diffie-Hellman assumption ([8])). *Let* $\mathcal{G}$ *be a bilinear group generator, and let* $\mathsf{bgpp} = (p, \mathbb{G}, \mathbb{G}_T, e, g) \leftarrow_{\mathcal{R}} \mathcal{G}(1^\lambda)$. *Let* $\eta, a \leftarrow_{\mathcal{R}} \mathbb{Z}_p$ *be chosen uniformly at random. We define the advantage of an adversary* $\mathcal{A}$ *in solving the q-BDHE problem as*

$$\mathbf{Adv}_{\mathcal{A}}^{q-\mathrm{BDHE}}(\lambda) = \Pr[\mathcal{A}(\mathsf{bgpp}, g^\eta, g^a, \ldots, g^{a^q}, g^{a^{q+2}}, \ldots, g^{a^{2q}}) = e(g,g)^{\eta a^{q+1}}]$$

We say that the q-BDHE assumption holds for $\mathcal{G}$ *if for every PPT algorithm* $\mathcal{A}$ *and any polynomially-bounded* $q = \mathsf{poly}(\lambda)$ *we have that* $\mathbf{Adv}_{\mathcal{A}}^{q-\mathrm{BDHE}}(\lambda)$ *is negligible in* λ.

In order to prove the theorem, we describe a series of hybrid experiments $\mathsf{G}_0 - \mathsf{G}_4$ defined as follows.

Game G_0 This is the adaptive soundness experiment described in Section 3.1 and Figure 3.

Game G_1 This is the same as G_0 except that the PRF $\mathsf{F}_S(\cdot,\cdot)$ is replaced by a truly random function $\mathcal{R} : \{0,1\}^* \to \mathbb{G}$. By the security of the PRF, G_1 is computationally indistinguishable from G_0, i.e.,

$$|\Pr[\mathsf{G}_0] - \Pr[\mathsf{G}_1]| \leq \mathbf{Adv}^{PRF}_{\mathcal{D},\mathsf{F}}(\lambda)$$

Game G_2**:** This is the same as G_1 except that the procedure $\mathtt{Ver}$ sets $\mathsf{bad}_2 \leftarrow \mathsf{true}$ if the adversary makes verification queries that (a) verify correctly with respect to equation $(A.1)$, and in which (b) there is a label $(\mathsf{L},\cdot) \notin \mathsf{T}$ (i.e., $\mathcal{A}$ never asked to authenticate a value under label L). Clearly, G_1 and G_2 are identical until Bad_2, i.e.,

$$|\Pr[\mathsf{G}_1] - \Pr[\mathsf{G}_2]| \leq \Pr[\mathsf{Bad}_2]$$

As in the proof of Theorem 3, it is possible to show that for every PPT adversary the probability $\Pr[\mathsf{Bad}_2]$ is (unconditionally) negligible. In particular, we can use essentially the same argument of Lemma 1 to show that $\Pr[\mathsf{Bad}_2] \leq \frac{Q}{p-Q}$.

Game G_3**:** This is the same as G_2 except for the following change when answering Type 2 verification queries, i.e., we assume every label L was previously used to authenticate a value. Let $\tilde{\sigma}_v, \tilde{V}_{at}$, $\tilde{\sigma}_w, \tilde{W}_{at}$, and $\tilde{\sigma}_y, \tilde{Y}_{at}$ be the elements in the proof $\tilde{\pi}$ queried by the adversary. In G_3 we compute $V^*_{at} = \prod_{k\in I_{at}}(V_k)^{c_k}$ (and W^*_{at}, Y^*_{at} in the similar way), as well as its corresponding authentication tag $\sigma^*_v = \prod_{k\in I_{at}} e(V_k, \sigma_k)$ (and σ^*_w, σ^*_y), where each σ_k is the tag previously generated for (L_k, c_k) upon the respective authentication query. Next, we replace the check of equation $(A.1)$ with checking whether

$$\begin{aligned} & \tilde{\sigma}_v/\sigma^*_v = e(\tilde{V}_{at}/V^*_{at}, g^z) \\ \wedge \quad & \tilde{\sigma}_w/\sigma^*_w = e(\tilde{W}_{at}/W^*_{at}, g^z) \\ \wedge \quad & \tilde{\sigma}_y/\sigma^*_y = e(\tilde{Y}_{at}/Y^*_{at}, g^z) \end{aligned} \tag{3}$$

is satisfied. Then, if the equations in (A.2) are satisfied, (hence $\tilde{V}'_{at} = (\tilde{V}_{at})^{\alpha_v}$, $\tilde{W}'_{at} = (\tilde{W}_{at})^{\alpha_w}$, $\tilde{Y}'_{at} = (\tilde{Y}_{at})^{\alpha_y}$), we can run an extractor $\mathcal{E}_{\mathcal{A}}$ to obtain polynomials $\tilde{v}_{at}(x), \tilde{w}_{at}(x), \tilde{y}_{at}(x)$ of degree at most d. If $\tilde{V}_{at} \neq (g^{r_v})^{\tilde{v}_{at}(s)}$ or $\tilde{W}_{at} \neq (g^{r_w})^{\tilde{w}_{at}(s)}$ or $\tilde{Y}_{at} \neq (g^{r_y})^{\tilde{y}_{at}(s)}$, then we set $\mathsf{bad}_3 \leftarrow \mathsf{true}$.

First, we observe that by correctness, checking equation (3) is equivalent to checking verification equation $(A.1)$. Indeed, if we let $R^*_v = \left[\prod_{k\in I_{at}} e(V_k, \mathcal{R}(\mathsf{L}_k))\right]$, then correctness implies that $\sigma^*_v = R^*_v \cdot e(V^*_{at}, g^z)$, and thus we can rewrite the first part of equation $(A.1)$, i.e., $\tilde{\sigma}_v = R^*_v \cdot e(\tilde{V}_{at}, g^z)$, as

$$\tilde{\sigma}_v = \frac{\sigma^*_v}{e(V^*_{at}, g^z)}\, e(\tilde{V}_{at}, g^z)$$

(and similar for $\tilde{\sigma}_w$ and $\tilde{\sigma}_y$) from which we obtain equation (3).

Second, to see that we can run the extractor $\mathcal{E}_{\mathcal{A}}$, we observe that the input received by the adversary $\mathcal{A}$ can indeed be expressed as a pair (S, aux), where $S = \{g^{s^i}, g^{\alpha s^i}\}_{i\in[0,d]}$ and aux is some auxiliary information independent of α – exactly as in the definition of the d-PKE assumption.

Hence, G_2 and G_3 are identical up to Bad_3, i.e.,

$$|\Pr[\mathsf{G}_2] - \Pr[\mathsf{G}_3]| \leq \Pr[\mathsf{Bad}_3]$$

and it is easy to see that the d-PKE assumption immediately implies that the probability of Bad_3 (i.e., that the extractor outputs a polynomial which is not a correct one) is negligible.

Game G_4: This game proceeds as G_3 except for the following change in the `Ver` procedure. Assume that the equations (3) are satisfied and that $\mathsf{bad}_3 \leftarrow \mathsf{true}$ is not set (i.e., $\tilde{V}_{at} = (g^{r_v})^{\tilde{v}_{at}(s)}$ holds, and similar the corresponding cases of $\tilde{W}_{at}$ and $\tilde{Y}_{at}$).
Then, compute the polynomials $v^*_{at}(x) = \sum_{k \in I_{at}} c_k v_k(x)$ and $\delta_v(x) = \tilde{v}_{at}(x) - v^*_{at}(x)$, where $\tilde{v}_{at}(x)$ is the polynomial obtained from the extractor. Similarly, compute $w^*_{at}(x), \delta_w(x), y^*_{at}(x), \delta_y(x)$.
If any among $\delta_v(x), \delta_w(x), \delta_y(x)$ is *not* divisible by $t(x)$ then set $\mathsf{bad}_4 \leftarrow \mathsf{true}$.
Clearly, G_3 and G_4 are identical up to Bad_4, i.e.,

$$|\Pr[\mathsf{G}_3] - \Pr[\mathsf{G}_4]| \leq \Pr[\mathsf{Bad}_4]$$

To show that the two games are negligibly close, we prove in Lemma 4 that $\Pr[\mathsf{Bad}_4]$ is negligible under the q-BDHE assumption, for some $q = 2d + 1$.

Finally, we observe that at this point, if Bad_4 does not occur, we have verified that $\tilde{V}_{at}$, $\tilde{W}_{at}$, and $\tilde{Y}_{at}$ were computed by using the correct (i.e., authenticated) statement values. Namely, except for having randomized elements $\tilde{V}_{at}$ (resp. $\tilde{W}_{at}$, $\tilde{Y}_{at}$), we are almost in the same conditions as in proof of security of Pinocchio. In fact, in Lemma 5 we show that if any adversary has advantage at most ϵ in breaking the security of Pinocchio (in the zero-knowledge SNARG version of the scheme), then $\Pr[\mathsf{G}_4] \leq Q \cdot \epsilon$, where Q is the number of `Gen` queries made by the adversary.

To conclude the proof, we prove our lemmas bounding, respectively, the probabilities $\Pr[\mathsf{Bad}_4]$ and $\Pr[\mathsf{G}_4]$.

Lemma 4. *If the q-BDHE assumption holds for $\mathcal{G}$, then for any PPT adversary $\mathcal{A}$ we have that* $\Pr[\mathsf{Bad}_4]$ *is negligible.*

Proof. Assume that there is an adversary $\mathcal{A}$ such that $\Pr[\mathsf{Bad}_4] \geq \epsilon$ is non-negligible. We show how to build an adversary $\mathcal{B}$ that breaks the q-BDHE assumption with probability $\epsilon/2DQ^2 - 1/|\mathbb{F}|$ such that: (a) $D = \mathsf{poly}(\lambda)$ is an upper bound on the number of multiplication gates (and thus the degree of the corresponding QAP) in the Q relations $R_1, \ldots, R_Q$ queried by $\mathcal{A}$ to `Gen`, and (b) $q = 2d^* + 1$ for some $d^* \leq D$, which is the degree of the QAP in the relation R^* for which Bad_4 occurs.

$\mathcal{B}$ takes as input an instance of the q-BDHE assumption $(\mathsf{bgpp}, g^\eta, g^a, g^{a^2}, \ldots, g^{a^q}, g^{a^{q+2}}, \ldots, g^{a^{2q}})$ and its goal is to compute the missing element $e(g^\eta, g^{a^{q+1}})$. To do so, it simulates G_4 to $\mathcal{A}$ as described in the following. Assume that Bad_4 occurs for the relation R^* which is the j-th relation queried to `Gen`.

`Initialize`()

- $\mathcal{B}$ runs `Initialize` as in G_4 with the following modifications.
- It picks random $j^* \leftarrow_{\mathcal{R}} \{1, \ldots, Q\}$, $d^* \leftarrow_{\mathcal{R}} \{1, \ldots, D\}$ to guess the query's index of R^* and its QAP's degree respectively.
- It picks a random $\nu \leftarrow_{\mathcal{R}} \{0, 1\}$ as a guess on whether Bad_4 will occur for either $\delta_v(x)$ or $\delta_y(x)$ ($\nu = 0$), or $\delta_w(x)$ or $\delta_y(x)$ ($\nu = 1$).
- $\mathcal{B}$ sets $q \leftarrow 2d^* + 1$, and takes as input an instance $(\mathsf{bgpp}, g^\eta, g^a, g^{a^2}, \ldots, g^{a^q}, g^{a^{q+2}}, \ldots, g^{a^{2q}})$ of the q-BDHE assumption.
- It defines the degree-d^* polynomial $t^*(x) = \prod_{k=1}^{d^*}(x - r_k)$ where $\{r_k\}$ is a set of canonical roots used to build the QAP.[12]

[12] The roots of Pinocchio's QAP target polynomial can be chosen arbitrarily.

– $\mathcal{B}$ chooses $z^*(x)$ as a random polynomial in $\mathbb{F}[x]$ of degree d^*+1 such that the polynomial $z^*(x)\,t^*(x)$ of degree $2d^*+1$ has a zero coefficient in front of x^{d^*+1}.
– $\mathcal{B}$ simulates the secret z with $\eta\, z^*(a)$ by computing $Z = e(g^\eta, g^{z^*(a)})$. Observe that $g^{z^*(a)}$ can be computed efficiently using $\{g^{a^i}\}_{i=1}^q$ from the q-BDHE instance.

$\underline{\mathtt{Gen}(R)}$
$\mathcal{B}$ proceeds as follows to simulate the i-th query.
– [Case $i \neq j^*$] $\mathcal{B}$ runs the real $\mathsf{Gen}(\mathsf{pap}, R)$ algorithm and returns its output.
– [Case $i = j^*$] Let us call R^* the queried relation. $\mathcal{B}$ simulates the answer to this query as follows. First, it builds the QAP for R^* and if its degree d is not d^*, then $\mathcal{B}$ aborts the simulation. Otherwise, we have $d = d^*$ and hence $t(x) = t^*(x)$ and $\mathcal{B}$ proceeds as follows.
For the value s, instead of randomly choosing it, $\mathcal{B}$ implicitly uses the value a from the q-DHE assumption as follows.
If $\nu = 0$, $\mathcal{B}$ implicitly sets $r_v = r'_v\, a^{d+1}$ and $r_y = r'_v\, r_w\, a^{d+1}$, where $r_w, r'_v \leftarrow_{\mathcal{R}} \mathbb{F}$, by computing

$$V_k = g^{r'_v\, a^{d+1}\, v_k(a)} \qquad Y_k = g^{r'_v\, r_w\, a^{d+1}\, v_k(a)} \qquad V_t = g^{r'_v\, a^{d+1}\, t(a)} \qquad Y_t = g^{r'_v\, r_w\, a^{d+1}\, t(a)}.$$

Notice that these values can be computed efficiently since all the polynomials $a^{d+1}\, v_k(a)$ and $a^{d+1}\, t(a)$ have degree at most $2d^*+1 = q$. Similarly, all the remaining values $\{W_k, Y_k\}_{k\in[m]}$ can be simulated as the degree of the polynomials encoded in the exponent is at most $d^* < q$.
If $\nu = 1$, $\mathcal{B}$ proceeds in the dual way by setting $r_w = r'_w\, a^{d+1}$ and $r_y = r_v\, r'_w\, a^{d+1}$ for randomly chosen $r_v, r'_w \leftarrow_{\mathcal{R}} \mathbb{F}$. From now on, we describe the simulation for the case $\nu = 0$ only. The other case can easily be adapted.
Finally, $\rho_v = Z^{r_v\, t(s)}$ is simulated by computing $e(g^\eta, g^{a^{d+1}\, z^*(a)\, t(a)})^{r'_v}$. Notice that $g^{a^{d+1}\, z^*(a) t(a)}$ can be computed since $a^{d+1}\, z^*(a)\, t(a)$ has degree $3d+2$ and has a zero coefficient in front of $a^{2d+2} = a^{q+1}$. The same holds for the computation of ρ_y, whereas computing $\rho_w = e(g^\eta, g^{z^*(a)\, t(a)})^{r_w}$ can be simulated since $z^*(a)\, t(a)$ has degree $2d+1 = q$.

$\underline{\mathtt{Auth}(\mathsf{L}, c)}$
To simulate authentication queries, $\mathcal{B}$ samples a random $R \leftarrow_{\mathcal{R}} \mathbb{G}$, updates $\mathsf{T} \leftarrow \mathsf{T} \cup \{(\mathsf{L}, c)\}$, and returns $\sigma = R$. Observe that such σ is identically distributed as an authentication tag returned by $\mathtt{Auth}$ in G_4. Also, although $\mathcal{B}$ is not explicitly generating $R \leftarrow \mathcal{R}(\mathsf{L})$, as one can notice, these values are no longer used to answer the verification queries.

$\underline{\mathtt{Ver}(R, \mathsf{L}, \{x_i\}_{\mathsf{L}_i \neq \star}, \tilde{\pi})}$
Finally, we describe how $\mathcal{B}$ handles verification queries. First, note that for those queries that fall in the Type 1 branch, $\mathcal{B}$ can directly answer $\perp$ (reject), and it does not have to use the values $\mathcal{R}(\mathsf{L})$. Clearly, due to definition of game G_4 and since Bad_2 does not occur, answers to these queries are correctly distributed. Second, for queries in the Type 2 branch, we distinguish two cases according to whether the queried relation R is R^* or not.
– If $R \neq R^*$, then we only show how $\mathcal{B}$ simulates the check of equations (3), i.e., $\tilde{\sigma}_v/\sigma^*_v = e(\tilde{V}_{at}/V^*_{at}, g^z)$, and similar for W and Y. Note that $\mathcal{B}$ does not know $g^z = g^{\eta z^*(a)}$.
First, let $s, r_v \in \mathbb{F}$ be the values chosen in $\mathtt{Gen}$, which $\mathcal{B}$ knows because of $R \neq R^*$. Then $\mathcal{B}$ proceeds as in G_4 except that it replaces equations (3) with

$$\tilde{\sigma}_v = \sigma^*_v\; e(g^\eta, g^{z^*(a)})^{r_v(\tilde{v}_{at}(s) - v^*_{at}(s))}$$

(and similar for W and Y). The polynomial $\tilde{v}_{at}(x)$ is obtained by the extractor. It is not hard to see that such replacement generates an equivalent check.

– If $R = R^*$, then $\mathcal{B}$ proceeds as in G_4. Set $\delta_v(x) \leftarrow \tilde{v}_{at}(x) - v^*_{at}(x)$, $\delta_w(x) \leftarrow \tilde{w}_{at}(x) - w^*_{at}(x)$, and $\delta_y(x) \leftarrow \tilde{y}_{at}(x) - y^*_{at}(x)$.
  - If $\delta_v(x)$ and $\delta_y(x)$ are divisible by $t^*(x)$, i.e., $\delta_v(x) \in \mathit{Span}(t^*(x))$ and $\delta_y(x) \in \mathit{Span}(t^*(x))$, then $\mathcal{B}$ replaces equation (3) with

$$\tilde{\sigma}_v = \sigma^*_v\, e(g^\eta, g^{a^{d+1}\delta_v(a)z^*(a)})^{r'_v}, \quad \tilde{\sigma}_w = \sigma^*_w\, e(g^\eta, g^{\delta_w(a)z^*(a)})^{r_w}, \quad \tilde{\sigma}_y = \sigma^*_y\, e(g^\eta, g^{a^{d+1}\delta_y(a)z^*(a)})^{r'_y}$$

    Recall that we assume $\nu = 0$ and observe that $g^{a^{d+1}\delta_v(a)z^*(a)}$ can indeed be computed as it has a zero coefficient in front of $a^{2d+2} = a^{q+1}$.
  - Otherwise, we assume that $\delta_v(x) \notin \mathit{Span}(t^*(x))$. The case for $\delta_y(x)$ is analogous.
    $\mathcal{B}$ checks whether $\omega(x) = \delta_v(x)z^*(x)$ is such that its coefficient ω_{d+1} is zero. If so, $\mathcal{B}$ aborts the simulation (however, by Lemma 10 [19], this happens with probability at most $1/|\mathbb{F}|$). Otherwise, if $\omega_{d+1} \neq 0$, $\mathcal{B}$ computes

$$\Omega = \left[\frac{\tilde{\sigma}}{\sigma^* \prod_{k=0,k\neq d+1}^{2d+1} e(g^\eta, g^{a^{d+k+1}})^{r'_v\omega_k}}\right]^{1/(\omega_{d+1}\, r'_v)}$$

    and inserts Ω in a list *List* and outputs $\perp$ (reject).

At the end of the simulation, $\mathcal{B}$ picks a random value Ω in *List* and returns Ω as its solution for the q-BDHE assumption. Notice that $\mathcal{B}$'s simulation is perfect except if $\mathcal{B}$ aborts. However, $\mathcal{B}$ can abort only in three cases: (a) if its guess on j^* is wrong, i.e., if $j \neq j^*$ (which happens with probability $1 - 1/Q$); (b) if its guess on d^* is wrong, i.e., if $d \neq d^*$ (which happens with probability $1 - 1/D$); and (c) if $\omega_{d+1} = 0$ (which holds unconditionally with probability at most $1/|\mathbb{F}|$). Also, it is not hard to see that if Bad_4 occurs and if the guess of ν is correct (which happens with probability $1/2$), then $\mathcal{B}$ must insert $\Omega^* = e(g^\eta, g^{a^{q+1}})$ in *List*. Since *List* contains at most Q values, $\mathcal{B}$ will pick the correct Ω^* with probability at least $1/Q$.

Therefore, by putting together the probability that $\mathcal{B}$ does not abort, and that the correct Ω^* is picked, with our assumption that $\Pr[\mathsf{Bad}_4] \geq \epsilon$, then we obtain that $\mathcal{B}$ breaks the q-BDHE assumption with probability $\geq \epsilon/2DQ^2 - 1/|\mathbb{F}|$. □

Lemma 5. *If Pinocchio is a secure verifiable computation scheme, then for any PPT adversary $\mathcal{A}$ we have that* $\Pr[\mathsf{G}_4]$ *is negligible.*

The proof is essentially the same as that of Lemma 3.

5.3 Proof of the Zero-Knowledge Property

Theorem 8. *The* AD/SNARG *scheme described in Section 5 is statistically zero-knowledge.*

Proof. The proof of this theorem is essentially the same as that for the scheme of Section 4. The only difference is the pseudorandom function.

References

1. B. Applebaum, Y. Ishai, and E. Kushilevitz. From secrecy to soundness: Efficient verification via secure computation. In S. Abramsky, C. Gavoille, C. Kirchner, F. Meyer auf der Heide, and P. G. Spirakis, editors, *ICALP 2010, Part I*, volume 6198 of *LNCS*, pages 152–163, Bordeaux, France, July 6–10, 2010. Springer, Berlin, Germany.
2. M. Backes, D. Fiore, and R. M. Reischuk. Verifiable delegation of computation on outsourced data. In *2013 ACM Conference on Computer and Communication Security*. ACM Press, November 2013.
3. BBC. Google unveils 'smart contact lens' to measure glucose levels. `http://www.bbc.com/news/technology-25771907`, 2014.
4. E. Ben-Sasson, A. Chiesa, D. Genkin, E. Tromer, and M. Virza. SNARKs for C: Verifying program executions succinctly and in zero knowledge. In *CRYPTO 2013*, LNCS, pages 90–108, Santa Barbara, CA, USA, Aug. 2013. Springer, Berlin, Germany.
5. N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *ITCS '12: Proceedings of the 3rd Symposium on Innovations in Theoretical Computer Science*, 2012.
6. N. Bitansky, A. Chiesa, Y. Ishai, R. Ostrovsky, and O. Paneth. Succinct non-interactive arguments via linear interactive proofs. In *TCC 2013*, LNCS, pages 315–333. Springer, Berlin, Germany, 2013.
7. D. Boneh and X. Boyen. Short signatures without random oracles. In C. Cachin and J. Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 56–73, Interlaken, Switzerland, May 2–6, 2004. Springer, Berlin, Germany.
8. D. Boneh, X. Boyen, and E.-J. Goh. Hierarchical identity based encryption with constant size ciphertext. In R. Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 440–456, Aarhus, Denmark, May 22–26, 2005. Springer, Berlin, Germany.
9. D. Boneh and D. M. Freeman. Homomorphic signatures for polynomial functions. In K. G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 149–168, Tallinn, Estonia, May 15–19, 2011. Springer, Berlin, Germany.
10. X. Boyen. Lattice mixing and vanishing trapdoors: A framework for fully secure short signatures and more. In P. Q. Nguyen and D. Pointcheval, editors, *PKC 2010*, volume 6056 of *LNCS*, pages 499–517, Paris, France, May 26–28, 2010. Springer, Berlin, Germany.
11. J. Camenisch, M. Kohlweiss, and C. Soriente. An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In S. Jarecki and G. Tsudik, editors, *PKC 2009*, volume 5443 of *LNCS*, pages 481–500, Irvine, CA, USA, Mar. 18–20, 2009. Springer, Berlin, Germany.
12. D. Catalano and D. Fiore. Practical homomorphic MACs for arithmetic circuits. In *Eurocrypt '13: Proceedings of the 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 2013.
13. D. Chaum. Security without identification: Transaction systems to make big brother obsolete. *Commun. ACM*, 28(10):1030–1044, Oct. 1985.
14. K.-M. Chung, Y. Kalai, and S. P. Vadhan. Improved delegation of computation using fully homomorphic encryption. In T. Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 483–501, Santa Barbara, CA, USA, Aug. 15–19, 2010. Springer, Berlin, Germany.
15. R. Cramer and V. Shoup. Signature schemes based on the strong RSA assumption. In *ACM CCS 99*, pages 46–51, Kent Ridge Digital Labs, Singapore, Nov. 1–4, 1999. ACM Press.
16. I. Damgård. Payment systems and credential mechanisms with provable security against abuse by individuals. In S. Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 328–335, Santa Barbara, CA, USA, Aug. 21–25, 1990. Springer, Berlin, Germany.
17. C. Fournet, M. Kohlweiss, G. Danezis, and Z. Luo. Zql: A compiler for privacy-preserving data processing. In *Proceedings of the 22Nd USENIX Conference on Security*, SEC'13, pages 163–178, Berkeley, CA, USA, 2013. USENIX Association.
18. R. Gennaro, C. Gentry, and B. Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In T. Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 465–482, Santa Barbara, CA, USA, Aug. 15–19, 2010. Springer, Berlin, Germany.
19. R. Gennaro, C. Gentry, B. Parno, and M. Raykova. Quadratic span programs and succinct NIZKs without PCPs. In *Eurocrypt '13: Proceedings of the 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 2013. Also in Cryptology ePrint Archive, Report 2012/215, `http://eprint.iacr.org/2012/215`.
20. R. Gennaro and D. Wichs. Fully homomorphic message authenticators. In *ASIACRYPT 2013*, 2013.

21. C. Gentry and D. Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In L. Fortnow and S. P. Vadhan, editors, *43rd ACM STOC*, pages 99–108, San Jose, California, USA, June 6–8, 2011. ACM Press.
22. C. Gentry and D. Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In *STOC '11: Proceedings of the 43rd Annual ACM Symposium on Theory of Computing*. ACM, 2011.
23. S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.
24. J. Groth. Short pairing-based non-interactive zero-knowledge arguments. In M. Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 321–340, Singapore, Dec. 5–9, 2010. Springer, Berlin, Germany.
25. R. Johnson, D. Molnar, D. X. Song, and D. Wagner. Homomorphic signature schemes. In B. Preneel, editor, *CT-RSA 2002*, volume 2271 of *LNCS*, pages 244–262, San Jose, CA, USA, Feb. 18–22, 2002. Springer, Berlin, Germany.
26. J. Kilian. A note on efficient zero-knowledge proofs and arguments. In *24th ACM STOC*, pages 723–732, Victoria, British Columbia, Canada, May 4–6, 1992. ACM Press.
27. J. Kilian. Improved efficient arguments (preliminary version). In D. Coppersmith, editor, *CRYPTO'95*, volume 963 of *LNCS*, pages 311–324, Santa Barbara, CA, USA, Aug. 27–31, 1995. Springer, Berlin, Germany.
28. H. Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In R. Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 169–189, Taormina, Sicily, Italy, Mar. 19–21, 2012. Springer, Berlin, Germany.
29. A. Lysyanskaya, R. L. Rivest, A. Sahai, and S. Wolf. Pseudonym systems. In H. M. Heys and C. M. Adams, editors, *SAC 1999*, volume 1758 of *LNCS*, pages 184–199, Kingston, Ontario, Canada, Aug. 9–10, 2000. Springer, Berlin, Germany.
30. S. Meiklejohn, C. C. Erway, A. Küpçü, T. Hinkle, and A. Lysyanskaya. Zkpdl: A language-based system for efficient zero-knowledge proofs and electronic cash. In *Proceedings of the 19th USENIX Conference on Security*, USENIX Security'10, pages 13–13, Berkeley, CA, USA, 2010. USENIX Association.
31. S. Micali. CS proofs. In *35th FOCS*, Santa Fe, New Mexico, Nov. 20–22, 1994.
32. M. Naor. On cryptographic assumptions and challenges (invited talk). In D. Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 96–109, Santa Barbara, CA, USA, Aug. 17–21, 2003. Springer, Berlin, Germany.
33. B. Parno, C. Gentry, J. Howell, and M. Raykova. Pinocchio: Nearly practical verifiable computation. In *IEEE Symposium on Security and Privacy, Oakland*, 2013. Corrected version (13 May 2013): `http://eprint.iacr.org/2013/279`.
34. B. Parno, M. Raykova, and V. Vaikuntanathan. How to delegate and verify in public: Verifiable computation from attribute-based encryption. In R. Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 422–439, Taormina, Sicily, Italy, Mar. 19–21, 2012. Springer, Berlin, Germany.
35. A. Rial and G. Danezis. Privacy-preserving smart metering. In *Proceedings of the 10th Annual ACM Workshop on Privacy in the Electronic Society*, WPES '11, pages 49–60, New York, NY, USA, 2011. ACM.
36. Vitalconnect. Healthpatch. `http://www.vitalconnect.com`, 2014.
37. B. R. Waters. Efficient identity-based encryption without random oracles. In R. Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 114–127, Aarhus, Denmark, May 22–26, 2005. Springer, Berlin, Germany.

A SNARGs

We recall the definition of SNARGs [31,22]. $\Pi = (\mathsf{Gen}, \mathsf{Prove}, \mathsf{Ver})$ is a *succinct non-interactive argument* for an $\mathcal{NP}$ language L with a corresponding $\mathcal{NP}$ relation R as follows:

- Given a relation R, the generation algorithm $\mathsf{Gen}(1^\lambda, R)$ generates a (public) reference string EK_R and the corresponding (private) verification information VK_R for R.
- Given statement x and witness w with $R(x, w)$, the prover produces a proof $\pi \leftarrow \mathsf{Prove}(\mathsf{EK}_R, x, w)$.
- The verifier runs $\{\bot, \top\} \leftarrow \mathsf{Ver}(\mathsf{VK}_R, x, \pi)$ to verify the validity of π.

A SNARG is called *adaptive* if the prover may choose the statement x after seeing the reference string EK_R. The following three properties need to be satisfied.

– **Completeness.** For all $(x, w) \in R$, we have that

$$\Pr[\mathsf{Ver}(\mathsf{VK}_R, x, \pi) = 0 : (\mathsf{EK}_R, \mathsf{VK}_R) \leftarrow \mathsf{Gen}(1^\lambda, R), \pi \leftarrow \mathsf{Prove}(\mathsf{EK}_R, x, w)] = negl(\lambda)$$

– **Soundness.** (Adaptive case) For all efficient Prove', we have

$$\Pr[\mathsf{Ver}(\mathsf{VK}_R, x, \pi) = \top \wedge x \notin L : (\mathsf{EK}_R, \mathsf{VK}_R) \leftarrow \mathsf{Gen}(1^\lambda, R), (x, \pi) \leftarrow \mathsf{Prove}'(\mathsf{EK}_R)] = negl(\lambda)$$

(Non-adaptive case) For all efficient Prove', and $x \notin L$, we have

$$\Pr[\mathsf{Ver}(\mathsf{VK}_R, x, \pi) = \top : (\mathsf{EK}_R, \mathsf{VK}_R) \leftarrow \mathsf{Gen}(1^\lambda, R), \pi \leftarrow \mathsf{Prove}'(\mathsf{EK}_R, x)] = negl(\lambda)$$

– **Succinctness.** The length of a proof π is given by $|\pi| = \text{poly}(\lambda)\text{polylog}(|x|, |w|)$.

Verifiable non-interactive computation on authenticated data via SNARGs. Verifiable computation over authenticated data [2] describes a setting in which a lightweight (possibly mobile) client outsources the evaluation of a function F over authenticated, possibly outsourced, data D to a more powerful (but untrusted) worker. Verification of the result $y = F(D)$ should require less work than the evaluation of $F(D)$ itself. Verification can be performed by the client who outsources the computation (designated verifiability), or by anyone using only public information (public verifiability).

Using SNARGs to implement a verifiable computation scheme, the idea is to encode possible values of $F : \{0,1\}^n \to \{0,1\}^m$ as an $\mathcal{NP}$-relation R as follows: for all x and y with $F(x) = y$, set $(x||y, w) \in R$. The client sends input x for the evaluation under F to the worker, who computes $y = F(x)$ and produces a corresponding witness w. The worker runs the Prove algorithm of the SNARG with the completed statement $x||y$ and witness w to obtain a succinct proof π. Finally, the worker returns y and π to the client, who in turn runs the SNARG verification algorithm.

In the case of authenticated data, the client does not send the input x itself to the worker, but instead, the client sends labels $\mathsf{L}_1, \ldots, \mathsf{L}_n$ that uniquely identify the authenticated input. The verification includes checks for the validity of the signatures to refer to the right computation input.

B The Pinocchio SNARG Scheme

We review the *corrected* SNARG version of the Pinocchio VC scheme, as published on the ePrint archive [33]. Pinocchio basically consists of the algorithms KeyGen, $\mathsf{Compute}$, and Verify, which are used in the context of verifiable computation. This section describes a small variation, where arbitrary $\mathcal{NP}$ relations $R \in \mathcal{R}$ are considered (instead of arithmetic functions), and where proofs are generated for statements x and witnesses w with $(x, w) \in R$ (instead of computation results for input u). The $\mathsf{Compute}$ algorithm is hence replaced by a Prove algorithm.

– $(\mathsf{EK}_R, \mathsf{VK}_R) \leftarrow \mathsf{KeyGen}(R, 1^\lambda)$: Let R be an $\mathcal{NP}$ relation with statements $x = (x_1, \ldots, x_a) \in \mathbb{F}^a$ and witnesses $w = (w_1, \ldots, w_b) \in \mathbb{F}^b$. Let $N = a + b$. Let C be R's characteristic circuit, i.e., $C(x, w) = 1$ whenever $(x, w) \in R$. Build the corresponding QAP $Q_R = (t(x), \mathcal{V}, \mathcal{W}, \mathcal{Y})$ for C with size m and degree d. Let $I_{mid} = \{a + 1, \ldots, a + b\} \cup \{N + 1, \ldots, m\}$ be the indices of the internal wires including the indices of the witness values. Let $I_{out} = \{m\}$ be the index of the

output wire. Let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a non-trivial bilinear map and let g be a generator of $\mathbb{G}$. Choose $r_v, r_w, s, \alpha_v, \alpha_w, \alpha_y, \beta, \gamma \leftarrow_{\mathcal{R}} \mathbb{F}$, set $r_y = r_v r_w$, and compute the following values:

$$\begin{aligned} & & T &= g^{r_y t(s)} & & \\ \forall k \in [m] \cup \{0\} : & & V_k &= g^{r_v v_k(s)}, & W_k &= g^{r_w w_k(s)}, & Y_k &= g^{r_y y_k(s)}, \\ \forall k \in I_{mid} : & & V'_k &= (V_k)^{\alpha_v}, & W'_k &= (W_k)^{\alpha_w}, & Y'_k &= (Y_k)^{\alpha_y}, & B_k &= (V_k\, W_k\, Y_k)^{\beta}. \end{aligned}$$

Additionally, compute the following values:

$$\begin{aligned} V_t &= g^{r_v t(s)}, & W_t &= g^{r_w t(s)}, & Y_t &= g^{r_y t(s)}, \\ V'_t &= (V_t)^{\alpha_v}, & W'_t &= (W_t)^{\alpha_w}, & Y'_t &= (Y_t)^{\alpha_y}, \\ B_v &= (V_t)^{\beta}, & B_w &= (W_t)^{\beta}, & B_y &= (Y_t)^{\beta}. \end{aligned}$$

Construct the public evaluation key and the public verification key

$$\begin{aligned} \mathsf{EK}_R = \Big(& \{V_k, V'_k, W_k, W'_k, Y_k, Y'_k, B_k\}_{k \in I_{mid}},\ \{g^{s^i}\}_{i \in [d]} \\ & V_t, V'_t,\ W_t, W'_t,\ Y_t, Y'_t,\ B_v, B_w, B_y,\ Q_R \Big) \\ \mathsf{VK}_R = \Big(& g,\ g^{\alpha_v}, g^{\alpha_w}, g^{\alpha_y}, g^{\gamma},\ g^{\beta\gamma},\ T,\ \{V_k,\, W_k,\, Y_k,\}_{k \in [N] \cup \{0,m\}} \Big) \end{aligned}$$

– $(\pi) \leftarrow \mathsf{Prove}(\mathsf{EK}_R, x, w)$: on input statement x and witness w, the prover evaluates the circuit $C(x, w)$ to obtain the internal circuit values $\{c_i\}_i \in I_{mid}$. For ease of description, we assume $c_i = x_i$ for $i \in [a]$, and $c_{a+i} = w_i$ for $i \in [b]$. The first b indices of I_{mid} hence represent the witness values w. Next, the prover computes the values

$$\begin{aligned} V_{mid} &= \prod_{k \in I_{mid}} (V_k)^{c_k}, & W_{mid} &= \prod_{k \in I_{mid}} (W_k)^{c_k}, & Y_{mid} &= \prod_{k \in I_{mid}} (Y_k)^{c_k}, \\ V'_{mid} &= \prod_{k \in I_{mid}} (V'_k)^{c_k}, & W'_{mid} &= \prod_{k \in I_{mid}} (W'_k)^{c_k}, & Y'_{mid} &= \prod_{k \in I_{mid}} (Y'_k)^{c_k}, & B_{mid} &= \prod_{k \in I_{mid}} (B_k)^{c_k} \end{aligned}$$

To make the proof zero-knowledge, pick random values $\delta^{(v)}_{mid}, \delta^{(w)}_{mid}, \delta^{(y)}_{mid} \leftarrow_{\mathcal{R}} \mathbb{F}$, and compute:

$$\begin{aligned} \tilde{V}_{mid} &= V_{mid} \cdot (V_t)^{\delta^{(v)}_{mid}}, & \tilde{W}_{mid} &= W_{mid} \cdot (W_t)^{\delta^{(w)}_{mid}}, & \tilde{Y}_{mid} &= Y_{mid} \cdot (Y_t)^{\delta^{(y)}_{mid}}, \\ \tilde{V}'_{mid} &= V'_{mid} \cdot (V'_t)^{\delta^{(v)}_{mid}}, & \tilde{W}'_{mid} &= W'_{mid} \cdot (W'_t)^{\delta^{(w)}_{mid}}, & \tilde{Y}'_{mid} &= Y'_{mid} \cdot (Y'_t)^{\delta^{(y)}_{mid}}, \\ \tilde{B}_{mid} &= B_{mid} \cdot (B_v)^{\delta^{(v)}_{mid}} \cdot (B_w)^{\delta^{(w)}_{mid}} \cdot (B_y)^{\delta^{(y)}_{mid}} \end{aligned}$$

Next, the prover solves the QAP Q_R by finding a polynomial $\tilde{h}(x)$ such that $\tilde{p}(x) = \tilde{h}(x) \cdot t(x)$ where the polynomial $\tilde{p}(x)$ includes the "perturbed versions" of the polynomials $v(x)$, $w(x)$, and $y(x)$:

$$\begin{aligned} \tilde{p}(x) = \Big(v_0(x) + \sum_{k \in [m]} c_k v_k(x) \ + \ \delta^{(v)}_{mid} t(x)\Big)\Big(w_0(x) + \sum_{k \in [m]} c_k w_k(x) \ + \ \delta^{(w)}_{mid} t(x)\Big) \\ - \Big(y_0(x) + \sum_{k \in [m]} c_k y_k(x) \ + \ \delta^{(y)}_{mid} t(x)\Big) \end{aligned}$$

Finally, the prover computes $\tilde{H} = g^{\tilde{h}(s)}$ using the values g^{s^i} contained in the evaluation key EK_R, and outputs $\tilde{\pi}_y = (\tilde{V}_{mid}, \tilde{V}'_{mid},\ \tilde{W}_{mid},\ \tilde{W}'_{mid},\ \tilde{Y}_{mid},\ \tilde{Y}'_{mid},\ \tilde{B}_{mid},\ \tilde{H})$.

– $\{0,1\} \leftarrow \mathsf{Verify}(\mathsf{VK}_R, x, \tilde{\pi})$: in order to verify a proof $\tilde{\pi}$ (as defined above) for statement x, perform the following steps.

(P.1) Check the satisfiability of the QAP by first computing $\tilde{V} = \tilde{V}_{mid} \cdot \prod_{k\in[a]}(V_k)^{c_k} \cdot V_m$, $\tilde{W} = \tilde{W}_{mid} \cdot \prod_{k\in[a]}(W_k)^{c_k} \cdot W_m$, $\tilde{Y} = \tilde{Y}_{mid} \cdot \prod_{k\in[a]}(Y_k)^{c_k} \cdot Y_m$, where the c_k with $k \in [a]$ are the statement wires of x. Second, perform the divisibility check:

$$e(V_0\ \tilde{V}\ ,\ W_0\ \tilde{W}\ ) = e(T,\ \tilde{H}\ ) \cdot e(Y_0\ \tilde{Y}\ ,\ g)$$

(P.2) Check that all linear combinations are in the appropriate spans:

$$e(\ \tilde{V}'_{mid}\ ,\ g) = e(\ \tilde{V}_{mid}\ ,\ g^{\alpha_v}) \quad \wedge \quad e(\ \tilde{W}'_{mid}\ ,\ g) = e(\ \tilde{W}_{mid}\ ,\ g^{\alpha_w}) \quad \wedge \quad e(\ \tilde{Y}'_{mid}\ ,\ g) = e(\ \tilde{Y}_{mid}\ ,\ g^{\alpha_y})$$

(P.3) Check that all the QAP linear combinations use the same coefficients:

$$e(\ \tilde{B}_{mid}\ ,\ g^{\gamma}) = e(\ \tilde{V}_{mid}\ \tilde{W}_{mid}\ \tilde{Y}_{mid}\ ,\ g^{\beta\gamma})$$

If all the checks above are satisfied, then return $\top$; otherwise return $\bot$.