Attribute-based Signatures for Unbounded Circuits in the ROM and Efficient Instantiations from Lattices

Ali El Kaafarani * and Shuichi Katsumata **

Abstract. Attribute-based signature (ABS), originally introduced by Maji et al. (CT-RSA'11), represents an essential mechanism to allow for fine-grained authentication. A user associated with an attribute x can sign w.r.t. a given public policy C only if his attribute satisfies C, i.e., $C(x) = 1$. So far, much effort on constructing bilinear map-based ABS schemes have been made, where the state-of-the-art scheme of Sakai et al. (PKC'16) supports the very wide class of *unbounded* circuits as policies. However, construction of ABS schemes without bilinear maps are less investigated, where it was not until recently that Tsabary (TCC'17) showed a lattice-based ABS scheme supporting *bounded* circuits as policies, at the cost of weakening the security requirement.
In this work, we affirmatively close the gap between ABS schemes based on bilinear maps and lattices by constructing the first lattice-based ABS scheme for *unbounded circuits* in the random oracle model. We start our work by providing a generic construction of ABS schemes for unbounded-circuits in the random oracle model, which in turn implies that one-way functions are sufficient to construct ABS schemes. To prove security, we formalize and prove a generalization of the Forking Lemma, which we call *"general multi-forking lemma with oracle access"*, capturing the situation where the simulator is interacting with some algorithms he *cannot* rewind, and also covering many features of the recent lattice-based ZKPs. This, in fact, was a formalization lacking in many existing anonymous signatures from lattices so far (e.g., group signatures). Therefore, this formalization is believed to be of independent interest. Finally, we provide a concrete instantiation of our generic ABS construction from lattices by introducing a new Σ-protocol, that highly departs from the previously known techniques, for proving possession of a valid signature of the lattice-based signature scheme of Boyen (PKC'10).

1 Introduction

1.1 Background

Attribute-based signature (ABS) was introduced by [MPR11] as a versatile tool allowing a signer to *anonymously* authenticate a message M w.r.t. a public signing policy C only if the signer has a signing key associated to an attribute

* University of Oxford, UK. E-mail: ali.elkaafarani@maths.ox.ac.uk
** The University of Tokyo, National Institute of Advanced Industrial Science and Technology (AIST). E-mail: shuichi_katsumata@it.k.u-tokyo.ac.jp

$x \in \{0,1\}^*$ that satisfies C, i.e., $C(x) = 1$. An attribute-based signature scheme reveals no information on the signer's identity or the attribute other than the fact that the signature is valid, hence the anonymity property of ABS schemes. One of the central research themes on ABS is to expand the expressiveness of the class of policies that can be supported by the schemes. In the bilinear map setting, there has been a long line of interesting works, including ABS schemes for threshold policy (e.g., [HLLR12]), boolean formula (e.g., [MPR11,OT11,OT13,EGK14]) and the current state-of-the-art; *unbounded circuits* [SAH16].[1]

On the other hand, the constructions of ABS schemes without bilinear maps, in particular ABS schemes from lattices, are much less investigated. To the best of our knowledge, there are only two major works concerning lattice-based ABS schemes [EE16,Tsa17]. Rachid et al. [EE16] construct a lattice-based ABS scheme for boolean formulas using a non-interactive zero-knowledge (NIZK) proof system as the main building block, following one of the most promising ways of constructing ABS schemes [MPR11,EGK14,SAH16]. Informally, a signature for a signer with attribute $\mathbf{x}$ is simply a zero-knowledge proof attesting to the fact that he has a certificate corresponding to the attribute $\mathbf{x}$ issued by the authority and that the policy C associated to the message M satisfies $C(\mathbf{x}) = 1$. Although this approach has been very effective in the bilinear map setting where [SAH16] were able to obtain ABS schemes for *unbounded circuits*, this has not been the case for lattices. One of the main reasons behind this is the lack of efficient lattice-based NIZK proof systems for a wide enough language. In particular, we only have efficient NIZK proof systems tailored for specific languages, such as proving possession of a solution to the short integer solution (SIS) problem or the learning with errors (LWE) problem [LNSW13], proving possession of a valid signature of the Boyen digital signature scheme [Boy10,LLNW14,LNW15] and so on, which in general does not seem strong enough for constructing ABS schemes. Recently, [YAL+17] showed (informally) how to construct lattice-based NIZK proof systems for languages accepted by monotone span programs, however, this still does not seem strong enough to use as a building block for ABS schemes supporting *unbounded* circuits as policies.

Tsabary [Tsa17] constructs lattice-based ABS schemes following a different approach; they show equivalence between a homomorphic signature (HS) scheme and a (message-policy) ABS scheme. Therefore, based on the HS construction of Gorbunov et al. [GVW15], they achieve a lattice-based ABS scheme for *bounded circuits* that does not make use of NIZK proof systems.[2] Here, by bounded, we mean that the required hardness assumptions on the LWE and/or SIS problems grow exponentially in the depth of the circuit, e.g., to base the security of the

[1] In our paper, we only consider *message-policy* ABS schemes. Recall that using universal circuits, we can convert message-policy ABS schemes into *key-policy* ABS schemes [BF14], where the functionality of the secret keys and messages are reversed.

[2] We note that the ABS scheme presented in [Tsa17] does not fulfill the standard security requirements of (message-policy) ABS schemes as originally defined in [MPR11]; achieving either unforgeability or anonymity in its full capacity comes at the cost of getting a much weaker version of the other, i.e., one has to choose between single-key-selective-unforgeability, or leaking information about the signing key.

ABS scheme under a polynomial LWE assumption, we need to restrict the depth of the circuit to be $O(\log \lambda)$, where λ is the security parameter. However, it seems challenging to improve their techniques to ABS schemes for unbounded circuits, due to the inherent noise-growth incurred by the homomorphic operations of matrices while computing the circuit gate-by-gate. The only known method of overcoming these $O(\log \lambda)$ depth barrier concerning homomorphic operations is the bootstrapping technique of fully homomorphic encryptions [Gen09], however, it is still an open problem whether there is a signature analogue of this technique.

1.2 Our Contribution

In this paper, we affirmatively close the gap between the state-of-the-art ABS schemes based on bilinear maps and lattices by constructing the first lattice-based ABS scheme for *unbounded circuits* in the random oracle model. We start by providing a general construction of ABS schemes supporting unbounded-circuits as policies. We then give an instantiation in the lattice setting showing that all the building blocks required by our generic construction is obtainable from lattices. We stress that, despite the expressiveness of the signing policy, we manage to prove the security of our scheme under surprisingly mild SIS and LWE assumptions with polynomial modulus size $q = \tilde{O}(\ell\lambda^{1.5})$, where ℓ denotes the length of the inputs to the circuits. Specifically, the required hardness assumptions are independent of the depth of the circuits that express the policies. Furthermore, the sizes of the public parameter, signing keys and signatures are $\tilde{O}(\ell\lambda^2)$, $\tilde{O}(\lambda)$ and $\tilde{O}((\ell\lambda+|C|)\lambda^2)$, respectively, where $|C|$ is the size of the circuit (i.e., policy) associated to the message.

To this end we prepare two new tools equipped for the lattice setting: we provide a generalization of the forking lemma of [PS00] which we call the *general multi-forking lemma with oracle access* and further construct a new lattice-based NIZK proof system for proving possession of a valid Boyen signature [Boy10] that departs from the previously known techniques (e.g., [LLNW14,LNW15]). Below, we give a more detailed overview of the techniques we used in our work.

Generic Construction of ABS for Unbounded Circuits. We propose a generic construction of ABS schemes supporting unbounded depth circuits as policies in the random oracle model[3], which employs the following primitives as its building blocks; a commitment scheme, a digital signature scheme and a Σ-protocol for a sufficiently wide relation. As a separate theoretical contribution, since all of the above primitives are implied from one-way functions, our result implies that one-way functions are sufficient to construct an ABS scheme for unbounded circuits in the random oracle model. Here, the random oracle is used only to convert the underlying Σ-protocol into a NIZK proof system via the Fiat-Shamir transformation [FS86].

At a high level, the generic construction of our ABS scheme follows closely the bilinear map based construction of [SAH16] (which is non-generic and proven in the standard model). We briefly review the construction in slightly more detail;

[3] In this paper, we only consider circuits that do not have random oracle gates.

first, the attribute authority issues a signature σ on an attribute $\mathbf{x} \in \{0,1\}^\ell$ to certify that a signer is indeed authorized to sign a message on behalf of that attribute. Then, to sign anonymously, the signer produces a zero-knowledge proof attesting to the following two facts:

(I) the signature σ issued by the authority is valid, and
(II) the corresponding secret attribute $\mathbf{x}$ satisfies the circuit C associated to the message M.

However, in spite of the similarities shared with the construction of [SAH16], the security proof of our construction requires a rather sensitive and technical analysis, which calls for new tools. This difficulty mainly stems from the fact that security proofs relying on the Fiat-Shamir-based NIZK proof systems are often times not as simple as the construction appears to be and in some cases the intuition may fail, e.g., [BPW12,BFW16].

Our proof of security of the generic ABS scheme relies on our generalization of the forking lemma of [PS00], which we call the *general multi-forking lemma with oracle access*. Our forking lemma can be seen as a generalization and a simplification of the general forking lemma of [BN06] and the improved forking lemma of [BPVY00]. In particular, we analyze the output behavior of an algorithm when run multiple times on related inputs, instead of when only run twice as in [BN06], while also providing it with oracle access to a deterministic algorithm. Recall that the original forking lemma of [PS00] applies to Fiat-Shamir type signature schemes and roughly states that, if there exists a valid forger $\mathcal{A}$, then one can rewind $\mathcal{A}$ initialized with the same randomness tape to find two accepting transcripts with the same commitment but different challenges, leading, via the special soundness property of Σ-protocols, to extract the secret signing key from the transcripts and hence a proof of security of the signature scheme in the random oracle model.

First, we require the forking lemma to analyze the output behavior of an algorithm on multiple runs to capture the situation arising in the recent lattice-based NIZK proof systems (e.g., [LNSW13,LLNW14,LNW15]) where the extractor of the underlying Σ-protocol requires more than two valid transcripts to extract a witness. Although the improved forking lemma of [BPVY00] captures this multiplicity of the forking lemma of a particular El Gamal-type signature scheme, it seems hard to apply in situations like ours where we are not dealing with regular signature schemes. Our forking lemma, similar to the one of [BN06], divorces the probabilistic essence of the forking lemma from any particular application context. Furthermore, our forking lemma provides worst-case rather than expected-time guarantees; the improved forking lemma of [BPVY00] roughly states that an expected $O(1/\epsilon)$ repeated executions of a forger $\mathcal{A}$ with advantage ϵ is required to extract a valid witness. We believe this feature to be more suitable for standard assumptions that are defined for PPT algorithms, as also stated in [BN06].

Second, and more importantly, our forking lemma allows the algorithm $\mathcal{A}$ that can be rewinded, to have oracle access to some algorithm $\mathcal{O}$ that *cannot* be rewinded. This is a useful feature for the forking algorithm to have in situations

where the simulator cannot rewind all the algorithms which he is interacting with. This may be easiest to explain with a concrete example; in particular, when we reduce the eu-cma security of the underlying digital signature scheme to the security of our ABS scheme, the simulator (which is the eu-cma adversary) simulates the view of an ABS security game to the ABS adversary $\mathcal{A}$, and answers the queries made by $\mathcal{A}$ using its eu-cma challenger $\mathcal{O}$. At some point when $\mathcal{A}$ outputs a forgery for the ABS security game, the simulator hopes to extract the witness from the forgery and use it to win his own eu-cma security game. However, for this particular situation, the problem with all the previous forking lemmas is that the simulator will not be able to run the forking algorithm in the specified way; the simulator *can* rewind $\mathcal{A}$ to a particular point where the fork happens, however, the simulator *cannot* rewind the eu-cma challenger $\mathcal{O}$ in the same way, since it is outside the simulator's (i.e., eu-cma adversary's) control. Then, since the behavior of $\mathcal{A}$ is implicitly dependent on the behavior of the eu-cma challenger, the standard forking lemma does not provide meaningful analysis of the output of $\mathcal{A}$ on multiple runs. We therefore present a general multi-forking lemma *with oracle access* to capture these situations where the simulator is restricted to rewinding only some of the algorithms he is interacting with. We note that in case one is willing to use some algebraic problem such as the SIS or LWE problem as the underlying hardness assumption, these situations do not show up, since once given a fixed problem instance, the simulator can reuse it in every run to simulate the view to $\mathcal{A}$.

Finally, one of the benefits of using the Fiat-Shamir-based NIZK proof system is that we do not have to rely on the dummy attribute technique of those ABS schemes based on Goth-Sahai NIZK proof systems [MPR11,SAH16] to prove adaptive unforgeability and hence obtaining a more efficient signing algorithm. At a high level, this is because Fiat-Shamir based NIZK proof systems can be simulation-sound and extractable at the same time, whereas Goth-Sahai NIZK proof systems can only be instantiated to have one of the two properties. Therefore, during the proof of adaptive unforgeability, since the simulator needs to set up the common reference string in the extractable mode to extract a witness from the forgery, the simulator has to rely on these extra dummy attributes, which are never used in the actual scheme, to simulate signatures (i.e., proofs).

Instantiation from Lattices. To instantiate our generic ABS construction from lattices, we require three primitives: a signature scheme, a commitment scheme, and a Σ-protocol for a relation capturing the aforementioned items (I) and (II). As for the signature scheme, we can use the simple and efficient lattice-based signature scheme of Boyen [Boy10], which has been extensively studied in the lattice-based NIZK literatures. In particular, Ling et al. [LNSW13] provides an efficient Σ-protocol for proving possession of a valid Boyen signature (i.e., item (I)). However, unfortunately, it is not known whether the Σ-protocol of Ling et al. can be extended to prove circuit satisfiability, which is what we require in item (II), and in fact, recent subsequent results of [LLM+16,YAL+17] suggest that they are not powerful enough to capture circuit satisfiability. On the other hand, Xie et al. [XXW13] provides a lattice-based Σ-protocol for proving NP

relations via arithmetic circuit satisfiability, which is what we exactly require in item (II), however, it does not seem possible to simply combine the two different types of Σ-protocols of [LNSW13] and [XXW13].

To this end, in this paper we present a new Σ-protocol for proving possession of a valid Boyen signature by expressing the verification algorithm of the Boyen signature as a simple arithmetic circuit that is compatible with the Σ-protocol of Xie et al. Specifically, since both items (I) and (II) can now be represented as arithmetic circuits, we can use the Σ-protocol of Xie et al. to obtain our desired Σ-protocol. The main observation is that, most operations that show up in lattice-based cryptography are composed of simple arithmetic operations such as matrix multiplications, and therefore naturally leads to simple arithmetic circuit representations. For our particular case, the verification algorithm of the Boyen signature scheme essentially boils down to checking two simple conditions; whether a vector $\mathbf{z}$ satisfies $\|\mathbf{z}\|_\infty \leq \beta$ and $\mathbf{Az} = \mathbf{u} \mod q$, where we intentionally dismiss the message for simplicity. As it can be seen, the latter equation is readily expressed by a very simple arithmetic circuit. On the other hand, the first inequality requires some extra work, however, this too can be expressed as an simple arithmetic circuit without much overhead by efficiently encoding predicates such as $x \stackrel{?}{\in} \{-1, 0, 1\}$ into arithmetic circuits.

2 Preliminaries

2.1 Commitment Schemes with Gap Openings

We define a standard commitment scheme that supports an additional notion we call *gap openings*. This additional notion will make it conceptually easier when we combine it with gap-Σ-protocols, which we later define. Informally, a commitment scheme with a gap opening is a standard commitment scheme where there may exist additional valid openings that are never created during the commitment algorithm.

Definition 1 (Commitments). *A commitment scheme with message space $\mathcal{M}$ and commitment space $\mathcal{C}$ is a triple of PPT algorithms* $(\mathsf{C.Gen}, \mathsf{C.Com}, \mathsf{C.Open})$ *of the following form:*

- $\mathsf{C.Gen}(1^\lambda) \to \mathsf{pk}$: *The key generation algorithm takes as input the security parameter 1^λ and outputs a public commitment key* pk.
- $\mathsf{C.Com}(\mathsf{pk}, \mathsf{M}) \to (c, d)$: *The commitment algorithm takes as inputs the commitment key* pk *and message $\mathsf{M} \in \mathcal{M}$, and outputs a commitment/opening pair (c, d). We denote $\mathcal{D}_{\mathsf{Com}}(\mathsf{pk}, \mathsf{M})$ as the set of all possible outputs of this algorithm under fixed* pk *and* M.
- $\mathsf{C.Open}(\mathsf{pk}, \mathsf{M}, c, d) \to 1 \backslash 0$: *The* deterministic *opening algorithm takes as inputs the commitment key* pk, *message* M *and commitment/opening pair (c, d) as inputs and outputs 1 or 0. We denote $\mathcal{D}_{\mathsf{G\text{-}Com}}(\mathsf{pk}, \mathsf{M})$ as the set of all possible pairs (c, d) this algorithm outputs 1 under fixed* pk *and* M.

Here, we require the commitment scheme to satisfy the following correctness notion: for all $\mathsf{M} \in \mathcal{M}, \mathsf{pk} \leftarrow \mathsf{C.Gen}(1^\lambda), (c, d) \leftarrow \mathsf{C.Com}(\mathsf{pk}, \mathsf{M})$ *we have* $\mathsf{C.Open}(\mathsf{pk}, \mathsf{M}, c, d) = 1$.

It is clear that we have $\mathcal{D}_{\mathsf{Com}}(\mathsf{pk}, \mathsf{M}) \subseteq \mathcal{D}_{\mathsf{G\text{-}Com}}(\mathsf{pk}, \mathsf{M})$ for all pk and $\mathsf{M} \in \mathcal{M}$. We say the commitment scheme has a *gap-opening* when $\mathcal{D}_{\mathsf{Com}} \subset \mathcal{D}_{\mathsf{G\text{-}Com}}$. We require the following security notions for a commitment scheme:

Binding. We call the scheme unconditionally (resp. computationally) binding if for all (resp. PPT) algorithm $\mathcal{A}$, we have the following:

$$\Pr[\mathsf{pk} \leftarrow \mathsf{C.Gen}(1^\lambda); (c, \mathsf{M}, \mathsf{M}', d, d') \leftarrow \mathcal{A}(\mathsf{pk}) : \mathsf{C.Open}(\mathsf{pk}, \mathsf{M}, c, d) = \mathsf{C.Open}(\mathsf{pk}, \mathsf{M}', c, d') = 1 \wedge \mathsf{M} \neq \mathsf{M}'] \leq \mathsf{negl}(\lambda)$$

Note that even though such a pair (c, d) may never be outputted by the commitment algorithm $\mathsf{C.Com}$, the binding property must hold even for adversaries that output $(c, d) \in \mathcal{D}_{\mathsf{G\text{-}Com}}(\mathsf{pk}, \mathsf{M}) \backslash \mathcal{D}_{\mathsf{Com}}(\mathsf{pk}, \mathsf{M})$.

Hiding. We call the scheme unconditionally (resp. computationally) hiding if for all (resp. PPT) algorithm $\mathcal{A}$ and any message $\mathsf{M} \in \mathcal{M}$, we have the following:[4]

$$\Pr[\mathsf{pk} \leftarrow \mathsf{C.Gen}(1^\lambda);\ b \leftarrow \{0,1\};\ c_0 \leftarrow \mathcal{C};\ (c_1, d) \leftarrow \mathsf{C.Com}(\mathsf{pk}, \mathsf{M});\ b' \leftarrow \mathcal{A}(\mathsf{pk}, \mathsf{M}, c_b) : b = b'] \leq 1/2 + \mathsf{negl}(\lambda)$$

2.2 Digital Signature Schemes.

In this paper, we use *deterministic* digital signature schemes; a scheme where the randomness of the signing algorithm is derived from the secret key and message. We briefly recall the standard syntax and security notions, and refer the full version for the exact definition. A deterministic digital signature scheme is a tuple of PPT algorithms $(\mathsf{S.KeyGen}, \mathsf{S.Sign}, \mathsf{S.Verify})$, such that the key generation algorithm $\mathsf{S.KeyGen}$ outputs a verification key vk and a signing key sk. The *deterministic* signing algorithm $\mathsf{S.Sign}$ on input the signing key sk and a message $\mathbf{x}$ outputs a signature σ, and the verification algorithm $\mathsf{S.Verify}$ verifies the signature σ using the verification key vk. We consider the standard security notion of existential unforgeability under an adaptive chosen message attack ($\mathsf{eu\text{-}cma}$).

2.3 Arithmetic Circuit Representation

Here, we explain how we represent an arithmetic circuit. Let C be an arithmetic circuit over a ring R having ℓ input wires, one output wire and N gates. Here the gates are labelled by either $+$ (addition) or $\times$ (product) gates. The input wires are indexed by $1, \cdots, \ell$, the internal wires are indexed by $\ell+1, \cdots, \ell+N-1$ and the output wire has index $\ell + N$. We assume each gate takes only two incoming

[4] We assume that the commitment space $\mathcal{C}$ is efficiently sampleable. Namely, as long as the hiding property holds, $\mathcal{C}$ may be larger than the set of all possible commitments. These situations come up in many of the lattice-based commitment schemes.

wires with multiple fan-out wires, where all the fan-out wires are indexed with the same index. We specify the topology of an arithmetic circuit by a function $\mathsf{topo} : \{\ell+1, \cdots, \ell+N\} \to \{+, \times\} \times \{1, \cdots, \ell+N-1\} \times \{1, \cdots, \ell+N-1\}$. They map a non-input wire to its first and second incoming wires in which these three wires are connected by either a gate labelled by $+$ or $\times$. For $(\star, i_1, i_2) \leftarrow \mathsf{topo}(i)$, we require that $i_1, i_2 < i$ where $\star \in \{+, \times\}$.

2.4 Attribute-Based Signature Scheme

An attribute-based signature scheme supporting the class of arithmetic circuits $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ and message space $\{0,1\}^*$ is defined by the following four probabilistic polynomial time algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$:

- $\mathsf{Setup}(1^\lambda, 1^\ell) \to (\mathsf{mpk}, \mathsf{msk})$: The setup algorithm takes as input the security parameter 1^λ and the input length 1^ℓ of the circuits in $\mathcal{C}_\ell$, and outputs the master public key mpk and the master secret key msk.
- $\mathsf{KeyGen}(\mathsf{mpk}, \mathsf{msk}, \mathbf{x}) \to \mathsf{sk}_\mathbf{x}$: The signing key generation algorithm takes as input the master public key mpk, the master secret key msk and an attribute $\mathbf{x} \in \{0,1\}^\ell$, and outputs a signing key $\mathsf{sk}_\mathbf{x}$.
- $\mathsf{Sign}(\mathsf{mpk}, \mathsf{sk}_\mathbf{x}, C, \mathsf{M}) \to \Sigma$: The signing algorithm takes as input the master public key mpk, a secret key $\mathsf{sk}_\mathbf{x}$ associated with an attribute $\mathbf{x}$, a circuit $C \in \mathcal{C}_\ell$ and a message $\mathsf{M} \in \{0,1\}^*$, and outputs a signature σ.
- $\mathsf{Verify}(\mathsf{mpk}, \mathsf{M}, C, \Sigma) \to \mathsf{Valid}/\mathsf{Invalid}$: The verification algorithm takes as input the master public key mpk, a message M, a circuit C and a signature Σ, and outputs Valid or $\mathsf{Invalid}$.

Correctness. We require the following correctness condition to hold: for all $\lambda, \ell \in \mathbb{N}$, $\mathbf{x} \in \{0,1\}^\ell$, $C \in \mathcal{C}_\ell$ such that $C(\mathbf{x}) = 1$, it holds with all but negligible probability that $\mathsf{Verify}(\mathsf{mpk}, \mathsf{M}, C, \mathsf{Sign}(\mathsf{mpk}, \mathsf{sk}_\mathbf{x}, C, \mathsf{M})) = \mathsf{Valid}$, where the probability is taken over the randomness used in $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$ and $\mathsf{sk}_\mathbf{x} \leftarrow \mathsf{KeyGen}(\mathsf{mpk}, \mathsf{msk}, \mathbf{x})$.

We require two types of security notions for attribute-based signature schemes.

Definition 2 (Privacy). *The security notion of privacy for an attribute-based signature scheme is defined by the following game between a challenger and an adversary $\mathcal{A}$:*

Setup. *The challenger runs* $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$ *and gives* $(\mathsf{mpk}, \mathsf{msk})$ *to $\mathcal{A}$.*

Challenge. *$\mathcal{A}$ outputs a message* $\mathsf{M} \in \{0,1\}^*$, *two attributes* $\mathbf{x}_0, \mathbf{x}_1 \in \{0,1\}^\ell$ *and a circuit* $C \in \mathcal{C}_\ell$ *such that* $C(\mathbf{x}_0) = C(\mathbf{x}_1) = 1$ *to the challenger. The challenger first runs* $\mathsf{sk}_{\mathbf{x}_\beta} \leftarrow \mathsf{KeyGen}(\mathsf{mpk}, \mathsf{msk}, \mathbf{x}_\beta)$ *for* $\beta = 0, 1$. *Then, it picks a random bit* $b \leftarrow \{0,1\}$ *and returns to $\mathcal{A}$ the signature* $\Sigma^* \leftarrow \mathsf{Sign}(\mathsf{mpk}, \mathsf{sk}_{\mathbf{x}_b}, C, \mathsf{M})$ *along with the two secret keys* $(\mathsf{sk}_{\mathbf{x}_0}, \mathsf{sk}_{\mathbf{x}_1})$.

Forgery. *Finally, $\mathcal{A}$ outputs a guess* $b' \in \{0,1\}$ *for* b.

The advantage of $\mathcal{A}$ is defined as $|\Pr[b' = b] - 1/2|$. We say that the attribute-based signature scheme is computationally private *if the advantage of any PPT algorithm $\mathcal{A}$ is negligible. We say it is* unconditionally private *if the advantage of any (possibly inefficient) algorithm $\mathcal{A}$ is negligible.*

Definition 3 (Unforgeability). *The security notion of adaptively unforgeable for an attribute-based signature scheme is defined by the following game between a challenger and an adversary $\mathcal{A}$:*

Setup. *The challenger runs* $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$ *and gives* mpk *to* $\mathcal{A}$.
Queries. $\mathcal{A}$ *may adaptively make the following queries to the challenger:*
- **Signing.** $\mathcal{A}$ *submits a signing query on any attribute, message and circuit tuple* $(\mathbf{x}, \mathsf{M}, C)$ *such that* $C(\mathbf{x}) = 1$ *to the challenger. The challenger runs* $\mathsf{sk}_{\mathbf{x}} \leftarrow \mathsf{KeyGen}(\mathsf{mpk}, \mathsf{msk}, \mathbf{x})$. *Then, it returns the signature* $\Sigma \leftarrow \mathsf{Sign}(\mathsf{mpk}, \mathsf{sk}_{\mathbf{x}}, C, \mathsf{M})$ *to* $\mathcal{A}$.
- **Key reveal.** $\mathcal{A}$ *submits a key reveal query on any attribute* $\mathbf{x}$ *to the challenger. The challenger returns the signing key* $\mathsf{sk}_{\mathbf{x}} \leftarrow \mathsf{KeyGen}(\mathsf{mpk}, \mathsf{msk}, \mathbf{x})$ *to* $\mathcal{A}$.

Forgery. *Finally,* $\mathcal{A}$ *outputs a signature* $(\mathsf{M}^*, C^*, \Sigma^*)$.

The adversary $\mathcal{A}$ wins the game if the following three conditions hold:

(i) $\mathsf{Verify}(\mathsf{mpk}, \mathsf{M}^*, C^*, \Sigma^*) = \mathsf{Valid}$,
(ii) *Adversary $\mathcal{A}$ did not submit a key reveal query for $\mathbf{x}$ such that $C^*(\mathbf{x}) = 1$,*
(iii) *Adversary $\mathcal{A}$ did not submit a signing query on $(\mathbf{x}, \mathsf{M}^*, C^*)$ for any $\mathbf{x}$ such that $C^*(\mathbf{x}) = 1$*

The advantage of $\mathcal{A}$ is defined as the probability of $\mathcal{A}$ winning the above game. We say that the attribute-based signature scheme is adaptively unforgeable *if the advantage of any PPT algorithm $\mathcal{A}$ is negligible.*

2.5 General Multi-Forking Lemma with Oracle Access

Here we state and prove an extended version of the forking lemma of [PS00], which will play a central role in our proof of security of our ABS scheme. Our forking lemma analyzes the output behavior of an algorithm $\mathcal{A}$ when run multiple times on related inputs, instead of when only run twice, while also providing it with oracle access to a *deterministic* algorithm $\mathcal{O}$.

Lemma 1 (General Multi-Forking Lemma with Oracle Access). *Fix an integer $q \geq 1$ and a set $\mathcal{H}$ of size $h \geq 2$. Let $\mathcal{A}$ be a randomized algorithm that has oracle access to some* deterministic *algorithm $\mathcal{O}$, where on input* $\mathsf{par}, h_1, \cdots, h_q$, *algorithm $\mathcal{A}$ returns a pair; the first element is an integer in the range $0, \cdots, q$ and the second element is what we refer to as a* side output. *Let* IG *be a randomized algorithm called the* input generator. *The accepting probability of $\mathcal{A}$, denoted* acc, *is defined as the probability that $J \geq 1$ in the experiment below:*

$$(\mathsf{par}, \overline{\mathsf{par}}) \leftarrow \mathsf{IG};\ h_1, \cdots, h_q \leftarrow \mathcal{H};\ (J, \sigma) \leftarrow \mathcal{A}^{\mathcal{O}(\overline{\mathsf{par}}, \cdot)}(\mathsf{par}, h_1, \cdots, h_q).$$

For a positive integer $\ell \geq 2$, the forking algorithm $\mathsf{F}_{\mathcal{A},\ell}^{\mathcal{O}(\overline{\mathsf{par}},\cdot)}$ *associated to* $\mathcal{A}^{\mathcal{O}(\overline{\mathsf{par}},\cdot)}$ *is a randomized oracle algorithm that takes input* par *and proceeds as in Fig. 1, where* $\{\epsilon_k\}_{k\in[\ell]}$ *denotes an arbitrary set of strings. Let*

$$\mathsf{frk} = \Pr[(\mathsf{par}, \overline{\mathsf{par}}) \leftarrow \mathsf{IG};\ (b, \{\sigma_k\}_{k\in[\ell]}) \leftarrow \mathsf{F}_{\mathcal{A},\ell}^{\mathcal{O}(\overline{\mathsf{par}},\cdot)}(\mathsf{par})\ :\ b = 1].$$

Then,

$$\mathsf{frk} \geq \mathsf{acc} \cdot \left(\left(\frac{\mathsf{acc}}{q} \right)^{\ell-1} - \frac{f(\ell)}{h} \right), \tag{1}$$

where $f(\ell)$ is some universal positive valued function that only depends on the value ℓ.

Algorithm $\mathsf{F}_{\mathcal{A},\ell}^{\mathcal{O}(\overline{\mathsf{par}},\cdot)}(\mathsf{par})$

Pick coin ρ for $\mathcal{A}$ at random.
$h_1^{(1)}, \cdots, h_q^{(1)} \leftarrow \mathcal{H}$
$(I^{(1)}, \sigma^{(1)}) \leftarrow \mathcal{A}^{\mathcal{O}(\overline{\mathsf{par}},\cdot)}(\mathsf{par}, h_1^{(1)}, \cdots, h_q^{(1)}; \rho)$
if $I^{(1)} = 0$ **then return** $(0, \{\epsilon_k\}_{k\in[\ell]})$
for $k = 2$ to ℓ **do**
    $h_{I^{(1)}}^{(k)}, \cdots, h_q^{(k)} \leftarrow \mathcal{H}$
    $(I^{(k)}, \sigma^{(k)}) \leftarrow \mathcal{A}^{\mathcal{O}(\overline{\mathsf{par}},\cdot)}(\mathsf{par}, h_1^{(1)}, \cdots, h_{I^{(1)}-1}^{(1)}, h_{I^{(1)}}^{(k)}, \cdots h_q^{(k)}; \rho)$
if $I^{(1)} = I^{(k)}$ and $h_{I^{(1)}}^{(k)} \neq h_{I^{(1)}}^{(k')}$ for all $k, k' \in [\ell]$ **then**
    return $(1, \{\sigma^{(k)}\}_{k\in[\ell]})$
else
    return $(0, \{\epsilon_k\}_{k\in[\ell]})$.

Fig. 1. Description of the forking algorithm $\mathsf{F}_{\mathcal{A},\ell}^{\mathcal{O}(\overline{\mathsf{par}},\cdot)}$.

Proof. For any input $x = (\mathsf{par}, \overline{\mathsf{par}})$, denote $\mathsf{acc}(x)$ as the probability that $J \geq 1$ in the following experiment:

$$h_1, \cdots, h_q \leftarrow H;\ (J, \sigma) \leftarrow \mathcal{A}^{\mathcal{O}(\overline{\mathsf{par}},\cdot)}(\mathsf{par}, h_1, \cdots, h_q).$$

Also, let $\mathsf{frk}(x) = \Pr[(b, \{\sigma_k\}_{k\in[\ell]}) \leftarrow \mathsf{F}_{\mathcal{A},\ell}^{\mathcal{O}(\overline{\mathsf{par}},\cdot)}(\mathsf{par})\ :\ b = 1]$. We claim that there exists some universal positive valued function $f(\ell)$ such that for all x,

$$\mathsf{frk}(x) \geq \mathsf{acc}(x) \cdot \left(\left(\frac{\mathsf{acc}(x)}{q} \right)^{\ell-1} - \frac{f(\ell)}{h} \right). \tag{2}$$

By taking the expectation of $\mathsf{frk}(x)$ over $x = (\mathsf{par}, \overline{\mathsf{par}}) \leftarrow \mathsf{IG}$ and using the fact $\mathbb{E}[\mathsf{acc}(x)^\ell] \geq \mathbb{E}[\mathsf{acc}(x)]^\ell$ (which follows from Jensen's inequality), we obtain Eq. (1). Therefore, to prove the claim, we must prove Eq. (2). Now, for any

input x, with the probabilities taken over the coin tosses of $\mathsf{F}^{\mathcal{O}(\overline{\mathsf{par}},\cdot)}_{\mathcal{A},\ell}(\mathsf{par})$, $\mathsf{frk}(x)$ is equivalent to the following.

$$\begin{aligned}
&\Pr\left[(I^{(1)} = I^{(k)} \text{ for all } k \in [\ell]) \wedge (I^{(1)} \geq 1) \wedge (h^{(k)}_{I^{(1)}} \neq h^{(k')}_{I^{(1)}} \text{ for all } k, k' \in [\ell])\right] \\
\geq &\Pr\left[(I^{(1)} = I^{(k)} \text{ for all } k \in [\ell]) \wedge (I^{(1)} \geq 1)\right] \\
&\qquad - \Pr\left[(I^{(1)} \geq 1) \wedge (h^{(k)}_{I^{(1)}} = h^{(k')}_{I^{(1)}} \text{ for some } k, k' \in [\ell])\right] \\
= &\Pr\left[(I^{(1)} = I^{(k)} \text{ for all } k \in [\ell]) \wedge (I^{(1)} \geq 1)\right] - \Pr\left[(I^{(1)} \geq 1)\right] \cdot (1 - \prod_{k=1}^{\ell-1} \frac{h-k}{h})
\end{aligned}$$

Here, we can rewrite $1 - \prod_{k=1}^{\ell-1} \frac{h-k}{h} = \frac{1}{h} \cdot \left(\sum_{k=0}^{\ell-2} \alpha_k(\ell) \cdot \frac{1}{h^k} \right)$, where $(\alpha_k(\ell))_{k=0}^{\ell-2}$ are functions that only depend on ℓ. Since $h \geq 1$, we can always upper bound the right hand side by $f(\ell)/h$ using some positive valued function $f(\ell)$ that only depends on ℓ, where for example, we can use $f(\ell) = (\ell - 1) \cdot \max\{|\alpha_k(\ell)|\}_{k=0}^{\ell-2}$. Here, note that $f(\ell)$ is some universal function that depends neither on $\mathcal{A}$ nor $\mathcal{O}$. Therefore, we can further rewrite the inequality as follows:

$$\mathsf{frk}(x) \geq \Pr\left[(I^{(1)} = I^{(k)} \text{ for all } k \in [\ell]) \ \wedge \ (I^{(1)} \geq 1)\right] - \frac{\mathsf{acc}(x) \cdot f(\ell)}{h}.$$

Hence, it remains to show that $\Pr\left[\left(I^{(1)} = I^{(k)} \text{ for all } k \in [\ell]\right) \ \wedge \ \left(I^{(1)} \geq 1\right)\right] \geq \mathsf{acc}(x)^\ell / q^{\ell-1}$. Let $\mathcal{R}$ denote the set from which $\mathcal{A}$ draws its random coins. For each $i \in [q]$, let $X_i : \mathcal{R} \times \mathcal{H}^{i-1} \to [0, 1]$ be defined by setting $X_i(\rho, h_1, \cdots, h_{i-1})$ to

$$\Pr[h_i, \cdots, h_q \leftarrow \mathcal{H} \ ; \ (J, \sigma) \leftarrow \mathcal{A}^{\mathcal{O}(\overline{\mathsf{par}},\cdot)}(\mathsf{par}, h_1, \cdots, h_q; \rho) \ : \ J = i]$$

for all $\rho \in \mathcal{R}$ and $h_1, \cdots, h_{i-1} \in \mathcal{H}$. Here, regard X_i as a random variable over the uniform distribution on its domain. Then,

$$\begin{aligned}
\Pr\left[(I^{(1)} = I^{(k)} \text{ for all } k \in [\ell]) \ \wedge \ (I^{(1)} \geq 1)\right] &= \sum_{i=1}^{q} \Pr\left[I^{(k)} = i \text{ for all } k \in [\ell]\right] \\
&= \sum_{i=1}^{q} \left(\Pr[I^{(1)} = i] \cdot \prod_{k=2}^{\ell} \Pr[I^{(k)} = i \mid I^{(1)} = i] \right) \qquad (3) \\
&= \sum_{i=1}^{q} \sum_{\rho, h_1, \cdots, h_{i-1}} X_i(\rho, h_1, \cdots, h_{i-1})^\ell \cdot \frac{1}{|\mathcal{R}| \cdot |\mathcal{H}|^{i-1}} \\
&= \sum_{i=1}^{q} \mathbb{E}[X_i^\ell] \quad \geq \quad \sum_{i=1}^{q} \mathbb{E}[X_i]^\ell. \qquad (4)
\end{aligned}$$

Here Eq. (3) follows from independence of $I^{(k)}$ and $I^{(k')}$ for $k, k' \in [2, \ell]$, and Eq. (4) follows from Jensen's inequality where we use the fact that $f(x) = x^\ell$ is a convex function. Finally, using Holder's inequality, we obtain

$$\sum_{i=1}^{q} \mathbb{E}[X_i]^\ell \geq \frac{1}{q^{\ell-1}} \cdot \left(\sum_{i=1}^{q} \mathbb{E}[X_i] \right)^\ell = \frac{1}{q^{\ell-1}} \cdot \mathsf{acc}(x)^\ell.$$

This completes the proof of Eq. (1), hence concluding our claim.

Remarks. As can be checked from the proof, we can set the function $f(\ell)$ so that in case $\ell = 2$, we have $f(2) = 1$. Therefore, by setting the deterministic oracle $\mathcal{O}$ to be an oracle that outputs nothing, the above lemma implies the general forking lemma of [BN06].

3 Gap-Σ-Protocols and Non-Interactive Zero-Knowledge Proofs

Before presenting the main tools we use in this paper, we first recall the definition of languages and relations. A *language* $\mathcal{L} \subseteq \{0,1\}^*$ is said to have polynomial time recognizable *relation* $\mathcal{R} \subseteq \{0,1\}^* \times \{0,1\}^*$ if $\mathcal{L} = \{x \mid \exists w \text{ s.t. } (x,w) \in \mathcal{R}\}$ where $|w| \leq \mathsf{poly}(|x|)$. We call the string w a *witness* to the *statement* $x \in \mathcal{L}$.

3.1 Gap-Σ-Protocols

Σ-protocols are a special type of 3-round interactive proof systems that is also a proof of knowledge. Below, we define (a special type of) the *gap-Σ-protocol*, which is a generalization of the standard Σ-protocol where we allow the extracted witness to lie in a slightly larger space than the actual witness being proven during the protocol. Furthermore, the special soundness is defined for cases where more than 2 valid transcripts are required to extract a witness. These non-standard formalizations are required, since most of the lattice-based Σ-protocols are not captured by the standard formalizations.

Definition 4 (Gap-Σ-protocols). *Let m be an integer constant and t an integer-valued function of the security parameter. Let $(\mathcal{P}, \mathcal{V})$ be a two-party protocol, where $\mathcal{V}$ is PPT, and let $\mathcal{L}, \mathcal{L}' \subseteq \{0,1\}^*$ be languages with witness relations $\mathcal{R}, \mathcal{R}'$ such that $\mathcal{R} \subseteq \mathcal{R}'$. Then $(\mathcal{P}, \mathcal{V})$ is called a gap-$\Sigma_{m,t}$-protocol for relations $(\mathcal{R}, \mathcal{R}')$ with challenge space $\mathcal{C} = \{0, 1, \cdots, m-1\}^t$, if it satisfies the following conditions:*

- **3-move form:** *The protocol is of the following form:*
  - *The prover $\mathcal{P}$, on input $(x,w) \in \mathcal{R}$, sends a* commitment *α to $\mathcal{V}$.*
  - *The verifier $\mathcal{V}$ samples a* challenge *$\beta \leftarrow \mathcal{C}$ and sends it to $\mathcal{P}$.*
  - *The prover $\mathcal{P}$ sends a* response *γ to $\mathcal{V}$, and $\mathcal{V}$ decides to accept of reject based on the protocol transcript (α, β, γ).*

  The protocol transcript (α, β, γ) is called a valid transcript *if the verifier $\mathcal{V}$ accepts the protocol run.*
- **Completeness:** *Whenever $(x,w) \in \mathcal{R}$, $\mathcal{V}$ accepts with probability 1.*
- **Soundness:** *If $(x,w) \notin \mathcal{R}$, then any cheating (possibly inefficient) prover $\mathcal{P}^*$ succeeds with probability at most $(\frac{m-1}{m})^t$. We call this value the* soundness error*.*

- **Special gap-soundness:** *There exists a PPT algorithm $\mathcal{E}$ (the* knowledge extractor*) which takes m valid transcripts $\{(\alpha, \beta_i, \gamma_i)\}_{i\in[m]}$ for some statement $x \in \mathcal{L}$, where there exists at least one index $j \in [t]$ such that $\{\beta_{i,j}\}_{i\in[m]} = \{0, 1, \cdots, m-1\}$ as inputs, and outputs w such that $(x, w) \in \mathcal{R}'$. Here $\beta_{i,j}$ denotes the j-th value of the string β_i. Note that the knowledge extractor outputs a witness in the* gap *relation.*
- **Special honest-verifier zero-knowledge (HVZK):** *There exists a PPT algorithm $\mathcal{S}$ (the* HVZK simulator*) taking $x \in \mathcal{L}$ as input, that outputs (α, β, γ) whose distribution is indistinguishable from an accepting protocol transcript generated by a real protocol run. Although no guarantees on the outputs are made, the simulator $\mathcal{S}$ is also defined over the inputs $x \notin \mathcal{L}$.*

We call the gap-$\Sigma_{m,t}$-protocol computationally (resp. statistically) special HVZK if the simulated transcript is computationally (resp. statistically) indistinguishable from a real transcript.

Lastly, we say the gap-Σ-protocol has high-commitment entropy *if for all $(x, w) \in \mathcal{R}$ and α, the probability that an honestly generated commitment by $\mathcal{P}$ takes on the value α is negligible.*

We omit the subscript (m, t) of the gap-$\Sigma_{m,t}$-protocol whenever it is irrelevant to the context. Occasionally, we omit t and simply write gap-Σ_m-protocol to emphasize that the soundness error is negligible in the security parameter. We note that the standard Σ-protocol is a special case of the gap-Σ-protocol where $m = 2, \mathcal{R} = \mathcal{R}'$. In this case the soundness error will simply be 2^{-t} and special gap-soundness implies special soundness, since if there exists an index $j \in [t]$ for which the binary strings (i.e., the challenges) differ, then it implies that the two challenges are different. Finally, we assume without loss of generality that all of the gap-Σ-protocols we consider in this paper have high-commitment entropy, since the condition can be easily met by appending a super-logarithmic number of public random bits to the commitments.

Often times, the gap in the relations allows for much more efficient schemes, and do not affect their usefulness in practice as long as $\mathcal{R}'$ is still a sufficiently hard relation, e.g., [FO97,DF02,AJLA+12,BCK+14]. We note that for simplicity, in this paper we only consider gap-Σ-protocols that are complete with probability 1. Namely, our formalization does not capture those gap-Σ-protocols that are based on the rejection sampling technique such as [Lyu09,Lyu12,BCK+14].[5]

Finally, we formally describe the Fiat-Shamir transformation [FS86] which is a technique to make any (gap-)Σ-protocol into a non-interactive proof system by using a cryptographic hash function.

[5] Note that we intentionally disregard [BKLP15] from our work. Although they offer an attractive rejection sampling-based gap-Σ-protocol for proving arbitrary arithmetic operations that are more efficient than those of [XXW13] which we use in Sec. 5, we were not able to verify the correctness of their proof sketch. In particular, the knowledge extractor for the protocol for proving multiplicative relations could not be constructed as stated in their paper.

Definition 5. *Let $(\mathcal{P}, \mathcal{V})$ be a gap-Σ-protocol with relation $(\mathcal{R}, \mathcal{R}')$, and $H(\cdot)$ a hash function with range equal to the verifier's challenge space $\mathcal{C}$. The* Fiat-Shamir *transformation of gap-Σ is the non-interactive proof system $(\mathcal{P}^H, \mathcal{V}^H)$ defined as follows:*

$\mathcal{P}^H(x, w)$: *Run $\mathcal{P}(x, w)$ to obtain a commitment α, and compute $\beta \leftarrow H(x, \alpha)$. Then complete the run of $\mathcal{P}$ with β as the challenge to get the response γ. Finally output the pair (α, γ)*

$\mathcal{V}^H(x, \alpha, \gamma)$: *Compute $\beta = H(x, \alpha)$ and return the output of $\mathcal{V}(\alpha, \beta, \gamma)$.*

3.2 Non-Interactive Zero-Knowledge Proof Systems

We formalize the notion of non-interactive zero-knowledge (NIZK) proof systems in the *explicitly programmable* random oracle model [Wee09], where the zero-knowledge (ZK) simulator is allowed to explicitly program the random oracle. We follow the notations provided in [FKMV12] for presentation. Namely, we model the ZK simulator of a NIZK proof system as a stateful PPT algorithm $\mathcal{S}$ that can operate in two modes: $(h, \mathsf{st}) \leftarrow \mathcal{S}(1, \mathsf{st}, q)$ takes care of answering random oracle queries, and $(\pi, \mathsf{st}) \leftarrow \mathcal{S}(2, \mathsf{st}, x)$ simulates the proof. Here, the calls to $\mathcal{S}(1, \cdots)$ and $\mathcal{S}(2, \cdots)$ share the common state st that is updated after each invocation of the simulator. Furthermore, we define three algorithms $\mathcal{S}_1, \mathcal{S}_2, \hat{\mathcal{S}}_2$ that run simulator $\mathcal{S}$ internally: $\mathcal{S}_1(q)$ returns the first output of $(h, \mathsf{st}) \leftarrow \mathcal{S}(1, \mathsf{st}, q)$, $\mathcal{S}_2(x, w)$ ignores the second input w and returns the first output of $(\pi, \mathsf{st}) \leftarrow \mathcal{S}(2, \mathsf{st}, x)$ if and only if $(x, w) \in \mathcal{R}$ (or equivalently $x \in \mathcal{L}$), and $\hat{\mathcal{S}}_2(x)$ is essentially the same as $\mathcal{S}_2(x, w)$ except that it does not take a second input w and is also defined for inputs such that $x \notin \mathcal{L}$. Observe that $\mathcal{S}_2$ and $\hat{\mathcal{S}}_2$ are identical for inputs $x \in \mathcal{L}$, and unlike $\mathcal{S}_2$, $\hat{\mathcal{S}}_2$ may be invoked to simulate proofs for invalid statements.

Definition 6 (Non-Interactive Zero-Knowledge Proof System). *Let $\mathcal{R}$ be a relation with an associated language $\mathcal{L}_\mathcal{R}$. We say a non-interactive proof system $(\mathcal{P}, \mathcal{V})$ is a statistical NIZK proof system for language $\mathcal{L}_\mathcal{R}$ with a (PPT) ZK simulator $\mathcal{S}$ in the random oracle model, if for any algorithm $\mathcal{D}$ we have*

$$\left|\Pr[\mathcal{D}^{H(\cdot), \mathcal{P}^H(\cdot,\cdot)}(1^\lambda) = 1] - \Pr[\mathcal{D}^{\mathcal{S}_1(\cdot), \mathcal{S}_2(\cdot,\cdot)}(1^\lambda) = 1]\right| = \mathsf{negl}(\lambda),$$

where $H(\cdot)$ is modeled as a random oracle, and both $\mathcal{P}$ and $\mathcal{S}_2$ output $\perp$ if $(x, w) \notin \mathcal{R}$. It is called a computational NIZK proof system in case the above holds only for all PPT algorithms $\mathcal{D}$.

It is a well known fact that in the random oracle model, the Fiat-Shamir transformation of any Σ-protocol is a NIZK proof system. It is straightforward to prove that it is also the case for gap-Σ-protocols, as we state in the following lemma.

Lemma 2 (Fiat-Shamir NIZK Proof Systems). *Let $(\mathcal{P}, \mathcal{V})$ be a gap-Σ-protocol with relation $(\mathcal{R}, \mathcal{R}')$ that is computationally (resp. statistically) special HVZK, and $H(\cdot)$ a hash function with range equal to the verifier's challenge space*

$\mathcal{C}$. Then, in the random oracle model, the non-interactive proof system $(\mathcal{P}^H, \mathcal{V}^H)$ obtained by the Fiat-Shamir transformation of gap-Σ is a computational (resp. statistical) non-interactive zero-knowledge proof system for the language $\mathcal{L}_\mathcal{R}$.

Proof (Proof sketch.). To prove that the proof system $(\mathcal{P}^H, \mathcal{V}^H)$ is a NIZK proof system for the language $\mathcal{L}_\mathcal{R}$, it suffices to show that there exists a ZK simulator $\mathcal{S}$ as in the above Def. 6. Below, we construct $\mathcal{S}$ by invoking the HVZK simulator $\mathcal{S}_\Sigma$ of the underlying gap-Σ-protocol $(\mathcal{P}, \mathcal{V})$:

- $\mathcal{S}(1, \mathsf{st}, q = (x, \alpha)) \to (h = \beta, \mathsf{st})$: To answer random oracle queries, it searches the table $\mathcal{T}_H$ kept in the state st whether an output for $q = (x, \alpha)$ is already defined. If so it returns the previously defined assigned value. If not, it samples a uniformly random value $\beta \leftarrow \mathcal{C}$ and stores $(q = (x, \alpha), h = \beta)$ in the table. Note that this corresponds to algorithm $\mathcal{S}_1$.
- $\mathcal{S}(2, \mathsf{st}, x) \to (\pi = (\alpha, \beta, \gamma), \mathsf{st})$: To simulate a proof for the statement $x \in \mathcal{L}_\mathcal{R}$, it runs the HVZK simulator $\mathcal{S}_\Sigma$ on input x to obtain a proof (α, β, γ). Then, it updates the table $\mathcal{T}_H$ by adding $(q = (x, \alpha), h = \beta)$. If $\mathcal{T}_H$ happens to be already defined on input $q = (x, \alpha)$, $\mathcal{S}$ aborts. This completely specifies algorithm $\mathcal{S}_2$ as required. Observe that the simulator $\mathcal{S}$ can also be run on statements $x \notin \mathcal{L}_\mathcal{R}$ using the above method, since $\mathcal{S}_\Sigma$ is well-defined for $x \in \mathcal{L}$ as well. In particular, the above description for $\mathcal{S}$ also specifies algorithm $\hat{\mathcal{S}}_2$ as well.

Since, we only consider gap-Σ-protocols with high-commitment entropy, the probability of simulator $\mathcal{S}$ aborting is negligible, which ends the proof sketch.

In the following, we use the above algorithm $\mathcal{S}$ as the ZK simulator for a NIZK proof system $(\mathcal{P}^H, \mathcal{V}^H)$ based on the Fiat-Shamir transformation of a gap-Σ-protocol $(\mathcal{P}, \mathcal{V})$. Note that we do not explicitly define the soundness property of the NIZK proof system, since this property will be implicitly implied when we construct a knowledge extractor during the security proof.

4 Generic Construction of Attribute-based Signatures

Preparation. Before presenting our construction, we describe the relations and languages we require for our NIZK proof system. Our construction relies on a gap-Σ-protocol for the relations $(\mathcal{R}_{\mathsf{ABS}}, \mathcal{R}'_{\mathsf{ABS}})$ defined below and employs the Fiat-Shamir transformation provided in Def. 5 to turn it in into a NIZK proof system. In the following, x_i for $i \in [\ell + 1, \ell + N - 1]$ denotes the values assigned to the i-th (internal) wire of C on input $\mathbf{x} = (x_1, \cdots, x_\ell)$ and $\mathsf{vk}_{\mathsf{Sign}}$, $\mathsf{pk}_{\mathsf{Com}}$ denotes the verification key and public commitment key of the underlying digital signature scheme and commitment scheme, respectively. Then the relation $\mathcal{R}_{\mathsf{ABS}}$ is defined as follows:

$$\mathcal{R}_{\mathsf{ABS}} = \Big\{ \Big(\mathsf{statement} = \big(\mathsf{vk}_{\mathsf{Sign}}, \mathsf{pk}_{\mathsf{Com}}, C \in \mathcal{C}_\ell, c_\sigma, (c_i)_{i=1}^{\ell+|C|-1}\big), \\ \mathsf{witness} = \big(\mathbf{x} = (x_1, \cdots, x_\ell), \sigma, d_\sigma, (d_i)_{i=1}^{\ell+|C|-1}\big)\Big) \Big|$$

the committed values in $c_\sigma, (c_i)_{i=1}^{\ell+|C|-1}$ satisfy the following conditions:

$$\left.\begin{array}{l}
\text{- } \mathsf{S.Verify}(\mathsf{vk}_{\mathsf{Sign}}, \mathbf{x}, \sigma) = 1 \\
\text{- } x_i = x_{i_1} \star_i x_{i_2} \text{ for } i \in [\ell+1, \ell+|C|-1] \text{ for } (\star_i, i_1, i_2) \leftarrow \mathsf{topo}_C(i) \\
\text{- } 1 = x_{(\ell+|C|)_1} \star_{\ell+|C|} x_{(\ell+|C|)_2} \text{ for } (\star_{\ell+|C|}, i_{(\ell+|C|)_1}, i_{(\ell+|C|)_2}) \leftarrow \mathsf{topo}_C(\ell+|C|) \\
\text{- } (c_\sigma, d_\sigma) \in \mathcal{D}_{\mathsf{Com}}(\mathsf{pk}_{\mathsf{Com}}, \sigma) \text{ and } (c_i, d_i) \in \mathcal{D}_{\mathsf{Com}}(\mathsf{pk}_{\mathsf{Com}}, x_i) \text{ for } i \in [\ell+|C|-1]
\end{array}\right\}$$

Here, recall that $\mathcal{D}_{\mathsf{Com}}(\mathsf{pk}_{\mathsf{Com}}, \mathsf{M})$ is the set of all possible outputs of the commitment algorithm $\mathsf{C.Com}(\mathsf{pk}_{\mathsf{Com}}, \mathsf{M})$. We simply define the corresponding language $\mathcal{L}_{\mathsf{ABS}}$ as the language induced by the relation $\mathcal{R}_{\mathsf{ABS}}$. Furthermore, the gap-relation $\mathcal{R}'_{\mathsf{ABS}}$ is defined analogously to $\mathcal{R}_{\mathsf{ABS}}$ except that we replace the last condition as follows:

$$\text{- } (c_\sigma, d_\sigma) \in \mathcal{D}_{\mathsf{G\text{-}Com}}(\mathsf{pk}_{\mathsf{Com}}, \sigma) \wedge (c_i, d_i) \in \mathcal{D}_{\mathsf{G\text{-}Com}}(\mathsf{pk}_{\mathsf{Com}}, x_i) \text{ for } i \in [\ell+|C|-1]$$

The only difference between the two relations are the condition on the commitment and opening pairs. In the latter, it is only required that the pairs are in the set $\mathcal{D}_{\mathsf{G\text{-}Com}}(\cdot)$ and not in the more restricted set $\mathcal{D}_{\mathsf{Com}}(\cdot)$. Recall that $\mathcal{D}_{\mathsf{G\text{-}Com}}(\mathsf{pk}_{\mathsf{Com}}, \mathsf{M})$ is the set of all commitment and opening pairs that the opening algorithm outputs 1 on message M. As we noted in Sec. 3.1, we require this gap-relation $\mathcal{R}'_{\mathsf{ABS}}$ purely for technical reasons, since in many of the lattice-based Σ-protocols we can only extract witnesses that lie in a slightly larger space than the actual witnesses being proven in the actual protocol. Similarly to above, we define the language $\mathcal{L}'_{\mathsf{ABS}}$ as the language induced by the relation $\mathcal{R}'_{\mathsf{ABS}}$.

For simplicity, in the following we omit $\mathsf{vk}_{\mathsf{Sign}}$ and $\mathsf{pk}_{\mathsf{Com}}$ from the statement, since they are fixed by the Setup algorithm and all signers use the same $\mathsf{vk}_{\mathsf{Sign}}$ and $\mathsf{pk}_{\mathsf{Com}}$.

Construction. Here, we provide our attribute-based signature scheme for unbounded (arithmetic) circuits. In the following, assume a digital signature scheme $(\mathsf{S.KeyGen}, \mathsf{S.Sign}, \mathsf{S.Verify})$, a commitment scheme $(\mathsf{C.Gen}, \mathsf{C.Com}, \mathsf{C.Open})$ and a NIZK proof system for the relation $\mathcal{R}_{\mathsf{ABS}}$.

$\mathsf{Setup}(1^\lambda, 1^\ell)$: On input the security parameter 1^λ and the input length 1^ℓ for the family of circuits $\mathcal{C}_\ell$, generate a verification key and a signing key $(\mathsf{vk}_{\mathsf{Sign}}, \mathsf{sk}_{\mathsf{Sign}}) \leftarrow \mathsf{S.KeyGen}(1^\lambda, 1^\ell)$ and a public commitment key $\mathsf{pk}_{\mathsf{Com}} \leftarrow \mathsf{C.Gen}(1^\lambda)$. Then output

$$\mathsf{mpk} = (\mathsf{vk}_{\mathsf{Sign}}, \mathsf{pk}_{\mathsf{Com}}, H(\cdot), G(\cdot)) \quad \text{and} \quad \mathsf{msk} = (\mathsf{sk}_{\mathsf{Sign}}).$$

Here, $H(\cdot)$ and $G(\cdot)$ are hash functions used by the NIZK proof system and by algorithm Sign, respectively, which are programmed as random oracles in the security reduction. Further, we assume the output space of $G(\cdot)$ to be $\{0,1\}^\ell$.[6]

[6] Here, we do not explicitly define the input and output space of the hash functions, since it may differ according to the underlying NIZK proof system being used.

$\mathsf{KeyGen}(\mathsf{mpk}, \mathsf{msk}, \mathbf{x})$: On input $\mathbf{x} = (x_1 \cdots, x_\ell) \in \{0,1\}^\ell$, create a signature on the attribute $\mathbf{x} \in \{0,1\}^\ell$ by running $\sigma \leftarrow \mathsf{S.Sign}(\mathsf{sk}_{\mathsf{Sign}}, \mathbf{x})$. Then, output the secret key as $\mathsf{sk}_{\mathbf{x}} = (\mathbf{x}, \sigma)$.

$\mathsf{Sign}(\mathsf{mpk}, \mathsf{sk}_{\mathbf{x}}, C, \mathsf{M})$: On input message $\mathsf{M} \in \{0,1\}^\star$ and circuit $C \in \mathcal{C}_\ell$ with an associating topology topo_C proceed as follows:

1. Compute $\mathbf{h} = (h_1, \cdots, h_\ell) \leftarrow G(\mathsf{M}, C)$[7] and create a new circuit $\hat{C} \in \mathcal{C}_\ell$ with two dummy gates connected to each of the input wires of C. Namely, to the input wires $i \in [\ell]$ of C, we add a series composition of two addition gates where one gate adds h_i and the other gate adds $-h_i$; on input x_i to the i-th input wire of $\hat{C}$, it first evaluates to $x_i + h_i$ and then evaluates back to x_i, on which point it gets fed to the i-th (input) wire of C. Here, the value $\mathbf{h}$ is hard-wired into $\hat{C}$, and is considered as one of the internal wires. Further, let N be the number of gates $|\hat{C}|$.
2. Compute the assignment to each non-input wires in $\hat{C}(x_1, \cdots, x_\ell)$: for all $i \in [\ell+1, \ell+(N-1)]$, compute $(\star_i, i_1, i_2) \leftarrow \mathsf{topo}(i)$ where $\star_i \in \{+, \times\}$, and denote the newly created values $(x_i)_{i=\ell+1}^{\ell+N-1}$ in ascending order as
$$\begin{cases} x_i = x_{i_1} + x_{i_2} & \text{if } \star_i = + \\ x_i = x_{i_1} \cdot x_{i_2} & \text{if } \star_i = \times \end{cases}.$$
3. Create a commitment $(c_\sigma, d_\sigma) \leftarrow \mathsf{C.Com}(\mathsf{pk}_{\mathsf{Com}}, \sigma)$ of the signature σ. Furthermore, for all $i \in [\ell + N - 1]$, create a commitment $(c_i, d_i) \leftarrow \mathsf{C.Com}(\mathsf{pk}_{\mathsf{Com}}, x_i)$ that commits to the value of each wire in $\hat{C}$ (except for the output wire).
4. Generate a NIZK proof π proving that the committed values satisfy relation $\mathcal{R}_{\mathsf{ABS}}$. Concretely, it generates a proof for the following conditions.[8]
   - The attribute $\mathbf{x} = (x_1, \cdots, x_\ell)$ committed to $(c_i)_{i=1}^\ell$ and the signature σ committed to c_σ satisfy the following verification equation:
   $$\mathsf{S.Verify}(\mathsf{vk}_{\mathsf{Sign}}, \mathbf{x}, \sigma) = 1. \tag{5}$$
   - For all $i \in [\ell+1, \ell+N-1]$, the value x_i committed to c_i satisfy the following equation:
   $$\begin{cases} x_i = x_{i_1} + x_{i_2} & \text{if } \star_i = + \\ x_i = x_{i_1} \cdot x_{i_2} & \text{if } \star_i = \times \end{cases}. \tag{6}$$
   - The values $x_{(\ell+N)_1}$ and $x_{(\ell+N)_2}$ committed to $c_{(\ell+N)_1}$ and $c_{(\ell+N)_2}$, respectively, satisfy the following equation:
   $$\begin{cases} 1 = x_{(\ell+N)_1} + x_{(\ell+N)_2} & \text{if } \star_{\ell+N} = + \\ 1 = x_{(\ell+N)_1} \cdot x_{(\ell+N)_2} & \text{if } \star_{\ell+N} = \times \end{cases}. \tag{7}$$

[7] Here, we assume that we can encode C uniquely into a binary string.

[8] Note that we intentionally dismiss the conditions $(c, d) \in \mathcal{D}_{\mathsf{Com}}(\mathsf{pk}_{\mathsf{Com}}, \star)$ as in the overview, i.e., proving knowledge of a valid opening, since they will be implicitly proven by the fact that the committed messages satisfy Eq. (5 - 7).

5. Finally, output $\Sigma = \left(c_\sigma, (c_i)_{i=1}^{\ell+N-1}, \pi\right)$.

$\mathsf{Verify}(\mathsf{mpk}, \mathsf{M}, C, \Sigma)$: Compute $\mathbf{h} \leftarrow G(\mathsf{M}, C)$ and construct the circuit $\hat{C}$ as in Step 1 of the Sign algorithm. Then, verify the proof with respect to the circuit $\hat{C}$. Output Valid if the proof is verified valid, and output $\mathsf{Invalid}$ otherwise.

Correctness. Observe that $\hat{C}(\mathbf{x}) = C(\mathbf{x})$ for all $\mathsf{M}, \mathbf{x}$. Therefore, the correctness of the scheme follows simply from the correctness of the underlying NIZK proof system. In particular, a signer that has a certified attribute $\mathbf{x}$ such that $C(\mathbf{x}) = 1$ can properly generate a proof proving Eq. (5 - 7).

4.1 Security Analysis

Theorem 1 (Privacy). *Assume a statistically hiding commitment scheme with gap-openings and a statistically special HVZK gap-Σ-protocol for relations* ($\mathcal{R}_{\mathsf{ABS}}$, $\mathcal{R}'_{\mathsf{ABS}}$). *Then, converting the gap-Σ-protocol into a Fiat-Shamir NIZK proof system, the above attribute-based signature scheme is statistically private in the random oracle model. In case either the hiding property or the special HVZK property only holds computationally, then we obtain computational privacy.*

The proof follows naturally from the zero-knowledge property of the NIZK proof system, and is deferred the full version.

Theorem 2 (Adaptive Unforgeability). *Assume a computationally hiding and a statistically binding commitment scheme with gap openings, a computationally special HVZK gap-Σ_m-protocol*[9] *for relations* ($\mathcal{R}_{\mathsf{ABS}}, \mathcal{R}'_{\mathsf{ABS}}$) *and an* $\mathsf{eu\text{-}cma}$ *secure (deterministic) digital signature scheme. Then, by converting the gap-Σ_m-protocol into a Fiat-Shamir NIZK proof system, the above attribute-based signature scheme is adaptively unforgeable in the random oracle model.*

Proof. Assume there exists a PPT adversary $\mathcal{B}_{\mathsf{ABS}}$ that wins the adaptive unforgeability game with advantage $\epsilon = \epsilon(\lambda)$. Furthermore, let $Q_H = Q_H(\lambda)$ be the number of unique random oracle queries $\mathcal{B}_{\mathsf{ABS}}$ makes to $H(\cdot)$ that is bounded by some polynomial in the security parameter λ. Our proof proceeds in a sequence of games, where X_i denotes the event the adversary wins in Game_i. Our final goal is to construct an adversary $\mathcal{B}_{\mathsf{Sign}}$ that breaks the $\mathsf{eu\text{-}cma}$ security of the underlying digital signature scheme by using $\mathcal{B}_{\mathsf{ABS}}$. For our Fiat-Shamir NIZK proof system, we use the ZK simulator $\mathcal{S}$ that we have defined in Lem. 2.

$\mathsf{Game}_{\mathsf{real}}$: This game is identical to the real adaptive unforgeability game where all the random oracle queries to $H(\cdot)$ and $G(\cdot)$ are answered randomly by the challenger. At the end of the game, $\mathcal{B}_{\mathsf{ABS}}$ outputs a valid forged signature $(\mathsf{M}^*, C^*, \Sigma^*)$ with probability $\Pr[X_{\mathsf{real}}] = \epsilon$.

[9] Here, recall that we write gap-Σ_m-protocol, when we make explicit of the fact that m valid transcripts are requried for special gap-soundness to hold. Furthermore, this notation also implies that the soundness error is negligible (See Sec. 3.1).

Game_1 : In this game, we change the way the challenger answers the random oracle queries to $H(\cdot)$ and the signing queries. Namely, we use the ZK simulator $\mathcal{S}$ associated to the NIZK proof system to answer these. Recall that simulator $\mathcal{S}$ has two modes for running the two oracles $\mathcal{S}_1$ and $\hat{\mathcal{S}}_2$. When $\mathcal{B}_{\mathsf{ABS}}$ submits a random oracle query to $H(\cdot)$, the challenger relays this to oracle $\mathcal{S}_1$ and returns the value outputted by $\mathcal{S}_1$ to $\mathcal{B}_{\mathsf{ABS}}$. Here, the random oracle queries to $G(\cdot)$ are answered by the Game_1 challenger as in the previous game. Furthermore, when $\mathcal{B}_{\mathsf{ABS}}$ submits a signing query on an attribute, message and circuit tuple $(\mathbf{x}, \mathsf{M}, C)$ such that $C(\mathbf{x}) = 1$, it first runs $\mathsf{sk}_{\mathbf{x}} = (\mathbf{x}, \sigma) \leftarrow \mathsf{KeyGen}(\mathsf{mpk}, \mathsf{msk}, \mathbf{x})$ and constructs the circuit $\hat{C}$ with N gates using $\mathbf{h} \leftarrow G(\mathsf{M}, C)$ as in Step 1 of the Sign algorithm. Then it proceeds with Step 2 and 3 to create commitments $\left(c_\sigma, (c_i)_{i=1}^{\ell+N-1}\right)$ along with valid openings $\left(d_\sigma, (d_i)_{i=1}^{\ell+N-1}\right)$. Finally, it invokes $\hat{\mathcal{S}}_2$ on input the statement $\left(\hat{C}, c_\sigma, (c_i)_{i=1}^{\ell+N-1}\right) \in \mathcal{L}_{\mathsf{ABS}}$[10] and obtains a proof π, and returns the signature $\Sigma = \left(c_\sigma, (c_i)_{i=1}^{\ell+N-1}, \pi\right)$ to $\mathcal{B}_{\mathsf{ABS}}$. Here, the simulated proofs of $\hat{\mathcal{S}}_2$ are distributed negligibly close to the actual proofs in $\mathsf{Game}_{\mathsf{real}}$ by the definition of the NIZK proof system (See Def. 6), and the fact that the oracles $\mathcal{S}_2$ and $\hat{\mathcal{S}}_2$ are equivalent in case the statement to be proven is in the language. Hence, $|\Pr[X_{\mathsf{real}}] - \Pr[X_1]| = \mathsf{negl}(\lambda)$.

Game_2 : In this game, we change the way the challenger creates the commitment for the signature σ produced during the signing query. In the previous game, when $\mathcal{B}_{\mathsf{ABS}}$ submitted a signing query on an attribute, message and circuit tuple $(\mathbf{x}, \mathsf{M}, C)$ such that $C(\mathbf{x}) = 1$, the challenger created a proper commitment c_σ for the signature σ following Step 3 of the Sign algorithm, i.e., $(c_\sigma, d_\sigma) \leftarrow \mathsf{Com}(\mathsf{pk}_{\mathsf{Com}}, \sigma)$. In this game, however, the Game_2 challenger will instead sample a random value c in the commitment space $\mathcal{C}_{\mathsf{Com}}$ and sets $c_\sigma = c$. Then, as in Game_2, it invokes $\hat{\mathcal{S}}_2$ on input $\left(\hat{C}, c_\sigma, (c_i)_{i=1}^{\ell+N-1}\right)$ and obtains a proof π, and returns the signature $\Sigma = \left(c_\sigma, (c_i)_{i=1}^{\ell+N-1}, \pi\right)$ to $\mathcal{B}_{\mathsf{ABS}}$. Here, recall that oracle $\hat{\mathcal{S}}_2$ is defined to simulate proofs for false statements that are not in the language $\mathcal{L}_{\mathsf{ABS}}$ as well. Then, the differences in the view of the adversary in Game_1 and Game_2 are computationally indistinguishable due to the computationally hiding property of the commitment scheme.[11] In other words, we have $|\Pr[X_1] - \Pr[X_2]| = \mathsf{negl}(\lambda)$.

Game_3 : In this game, we add an additional winning condition for adversary $\mathcal{B}_{\mathsf{ABS}}$ to satisfy. Namely, when $\mathcal{B}_{\mathsf{ABS}}$ outputs a forgery $(\mathsf{M}^*, C^*, \Sigma^*)$, the Game_3 challenger checks if the random oracle $G(\cdot)$ was ever queried on a message-circuit pair $(\mathsf{M}, C) \neq (\mathsf{M}^*, C^*)$ such that $\hat{C} = \hat{C}^*$. Note that this implies $G(\mathsf{M}, C) = G(\mathsf{M}^*, C^*)$. Hereafter, we say $\mathcal{B}_{\mathsf{ABS}}$ wins if and only if

[10] Recall we ignore the public parameters $\mathsf{vk}_{\mathsf{Sign}}$ and $\mathsf{pk}_{\mathsf{Com}}$ from the statement for simplicity.

[11] More formally, we create q_{sign} hybrid games and swap the commitments of the signature to a random value in the commitment space one hybrid game at a time until we have swapped every signature commitments into the desired random form, where q_{sign} is the number of signature queries $\mathcal{B}_{\mathsf{ABS}}$ makes.

in addition to the winning condition of the previous game, there are no such message-circuit pairs. Since, the output values of the random oracle $G(\cdot)$ are uniformly random over $\{0,1\}^\ell$ for $\ell = \mathsf{poly}(n)$, the probability that a collision occurs for different message-circuit pairs is negligible. Hence, $|\Pr[X_2] - \Pr[X_3]| = \mathsf{negl}(\lambda)$. Below, we denote $\epsilon_3 = \Pr[X_3]$.

In the following, we define the algorithms $\mathcal{A}$ and $\mathcal{O}$ to be used in the forking algorithm $\mathsf{F}_{\mathcal{A},m}^{\mathcal{O}(\overline{\mathsf{par}},\cdot)}$ of the generfal multi-forking lemma with oracle access (See Lem. 1). Looking ahead, the forking algorithm will be used by adversary $\mathcal{B}_{\mathsf{Sign}}$ to win the $\mathsf{eu\text{-}cma}$ security of the underlying digital signature scheme.

To provide the full description of algorithms $\mathcal{A}$ and $\mathcal{O}$, we first define the input generator IG, the set $\mathcal{H}$ and the integer q, which are required to define the inputs for $\mathcal{A}$ and $\mathcal{O}$. First, the input generator IG outputs $(\mathsf{par}, \overline{\mathsf{par}})$ where par constitutes of the verification key $\mathsf{vk}_{\mathsf{Sign}}$, public commitment key $\mathsf{pk}_{\mathsf{Com}}$ and any extra auxiliary parameters required to specify the ABS scheme (e.g., the family of circuits), and $\overline{\mathsf{par}}$ is simply the signing key $\mathsf{sk}_{\mathsf{Sign}}$. Here, $\mathsf{vk}_{\mathsf{Sign}}, \mathsf{sk}_{\mathsf{Sign}}$ and $\mathsf{pk}_{\mathsf{Com}}$ are generated by running $(\mathsf{vk}_{\mathsf{Sign}}, \mathsf{sk}_{\mathsf{Sign}}) \leftarrow \mathsf{S.KeyGen}(1^\lambda, 1^\ell)$ and $\mathsf{pk}_{\mathsf{Com}} \leftarrow \mathsf{C.Gen}(1^\lambda)$. Furthermore, we define the set $\mathcal{H}$ to be the verifier's challenge space $\mathcal{C}_\Sigma$ of the underlying gap-Σ_m-protocol, and set q as Q_H; the number of unique random oracle queries made to $H(\cdot)$ by $\mathcal{B}_{\mathsf{ABS}}$. To summarize, $\mathcal{A}$ will be given par and $h_1, \cdots, h_{Q_H} \in \mathcal{H}$ as input.

We next specify how algorithms $\mathcal{A}$ and $\mathcal{O}$ run. First, the deterministic algorithm $\mathcal{O}$ is simply defined as the signing algorithm of the underlying deterministic digital signature scheme; $\mathcal{O}(\overline{\mathsf{par}}, \cdot) = \mathsf{S.Sign}(\mathsf{sk}_{\mathsf{Sign}}, \cdot)$. Here, $\mathcal{O}$ is deterministic since the signing algorithm is deterministic once fixed a signing key $\mathsf{sk}_{\mathsf{Sign}}$. Next, we define $\mathcal{A}$ as the randomized algorithm that simulates Game_3 and outputs a small modification of the forgery returned by $\mathcal{B}_{\mathsf{ABS}}$. We first explain how $\mathcal{A}$ simulates Game_3: $\mathcal{A}$ essentially runs the Game_3 challenger, $\mathcal{B}_{\mathsf{ABS}}$ and the ZK simulator $\mathcal{S}$ internally, with two conceptual changes concerning the Game_3 challenger and the ZK simulator $\mathcal{S}$. In particular the Game_3 challenger is modified to an algorithm which we call the Game_3' challenger, so that it does not run $(\mathsf{vk}_{\mathsf{Sign}}, \mathsf{sk}_{\mathsf{Sign}}) \leftarrow \mathsf{S.KeyGen}(1^\lambda, 1^\ell)$ anymore. Instead of generating $(\mathsf{vk}_{\mathsf{Sign}}, \mathsf{sk}_{\mathsf{Sign}})$ on its own, the Game_3' challenger is provided with $\mathsf{vk}_{\mathsf{Sign}}$ by $\mathcal{A}$, and no longer possesses $\mathsf{sk}_{\mathsf{Sign}}$. Whenever the Game_3' challenger requires to run the signing algorithm $\mathsf{S.Sign}(\mathsf{sk}_{\mathsf{Sign}}, \cdot)$, $\mathcal{A}$ simply invokes $\mathcal{O}(\overline{\mathsf{par}}, \cdot) = \mathsf{S.Sign}(\mathsf{sk}_{\mathsf{Sign}}, \cdot)$, which it has oracle access to, and returns whatever outputtd by $\mathcal{O}$ to the Game_3' challenger. Furthermore, the ZK simulator $\mathcal{S}$ (See Lem. 2) is modified in a way so that it does not sample a random value $h_i \leftarrow \mathcal{C}_\Sigma$ when invoked on a random oracle query to $H(\cdot)$. Concretely, on the i-th unique random oracle query to $H(\cdot)$, it simply outputs the value h_i provided by $\mathcal{A}$.[12] This is only a conceptual change, since $\mathcal{C}_\Sigma = \mathcal{H}$ and h_i are sampled uniformly over $\mathcal{H}$. Therefore, the above changes do not alter the view of $\mathcal{B}_{\mathsf{ABS}}$. Hence the advantage of $\mathcal{B}_{\mathsf{ABS}}$ winning the game

[12] More formally, we can think the state st provided to the ZK simulator $\mathcal{S}$ includes $(h_i)_{i=1}^{Q_H}$, assuming without loss of generality that $\mathcal{S}$ knows the bound on the number of query made by $\mathcal{B}_{\mathsf{ABS}}$.

simulated by $\mathcal{A}$ is exactly the same as of Game_3. Finally, we describe the output of $\mathcal{A}$. In particular, at the end of the simulation of Game_3, $\mathcal{B}_{\mathsf{ABS}}$ outputs a valid forgery $(\mathsf{M}^*, C^*, \Sigma^*)$ where $\Sigma^* = \left(c_\sigma^*, (c_i^*)_{i=1}^{\ell+N-1}, \pi^*\right)$ with probability ϵ_3. In the following let χ^* denote the statement $(\hat{C}^*, c_\sigma^*, (c_i^*)_{i=1}^{\ell+N-1})$, where $\hat{C}^*$ is the circuit with N gates constructed from C^* in Step 1 of the Sign algorithm. Since this is a valid forgery, we must have $\chi^* \in \mathcal{L}_{\mathsf{ABS}}$. Given the forgery of $\mathcal{B}_{\mathsf{ABS}}$, $\mathcal{A}$ first parses the proof π^* as (α^*, γ^*), where α^*, γ^* are the commitment and response of the underlying gap-Σ_m-protocol (See Def. 5), respectively. $\mathcal{A}$ then checks whether $H(\cdot)$ was queried on (χ^*, α^*). If not it outputs $(0, \epsilon_1)$. Otherwise, there exists an index $i^* \in [Q_H]$ for which the challenge $H(\chi^*, \alpha^*)$ is set to h_{i^*}. In this case, it outputs $(i^*, (\alpha^*, h_{i^*}, \gamma^*, \chi^*, \mathsf{M}^*, C^*))$. Now, since $\mathcal{A}$ simulates Game_3 perfectly and the probability of $\mathcal{B}_{\mathsf{ABS}}$ outputting a valid forgery without knowledge of the output of $H(\chi^*, \alpha^*)$ (i.e., the challenge) is negligible, we have

$$\begin{aligned}\mathsf{acc} &= \Pr\left[(i^*, (\alpha^*, h_{i^*}, \gamma^*, \chi^*, \mathsf{M}^*, C^*)) \leftarrow \mathcal{A}^{\mathcal{O}(\overline{\mathsf{par}}, \cdot)}(\mathsf{par}, h_1, \cdots, h_{Q_H}) \ : \ i^* \geq 1\right] \\ &\geq \epsilon_3 - \mathsf{negl}(\lambda), \end{aligned} \tag{8}$$

where the probability is taken over the choice of $(\mathsf{par}, \overline{\mathsf{par}}), (h_i)_{i=1}^{Q_H}$ and the randomness used by $\mathcal{A}$.

Finally we construct an adversary $\mathcal{B}_{\mathsf{Sign}}$ against the $\mathsf{eu\text{-}cma}$ security of the underlying digital signature scheme using the forking algorithm $\mathsf{F}_{\mathcal{A},m}^{\mathcal{O}(\overline{\mathsf{par}}, \cdot)}$. In particular the advantage of $\mathcal{B}_{\mathsf{Sign}}$ will be $\epsilon_3^m / Q_H^{m-1} - \mathsf{negl}(\lambda)$ for a constant m. Hence, assuming the $\mathsf{eu\text{-}cma}$ security of the digital signature scheme, ϵ_3 is negligible. Therefore, since $\epsilon = \epsilon_3 \pm \mathsf{negl}(\lambda)$, we conclude that ϵ is negligible, thus completing the proof. Below, let $\mathcal{C}_{\mathsf{Sign}}$ be the challenger for the $\mathsf{eu\text{-}cma}$ game of the underlying digital signature scheme. Also, let $\mathsf{vk}_{\mathsf{Sign}}$ be the verification key given to $\mathcal{B}_{\mathsf{Sign}}$ and $\mathsf{sk}_{\mathsf{Sign}}$ be the signing key used by $\mathcal{C}_{\mathsf{Sign}}$ to answer the signature queries. In particular, $\mathcal{C}_{\mathsf{Sign}}$ uses the signing algorithm $\mathsf{S.Sign}(\mathsf{sk}_{\mathsf{Sign}}, \cdot)$ to answer signature queries made be $\mathcal{B}_{\mathsf{ABS}}$. Now, given $\mathsf{vk}_{\mathsf{Sign}}$, $\mathcal{B}_{\mathsf{Sign}}$ runs $\mathsf{pk}_{\mathsf{Com}} \leftarrow \mathsf{C.Gen}(1^\lambda)$ and prepares par, i.e., the input to $\mathcal{A}$ provided by the input generator IG. This can be done efficiently since par constitutes only of public values: $\mathsf{vk}_{\mathsf{Sign}}, \mathsf{pk}_{\mathsf{Com}}$ and some other public auxiliary parameters specifying the ABS scheme. Since the forking algorithm only requires oracle access to the deterministic algorithm $\mathcal{O}(\overline{\mathsf{par}}, \cdot) = \mathsf{S.Sign}(\mathsf{sk}_{\mathsf{Sign}}, \cdot)$, which is provided by $\mathcal{C}_{\mathsf{Sign}}$, $\mathcal{B}_{\mathsf{Sign}}$ can properly run the forking algorithm $\mathsf{F}_{\mathcal{A},m}^{\mathcal{O}(\overline{\mathsf{par}}, \cdot)}(\mathsf{par})$ as specified. Note that $\mathsf{par}, \overline{\mathsf{par}}$ are distributed exactly as the output of the input generator IG defined above. Now, due to the general multi-forking lemma with oracle access (Lem. 1), we obtain the following pairs with probability frk:

$$\left(1, \quad \left\{\alpha^{(k)}, h^{(k)}, \gamma^{(k)}, \chi^{(k)}, \mathsf{M}^{(k)}, C^{(k)}\right\}_{k \in [m]}\right), \tag{9}$$

where $\chi^{(k)} = \left(\hat{C}^{(k)}, c_\sigma^{(k)}, (c_i^{(k)})_{i=1}^{\ell+N-1}\right)_{k \in [m]}$. Here, by Eq. (1) of Lem. 1, we have

$$\mathsf{frk} \geq \mathsf{acc} \cdot \left(\left(\frac{\mathsf{acc}}{Q_H}\right)^{m-1} - \frac{f(m)}{|\mathcal{C}_\Sigma|}\right) = \frac{\mathsf{acc}^m}{Q_H^{m-1}} - \mathsf{negl}(\lambda), \tag{10}$$

where $\mathcal{C}_\Sigma$ is the output range of $H(\cdot)$ that is super-polynomially large, m is a constant representing the number of valid transcripts we require to extract a witness and $f(m)$ is a universal positive valued function that only depends on m, i.e., a constant value when viewed as a funtion on the security parameter λ. Now, we argue that for all $k \in [m]$, the values of the commitments $\alpha^{(k)}$ and statements $\chi^{(k)}$ are equivalent, respectively. Let $i^* \in [Q_H]$ be the index outputted by $\mathcal{A}$ in the first run inside the forking algorithm $\mathsf{F}_{\mathcal{A},m}^{\mathcal{O}(\overline{\mathsf{par}},\cdot)}(\mathsf{par})$. Then, up until the i^*-th unique random oracle query to $H(\cdot)$, the behavior of $\mathcal{B}_{\mathsf{ABS}}$ is the same for every run, since we fix the randomness being used by the challenger Game'_3, $\mathcal{B}_{\mathsf{ABS}}$ and the ZK simulator $\mathcal{S}$. This implies that whatever submitted by $\mathcal{B}_{\mathsf{ABS}}$ on the i^*-th unique random oracle query to $H(\cdot)$, which is the pair $(\alpha^{(k)}, \chi^{(k)})$, must be the same in every run. Let us denote this as $(\alpha^*, \chi^* = (\hat{C}^*, c^*_\sigma, (c^*_i)_{i=1}^{\ell+N-1}))$. Therefore, by running $\mathsf{F}_{\mathcal{A},m}^{\mathcal{O}(\overline{\mathsf{par}},\cdot)}(\mathsf{par})$, $\mathcal{B}_{\mathsf{Sign}}$ obtains m valid transcript of the form $\left(\alpha^*, h^{(k)}, \gamma^{(k)}, \chi^*, \mathsf{M}^*, C^*\right)_{k\in[m]}$ where M^*, C^* are the same in every run as well, due to the winning condition we added in Game_3 and the fact that $\hat{C}^*$ is the same in every run.

Next, we show that $\mathcal{B}_{\mathsf{Sign}}$ can properly extract a witness from the valid transcripts using the knowledge extractor of the underlying gap-Σ_m-protocol (See special gap-soundness of Def. 4). Recall that the range of the random oracle $H(\cdot)$ is $\mathcal{C}_\Sigma = \{0, 1, \cdots, m-1\}^t$ for some constant m and an integer-valued function t that is poly-logarithmic in the security parameter λ. Now, by Def. 4, in order to extract a witness there needs to exist at least one index $j \in [t]$ such that $\{h_j^{(k)}\}_{k\in[m]} = \{0, 1, \cdots, m-1\}$. Since each $h^{(k)}$ are sampled uniformly random over $\mathcal{C}_H = \{0, 1, \cdots, m-1\}^t$, the probability of no such $j \in [t]$ existing is $(1 - \frac{m!}{m^m})^t$, which is negligible in the security parameter for our choices of m, t. Therefore, with all but negligible probability, $\mathcal{B}_{\mathsf{Sign}}$ is able to extract a witness $(\mathbf{x}^*, \sigma^*, d^*_\sigma, (d^*_i)_{i=1}^{\ell+N-1})$ in the gap-language $\mathcal{L}'_{\mathsf{ABS}}$ from the m valid transcripts. Furthermore, since we use a statistically binding commitment scheme, the $(\mathbf{x}^*, \sigma^*)$ pair extracted from the transcripts are the actual pairs used by $\mathcal{B}_{\mathsf{ABS}}$ to create a forgery, with all but negligible probability.

Finally, we show that $(\mathbf{x}^*, \sigma^*)$ is a valid signature forgery that allows $\mathcal{B}_{\mathsf{Sign}}$ to win the $\mathsf{eu\text{-}cma}$ game between the challenger $\mathcal{C}_{\mathsf{Sign}}$. Namely, we show that $\mathbf{x}^*$ was never queried as the key reveal query by $\mathcal{B}_{\mathsf{ABS}}$ in all of the m runs of $\mathcal{A}$. Note that the only situation $\mathcal{A}$ invokes the signing oracle $\mathcal{O}(\overline{\mathsf{par}}, \cdot) = \mathsf{S.Sign}(\mathsf{sk}_{\mathsf{Sign}}, \cdot)$ is when $\mathcal{B}_{\mathsf{ABS}}$ submits a key reveal query to the Game'_3 challenger. This is because we altered the game in Game_2 so that the ZK simulator is used to answer the signing queries made by $\mathcal{B}_{\mathsf{ABS}}$. Now, since $\mathcal{B}_{\mathsf{ABS}}$ outputs a valid forgery we have $\hat{C}^*(\mathbf{x}^*) = 1$. Then, by the way we construct $\hat{C}^*(\mathbf{x}^*)$ in Step 1 of the Sign algorithm, we have $C(\mathbf{x}^*) = 1$ as well. On the other hand, due to the winning condition of $\mathcal{B}_{\mathsf{ABS}}$, $\mathcal{B}_{\mathsf{ABS}}$ must have never made a key reveal query on $\mathbf{x}^*$ such that $C^*(\mathbf{x}^*) = 1$ (in any of the runs). Therefore, we conclude that $\mathbf{x}^*$ was never queried to the Game'_3 challenger by $\mathcal{B}_{\mathsf{ABS}}$ in any of the runs of $\mathcal{A}$; $(\mathbf{x}^*, \sigma^*)$ is a valid forgery. Hence, combining Eq. (8) and (10), the advantage of $\mathcal{B}_{\mathsf{Sign}}$ is $\epsilon_3^m / Q_H^{m-1} - \mathsf{negl}(\lambda)$.

4.2 Implications

Since a computationally hiding and statistically binding commitment scheme, a deterministic digital signature scheme and a computationally special HVZK Σ-protocols for any NP-language are all implied from one-way functions (See for example [Nao91,Rom90,PSV06]), we obtain the following lemma as an implication of our above result:

Lemma 3. *If one-way functions exist, then there exist computationally private and adaptive unforgeable attribute-based signature schemes for unbounded circuits in the random oracle model.*

5 ABS for Unbounded Circuits from Lattices

In this section, we provide an efficient instantiation of our generic ABS construction for unbounded circuits from lattices. In particular, we prepare a lattice-based signature scheme and a commitment scheme with gap-openings, and construct an associating lattice-based gap-Σ-protocol for the relation $\mathcal{R}_{\mathsf{ABS}}$. We believe our gap-$\Sigma$-protocol for proving possession of a valid signature, which departs from the previously known stern-type protocol of [LNSW13], to have applications in other contexts such as group signatures.

5.1 Preparing Tools

We present the underlying lattice-based digital signature scheme and commitment scheme with gap-openings that we use as building blocks for our lattice-based ABS scheme.

Digital Signature Scheme. Here, we review the lattice-based digital signature scheme of Boyen [Boy10] with an improved security reduction by [MP12]. Below, we provide a deterministic version of Boyen's signature scheme, where the signing algorithm uses a PRF for generating the required randomness. We defer the formal definition of lattices and PRFs to the full version. In the following, by lattice convention, we use the dimension of the lattice n to denote the security parameter.

Theorem 3. *Let n, m, q be positive integers such that $m \geq 2n \log q$. Let α, β be positive reals such that $\alpha = \Omega(\sqrt{\ell n \log q} \log n)$ and $\beta = \alpha\omega(\sqrt{\log m})$. Then, the following algorithms $(\mathsf{S.KeyGen}, \mathsf{S.Sign}, \mathsf{S.Verify})$ form a deterministic digital signature scheme with message space $\mathcal{M} = \{0,1\}^\ell$ that is* eu-cma *secure under hardness of the $\mathsf{SIS}^\infty_{n,m,q,\ell\tilde{O}(n)}$ problem.*

$\mathsf{S.KeyGen}(1^n, 1^\ell)$: *It samples a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ with a trapdoor $\mathbf{T_A} \in \mathbb{Z}^{m \times m}$ using algorithm $\mathsf{TrapGen}(1^n, 1^m, q)$. It also samples matrices $\mathbf{A}_i \leftarrow \mathbb{Z}_q^{n \times m}$ for $i \in [0, \ell]$, a vector $\mathbf{u} \in \mathbb{Z}_q^n$ and generates a seed for a PRF by running $r \leftarrow \mathsf{PRF.Gen}(1^n)$. Finally it outputs the verification key vk and signing key sk as*

$$\mathsf{vk} = (\mathbf{A}, \mathbf{A}_0, \cdots, \mathbf{A}_\ell, \mathbf{u}), \quad \mathsf{sk} = (\mathbf{T_A}, r).$$

$\mathsf{S.Sign}(\mathsf{sk}, \mathbf{x})$: *On input the message* $\mathbf{x} \in \{0,1\}^\ell$*, it first constructs the matrix* $\mathbf{A}_\mathbf{x} = \mathbf{A}_0 + \sum_{i=1}^{\ell} x_i \mathbf{A}_i \in \mathbb{Z}_q^{n \times m}$*, where* x_i *is the* i*-th bit of* $\mathbf{x}$*. Then using* $\mathbf{T}_\mathbf{A}$*, it samples a short vector* $\mathbf{z} \in \mathbb{Z}^{2m}$ *such that* $[\mathbf{A}|\mathbf{A}_\mathbf{x}]\mathbf{z} = \mathbf{u} \mod q$ *using algorithm* $\mathsf{SampleLeft}(\mathbf{A}, \mathbf{A}_\mathbf{x}, \mathbf{u}, \mathbf{T}_\mathbf{A}, \alpha)$*, where the output of* $\mathsf{PRF.Eval}(r, \mathbf{x})$ *is used as the randomness. Finally, it outputs* $\sigma = \mathbf{z}$ *as the signature.*

$\mathsf{S.Verify}(\mathsf{vk}, \mathbf{x}, \sigma)$: *It first checks that* $\mathbf{x} \in \{0,1\}^\ell$*. Next, it checks whether* $[\mathbf{A}|\mathbf{A}_\mathbf{x}]\mathbf{z} = \mathbf{u} \mod q$ *and* $\|\mathbf{z}\|_\infty \leq \beta$*. It outputs 1 if all the above check passes, otherwise it outputs 0.*

Commitment Scheme. Here, we present the commitment scheme of [XXW13] with minor modification. In the following let $[\cdot||\cdot]$ denote the vertical concatenation of vectors.

Theorem 4. *Let* $n, \bar{m}, q$ *be positive integers such that* $\bar{m} \geq 3n$*,* q *a prime. Further, let* γ, γ' *be positive reals such that* $q \geq (4\gamma + 1)^2$ *and* $\gamma \geq \gamma' \omega(\log n)$*.*[13] *Then, the following algorithms* $(\mathsf{C.Gen}, \mathsf{C.Com}, \mathsf{C.Open})$ *form a computationally hiding and statistically binding commitment scheme with gap openings under the hardness of the* $\mathsf{LWE}_{n,\bar{m},q,D_{\mathbb{Z},\gamma}}$ *problem. Here the message space* $\mathcal{M}$ *is* $\mathbb{Z}_q$ *and the commitment space* $\mathcal{C}$ *is* $\mathbb{Z}_q^{\bar{m}}$*.*

$\mathsf{C.Gen}(1^n)$: *It samples* $\mathbf{B} \leftarrow \mathbb{Z}_q^{(n+1) \times \bar{m}}$ *and outputs* $\mathsf{pk} = \mathbf{B}$*.*

$\mathsf{C.Com}(\mathsf{pk}, \mathsf{M})$: *For a message* $\mathsf{M} \in \mathbb{Z}_q$*, it samples a random vector* $\mathbf{s} \leftarrow \mathbb{Z}_q^n$*. Then, it samples* $\mathbf{e} \leftarrow D_{\mathbb{Z}^{\bar{m}},\gamma'}$ *until* $\|\mathbf{e}\|_\infty \leq \gamma$ *holds.*[14] *Finally, it outputs* $(c, d) = (\mathbf{B}^\top[\mathbf{s}||\mathsf{M}] + \mathbf{e} \mod q, (\mathbf{s}, \mathbf{e}))$*.*

$\mathsf{C.Open}(\mathsf{pk}, \mathsf{M}, c, d)$: *It first checks if* $\mathsf{M} \in \mathbb{Z}_q$*. It then parses* $d = (\mathbf{s}, \mathbf{e})$ *and checks if* $c = \mathbf{B}^\top[\mathbf{s}||\mathsf{M}] + \mathbf{e} \mod q$ *and* $\|\mathbf{e}\|_\infty \leq 2\gamma$ *hold. If all the check passes it outputs 1, otherwise it outputs 0.*

Observe that the above commitment scheme has gap-openings; although the commitment algorithm $\mathsf{C.Com}$ only samples vectors $\mathbf{e}$ such that $\|\mathbf{e}\|_\infty \leq \gamma$, the opening algorithm $\mathsf{C.Open}$ accepts $\mathbf{e}$ such that $\gamma < \|\mathbf{e}\|_\infty \leq 2\gamma$ as well.

Furthermore, [XXW13] provides three gap-Σ-protocols for proving useful relations over committed values: Σ_{Open} for proving knowledge of a valid opening and Σ_{Add}, Σ_{Mult}[15] for proving arithmetic relations (over $\mathbb{Z}_q$) of committed values. We additionally construct one useful gap-Σ-protocol $\Sigma_{\mathsf{EqTo}\star}$ for proving that a commitment opens to a specific value. The details of the construction are provided in the full version. Then, the above commitment scheme is equipped with the following four *basic* gap-Σ-protocols.

[13] Here, we use Lem. 4 of [LLNW14] instead of Lem. 1 of [XXW13] to optimize the required parameters of the commitment scheme.

[14] For our parameter selection, this procedure will end in a constant number of trials with all but negligible probability.

[15] In their paper, they present two protocols for proving arithmetic relations, however, in our paper we only consider the more efficient protocol in [XXW13], Sec. 4.3.

Theorem 5. *The commitment scheme with gap openings in Theorem 4 has associating computationally special HVZK gap-Σ-protocols* $(\Sigma_{\mathsf{Open}}, \Sigma_{\mathsf{EqTo}\star}, \Sigma_{\mathsf{Add}}, \Sigma_{\mathsf{Mult}})$ *for the following four relations:*

$$\begin{aligned}
\mathcal{R}_{\mathsf{Open}} &= \{(\mathsf{pk}, c), (\mathsf{M}, d) \mid (c, d) \in \mathcal{D}_{\mathsf{Com}}(\mathsf{pk}, \mathsf{M})\},\\
\mathcal{R}_{\mathsf{EqTo}\star} &= \{(\mathsf{pk}, c, \mathsf{M}), d \mid (c, d) \in \mathcal{D}_{\mathsf{Com}}(\mathsf{pk}, \mathsf{M})\},\\
\mathcal{R}_{\mathsf{Add}} &= \{(\mathsf{pk}, (c_i)_{i=1}^3), ((\mathsf{M}_i, d_i)_{i=1}^3) \mid \mathsf{M}_3 = \mathsf{M}_1 + \mathsf{M}_2\\
&\qquad\qquad \wedge\ (c_i, d_i) \in \mathcal{D}_{\mathsf{Com}}(\mathsf{pk}, \mathsf{M}_i) \text{ for } i \in [3]\},\\
\mathcal{R}_{\mathsf{Mult}} &= \{(\mathsf{pk}, (c_i)_{i=1}^3), ((\mathsf{M}_i, d_i)_{i=1}^3) \mid \mathsf{M}_3 = \mathsf{M}_1 \cdot \mathsf{M}_2\\
&\qquad\qquad \wedge\ (c_i, d_i) \in \mathcal{D}_{\mathsf{Com}}(\mathsf{pk}, \mathsf{M}_i) \text{ for } i \in [3]\}.
\end{aligned}$$

The gap-relations $(\Sigma'_{\mathsf{Open}}, \Sigma'_{\mathsf{EqTo}\star}, \Sigma'_{\mathsf{Add}}, \Sigma'_{\mathsf{Mult}})$ *are defined similarly except that the set* $\mathcal{D}_{\mathsf{G\text{-}Com}}$ *is used instead of* $\mathcal{D}_{\mathsf{Com}}$.

The above gap-Σ-protocols of [XXW13] additionally require internally a standard commitment scheme, which is used by the prover in the first round to send a commitment to the verifier. Although, we can use the commitment scheme of [XXW13] provided above, we use the more efficient lattice-based commitment scheme of Kawachi et al. [KTX08] to instantiate the gap-Σ-protocols. In this case, the communication costs of $\Sigma_{\mathsf{Open}}, \Sigma_{\mathsf{EqTo}\star}$ are $\omega(\bar{m} \log q \log \gamma \log n)$ and $\Sigma_{\mathsf{Add}}, \Sigma_{\mathsf{Mult}}$ are $\omega(\bar{m} \log^3 q \log \gamma \log n)$.

Remark 1. The above four basic gap-Σ-protocols can be composed in parallel to obtain a gap-Σ-protocol for larger relations, e.g., provided with commitments $(c_i)_{i=1}^4$ of the values $(\mathsf{M}_i)_{i=1}^4$ satisfying $\mathsf{M}_4 = \sum_{i=1}^3 \mathsf{M}_i$, we can prove this relation by creating one extra auxiliary commitment c_{aux} for $\mathsf{M}_{\mathsf{aux}} = \mathsf{M}_1 + \mathsf{M}_2$ and running two Σ_{Add} in parallel for the statement pairs $(\mathsf{pk}, c_1, c_2, c_{\mathsf{aux}})$ and $(\mathsf{pk}, c_{\mathsf{aux}}, c_3, c_4)$.

5.2 ABS for Unbounded Circuits Based on Lattices

To instantiate the generic ABS construction in Sec. 4 from lattices, it is sufficient to prove that the above digital signature scheme and commitment scheme are equipped with a gap-Σ-protocol for the relation $\mathcal{R}_{\mathsf{ABS}}$. Therefore, below we aim at constructing a gap-Σ protocol for proving Eq. (5), (6) and (7) in our ABS construction, where the attribute $\mathbf{x}$ and Boyen signatures σ are committed using the commitment scheme of [XXW13]. Here, taking the above Rem. 1 into consideration, a gap-Σ-protocol for proving Eq. (6) and (7), which are essentially proving that the circuit is computed correctly, can be constructed by simply composing the basic gap-Σ-protocols $\Sigma_{\mathsf{EqTo}\star}, \Sigma_{\mathsf{Add}}, \Sigma_{\mathsf{Mult}}$ in parallel. In more detail, we use Σ_{Add} and Σ_{Mult} to prove that we computed each gates correctly, and use $\Sigma_{\mathsf{EqTo}\star}$ to prove that the value associated to the output wire is equal to 1. Therefore, in the following, we only focus on how to construct a gap-Σ-protocol for proving Eq. (5); we construct a gap-Σ-protocol for proving possession of a valid Boyen-signature using $\Sigma_{\mathsf{EqTo}\star}, \Sigma_{\mathsf{Add}}, \Sigma_{\mathsf{Mult}}$. Here, we stress that we cannot simply use the gap-Σ-protocol for proving possession of a valid Boyen-signature

of [LNSW13] for our purpose, since their protocol does not allow us to efficiently prove possession of messages satisfying complex arithmetic relations.[16] In other words, since Eq. (5) and (6) share the same witness $\mathbf{x} = (x_1, \cdots, x_\ell)$, we will not be able to combine the different types of gap-Σ-protocols of [LNSW13] and [XXW13] to construct a gap-Σ protocol for the relation $\mathcal{R}_{\mathsf{ABS}}$.

To summarize, our goal is to construct a gap-Σ-protocol for proving possession of a valid Boyen signature $\sigma = \mathbf{z} = [z_1, \cdots, z_{2m}]^\top \in \mathbb{Z}^{2m}$, where $\mathbf{x} = (x_1, \cdots, x_\ell) \in \{0,1\}^\ell$ is viewed as the message, provided the verification key $\mathsf{vk}_{\mathsf{Sign}}$ and the commitments to the signature σ and message $\mathbf{x}$. Then, since the basic gap-Σ-protocols of Thm. 4 allows for parallel composition, our desired gap-Σ-protocol for the relation $\mathcal{R}_{\mathsf{ABS}}$ is obtained by composing the gap-Σ protocol for the Boyen signature with the gap-Σ-protocols for Eq. (6) and (7) together. Below, we assume the commitment c_σ of the signature is provided in the form $(\bar{c}_k)_{k\in[2m]}$ where each $\bar{c}_i$ is a commitment of the k-th element $z_k \in \mathbb{Z}$ of $\mathbf{z}$ (viewed as an element in $\mathbb{Z}_q$), and the commitment of the message $c_{\mathbf{x}}$ is provided in the form $(c_i)_{i\in[\ell]}$ where each c_i is a commitment of the value $x_i \in \{0,1\}$. Now, due to the verification algorithm of the Boyen signature scheme, proving a signature is valid is equivalent to proving the following three statements:

$$\mathbf{x} \in \{0,1\}^\ell \iff x_i \in \{0,1\} \text{ for } i \in [\ell], \tag{11}$$

$$\|\mathbf{z}\|_\infty \leq \beta \iff |z_k| \leq \beta \text{ for } k \in [2m], \tag{12}$$

$$\left[\mathbf{A}|\mathbf{A}_0 + \sum_{i=1}^{\ell} x_i \mathbf{A}_i\right]\mathbf{z} = \mathbf{u} \mod q, . \tag{13}$$

Below we construct gap-Σ-protocols respectively for the above equations by converting each of them into an arithmetic circuit, and using the basic gap-Σ-protocols provided in Thm. 4 as building blocks to prove the satisfiability of each circuit.

Gap-Σ-Protocol for Proving Eq. (11). It is sufficient to prove that for every $i \in [\ell]$, the commitment $c_i \leftarrow \mathsf{C.Com}(\mathsf{pk}, x_i)$ opens to either 0 or 1. To do so, we first create auxiliary commitments $c_{\mathsf{zero}} \leftarrow \mathsf{C.Com}(\mathsf{pk}, 0)$ and $g_i \leftarrow \mathsf{C.Com}(\mathsf{pk}, x_i^2)$ for $i \in [\ell]$. Then using the commitments $(c_i)_{i\in[\ell]}$ and the auxiliary commitments, and combining the basic gap-Σ-protocols $\Sigma_{\mathsf{EqTo}\star}$, Σ_{Add} and Σ_{Mult} together, we construct a gap-Σ-protocol for proving the following statement for all $i \in [\ell]$:

$$c_{\mathsf{zero}} \text{ opens to } 0 \quad \wedge \quad x_i^2 = x_i \cdot x_i \quad \wedge \quad 0 = x_i^2 - x_i$$

Since all arithmetic operations are over the finite field $\mathbb{Z}_q$, the only x_i that satisfy the above relations are $x_i = 0$ or 1. Therefore, the above gap-Σ-protocol indeed proves Eq. (11). The total communication cost is $\omega(\ell \bar{m} \log^3 q \log \gamma \log n)$[17].

[16] The subsequent works of [LLM+16,YAL+17] allow proving possession of a valid Boyen-signature while also proving possession of messages satisfying some simple arithmetic relations. However, their protocols are not strong enough to prove arbitrary circuits in zero-knowledge.

[17] Note that $\omega(f(X))$ denotes any function that grows asymptotically faster than $f(X)$, e.g., $\log^2(X) = \omega(\log(X))$.

Gap-Σ-Protocol for Proving Eq. (12). Here, for simplicity of the protocol, we assume that β can be written as $2^\zeta - 1$ for some positive integer ζ. Equivalently, $\zeta = \log(\beta+1)$. This does not harm the efficiency nor the security of the signature scheme by much, since given any β, there always exists a value of the form $2^\zeta - 1$ in between β and 2β.

First, we prepare some notations. For $k \in [2m]$, let $z_{k,j}$ be the j-th bit of the binary representation of $z_k \in \mathbb{Z}$ for $j \in [\zeta]$. Note that, we extend the standard binary decomposition to negative integers as well in the obvious way. In particular, we can bit decompose any $z_k \in [-\beta, \beta]$ as $z_k = \sum_{j=1}^{\zeta} 2^{j-1} z_{k,j}$, where $z_{k,j} \in \{-1, 0, 1\}$.[18] Further, set $w_{k,j} = 2^{j-1} z_{k,j}$ for $j \in [\zeta]$ and $w_{k,[j']} = \sum_{j=1}^{j'} w_{k,j}$ for $j' \in [2, \zeta]$. Finally, define $w_{k,[1]} = w_{k,1}$. Next, create the following auxiliary commitments for $k \in [2m]$: $c_{\mathsf{zero}} \leftarrow \mathsf{C.Com}(\mathsf{pk}, 0)$, $c_{\mathsf{coeff},j} \leftarrow \mathsf{C.Com}(\mathsf{pk}, 2^{j-1})$, $\bar{c}_{k,j,\mu} \leftarrow \mathsf{C.Com}(\mathsf{pk}, z_{k,j}^{\mu})$, $h_{k,j} \leftarrow \mathsf{C.Com}(\mathsf{pk}, w_{k,j})$ for $\mu \in [3]$, $j \in [\zeta]$, and $h_{k,[j']} \leftarrow \mathsf{C.Com}(\mathsf{pk}, w_{k,[j']})$ for $j' \in [2, \zeta]$. Then, using the commitments $(\bar{c}_k)_{k \in [2m]}$, the auxiliary commitments and composing the gap-Σ-protocols $\Sigma_{\mathsf{EqTo}\star}$, Σ_{Add} and Σ_{Mult} together, we construct a gap-Σ-protocol for the following statement for all $k \in [2m], j \in [\zeta]$ and $j' \in [2, \zeta]$:[19]

$$c_{\mathsf{zero}} \text{ opens to } 0 \ \wedge \ c_{\mathsf{coeff},j} \text{ opens to } 2^j \ \wedge \ z_{k,j}^2 = z_{k,j} \cdot z_{k,j} \ \wedge \ z_{k,j}^3 = z_{k,j}^2 \cdot z_{k,j} \ \wedge$$
$$0 = z_{k,j}^3 - z_{k,j} \wedge w_{k,j} = 2^{j-1} \cdot z_{k,j} \wedge w_{k,[j']} = w_{k,j'} + w_{k,[j'-1]} \wedge 0 = z_k - w_{k,[\zeta]}.$$

We check that the above statement is equivalent to Eq. (12), i.e., each z_k satisfy $|z_k| \leq \beta$ for all $k \in [2m]$. First, since q is a prime, the only $z_{k,j}$ satisfying $z_{k,j}^3 - z_{k,j} = 0$ over $\mathbb{Z}_q$ are $-1, 0, 1$. Hence, the above statement proves that $z_{k,j} \in \{-1, 0, 1\}$. Furthermore, when $z_{k,j} \in \{-1, 0, 1\}$, we have $|z_k| \leq \sum_{j=1}^{\zeta} 2^{j-1} |z_{k,j}| \leq 2^{\zeta-1} = \beta$. Therefore, if the above statement holds, then we must have $|z_k| \leq \beta$ for all $k \in [2m]$. The total communication cost is $\omega(m\bar{m} \log \beta \log^3 q \log \gamma \log n)$.

Gap-Σ-Protocol for Proving Eq. (13). We first prepare some notations. Let $a_{s,k}$ (resp., $a_{i,s,k}$) denote the (s, k_1)-th (resp., $(s, k_2 - m)$-th) entry of $\mathbf{A}$ (resp., $\mathbf{A}_i$) $\in \mathbb{Z}_q^{n \times m}$, for $s \in [n]$, $k_1 \in [m]$ (resp., $k_2 \in [m+1, 2m]$) and $i \in [0, \ell]$. Then, observe that we can rewrite Eq. (13) using the following equations for $s \in [n]$:

$$\sum_{k_1=1}^{m} a_{s,k_1} \cdot z_{k_1} + \sum_{k_2=m+1}^{2m} \left(a_{0,s,k_2} + \sum_{i=1}^{\ell} x_i \cdot a_{i,s,k_2} \right) \cdot z_{k_2} = u_s \tag{14}$$

Next, we prepare some auxiliary values for $s \in [n]$ in order to prove the above equations using the gap-Σ-protocols $\Sigma_{\mathsf{EqTo}\star}$, Σ_{Add} and Σ_{Mult}: $w_{i,s,k_2} = x_i \cdot a_{i,s,k_2}$,

[18] A subtly is that unlike standard bit decomposition, the bit representation is not unique anymore, e.g., 11 can be decomposed as $(1, 1, 0, 1)$ or $(-1, 0, 1, 1)$. However, this will not affect our following argument.

[19] Since we prove c_{zero} opens to 0 in the above gap-Σ-protocol for proving Eq. (11), we will not require this when we compose the gap-Σ-protocols together. The same holds for the aforementioned gap-Σ-protocol for proving Eq. (13).

$w_{[i'],s,k_2} = \sum_{i=1}^{i'} w_{i,s,k_2}$, $a_{s,k_2} = a_{0,s,k_2} + w_{[\ell],s,k_2}$ for $i \in [\ell]$, $i' \in [2,\ell]$, $k_2 \in [m+1, 2m]$, $b_{s,k} = a_{s,k} \cdot z_k$ for $k \in [2m]$, $b_{s,[k']} = \sum_{k_1=1}^{k'} b_{s,k_1}$ for $k' \in [2,m]$, $b_{s,[k']} = \sum_{k_2=m+1}^{k'} b_{s,k_2}$ for $k' \in [m+2, 2m]$ and $t_s = b_{s,[m]} + b_{s,[2m]}$. Further define $w_{[1],s,k_2} = w_{1,s,k_2}$, $b_{s,[1]} = b_{s,1}$ and $b_{s,[m+1]} = b_{s,m+1}$. Next, we create auxiliary commitments for the related values for $s \in [n]$: $c_{\mathsf{mat},s,k_1} \leftarrow \mathsf{C.Com}(\mathsf{pk}, a_{s,k_1})$, $c_{\mathsf{mat},i,s,k_2} \leftarrow \mathsf{C.Com}(\mathsf{pk}, a_{i,s,k_2})$ for $i \in [0,\ell]$, $k_1 \in [m]$, $k_2 \in [m+1, 2m]$, $\omega_{i,s,k_2} \leftarrow \mathsf{C.Com}(\mathsf{pk}, w_{i,s,k_2})$, $\omega_{[i'],s,k_2} \leftarrow \mathsf{C.Com}(\mathsf{pk}, w_{[i'],s,k_2})$, $\alpha_{s,k_2} \leftarrow \mathsf{C.Com}(\mathsf{pk}, a_{s,k_2})$ for $i \in [\ell]$, $i' \in [2,\ell]$, $k_2 \in [m+1, 2m]$, $\beta_{s,k} \leftarrow \mathsf{C.Com}(\mathsf{pk}, b_{s,k})$ for $k \in [2m]$, $\beta_{s,[k']} \leftarrow \mathsf{C.Com}(\mathsf{pk}, b_{s,[k']})$ for $k' \in [2,m] \cup [m+2, 2m]$. Then, using the commitment $(c_i)_{i=1}^{\ell}$, $(\bar{c}_k)_{k\in[2m]}$, the auxiliary commitments and composing the gap-Σ-protocols $\Sigma_{\mathsf{EqTo}\star}$, Σ_{Add} and Σ_{Mult} together, we construct a gap-Σ-protocol for the following statement for all $s \in [n]$, $i \in [\ell]$, $i' \in [2,\ell]$, $k_1 \in [m]$, $k_2 \in [m+1, 2m]$, $k \in [2m]$, $k' \in [2,m] \cup [m+2, 2m]$:

$$
\begin{aligned}
& c_{\mathsf{zero}} \text{ opens to } 0 \quad \wedge \\
& c_{\mathsf{mat},s,k_1},\, c_{\mathsf{mat},0,s,k_2},\, c_{\mathsf{mat},i,s,k_2} \text{ opens to } a_{s,k}, a_{0,s,k}, a_{i,s,k}, \text{ respectively} \quad \wedge \\
& w_{i,s,k_2} = x_i \cdot a_{i,s,k_2} \quad \wedge \quad w_{[i'],s,k_2} = w_{i',s,k_2} + w_{[i'-1],s,k_2} \quad \wedge \\
& a_{s,k_2} = a_{0,s,k_2} + w_{[\ell],s,k_2} \quad \wedge \quad b_{s,k} = a_{s,k} \cdot z_k \quad \wedge \quad b_{s,[k']} = b_{s,k'} + b_{s,[k'-1]} \quad \wedge \\
& t_s = b_{s,[m]} + b_{s,[2m]} \quad \wedge \quad 0 = u_s - t_s
\end{aligned}
$$

The above statement can be checked that it is equivalent to proving Eq. (14) for $s \in [n]$. The total communication cost is $\omega(\ell n m \bar{m} \log^3 q \log \gamma \log n)$.

Gap-Σ-Protocol for $\mathcal{R}_{\mathsf{ABS}}$. To summarize, we obtain a gap-Σ-protocol for proving possession of a valid Boyen signature by composing the gap-Σ-protocols for proving Eq. (11-13) together. Then, by composing this protocol with the aforementioned gap-Σ-protocols for proving Eq. (6) and (7), we obtain our desired gap-Σ-protocol for the relation $\mathcal{R}_{\mathsf{ABS}}$ where the total communication cost is $\omega((m(\ell n + \log \beta) + |C|)\bar{m} \log^3 q \log \gamma \log n)$. Here, $|C|$ is size of the circuit (i.e., policy) associated to the message. Thus, we obtain our lattice-based ABS scheme for *unbounded circuits* in the random oracle model by instantiating the generic ABS construction in Sec. 4 with our gap-Σ protocol for $\mathcal{R}_{\mathsf{ABS}}$.

Acknowledgement. We would like to thank the anonymous reviewers of PKC 2018 for insightful comments. In particular, we are grateful for Yusuke Sakai and Takahiro Matsuda for the helpful discussions and feedback on this work. The first author was funded by a research grant from the UK Government. The second author was partially supported by JST CREST Grant Number JPMJCR1302 and JSPS KAKENHI Grant Number 17J05603.

References

AJLA+12. Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold fhe. In *EUROCRYPT*, pages 483–501. Springer, 2012.

BCK+14. Fabrice Benhamouda, Jan Camenisch, Stephan Krenn, Vadim Lyubashevsky, and Gregory Neven. Better zero-knowledge proofs for lattice encryption and their application to group signatures. In *ASIACRYPT*, pages 551–572. Springer, 2014.

BF14. Mihir Bellare and Georg Fuchsbauer. Policy-based signatures. In *PKC*, pages 520–537. Springer, 2014.

BFW16. David Bernhard, Marc Fischlin, and Bogdan Warinschi. On the hardness of proving cca-security of signed elgamal. In *PKC*, pages 47–69. Springer, 2016.

BKLP15. Fabrice Benhamouda, Stephan Krenn, Vadim Lyubashevsky, and Krzysztof Pietrzak. Efficient zero-knowledge proofs for commitments from learning with errors over rings. In *ESORICS*, pages 305–325. Springer, 2015.

BN06. Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In *CCS*, pages 390–399. ACM, 2006.

Boy10. Xavier Boyen. Lattice mixing and vanishing trapdoors: A framework for fully secure short signatures and more. In *PKC*, pages 499–517. Springer, 2010.

BPVY00. Ernest Brickell, David Pointcheval, Serge Vaudenay, and Moti Yung. Design validations for discrete logarithm based signature schemes. In *PKC*, pages 276–292. Springer, 2000.

BPW12. David Bernhard, Olivier Pereira, and Bogdan Warinschi. How not to prove yourself: Pitfalls of the fiat-shamir heuristic and applications to helios. In *ASIACRYPT*, pages 626–643. Springer, 2012.

DF02. Ivan Damgård and Eiichiro Fujisaki. A statistically-hiding integer commitment scheme based on groups with hidden order. In *ASIACRYPT*, pages 77–85. Springer, 2002.

EE16. Rachid El Bansarkhani and Ali El Kaafarani. Post-quantum attribute-based signatures from lattice assumptions. Cryptology ePrint Archive, Report 2016/823, 2016.

EGK14. Ali El Kaafarani, Essam Ghadafi, and Dalia Khader. Decentralized traceable attribute-based signatures. In *CT-RSA*, pages 327–348. Springer, 2014.

FKMV12. Sebastian Faust, Markulf Kohlweiss, Giorgia Azzurra Marson, and Daniele Venturi. On the non-malleability of the fiat-shamir transform. In *INDOCRYPT*, pages 60–79. Springer, 2012.

FO97. Eiichiro Fujisaki and Tatsuaki Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In *CRYPTO*, pages 16–30. Springer, 1997.

FS86. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, pages 186–194. Springer, 1986.

Gen09. Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–169. ACM, 2009.

GVW15. Sergey Gorbunov, Vinod Vaikuntanathan, and Daniel Wichs. Leveled fully homomorphic signatures from standard lattices. In *STOC*, pages 469–477. ACM, 2015.

HLLR12. Javier Herranz, Fabien Laguillaumie, Benoît Libert, and Carla Ràfols. Short attribute-based signatures for threshold predicates. In *CT-RSA*, pages 51–67. Springer, 2012.

KTX08. Akinori Kawachi, Keisuke Tanaka, and Keita Xagawa. Concurrently secure identification schemes based on the worst-case hardness of lattice problems. In *ASIACRYPT*, pages 372–389. Springer, 2008.

LLM+16. Benoît Libert, San Ling, Fabrice Mouhartem, Khoa Nguyen, and Huaxiong Wang. Zero-knowledge arguments for matrix-vector relations and lattice-based group encryption. In *ASIACRYPT*, pages 101–131. Springer, 2016.

LLNW14. Adeline Langlois, San Ling, Khoa Nguyen, and Huaxiong Wang. Lattice-based group signature scheme with verifier-local revocation. In *PKC*, pages 345–361. Springer, 2014.

LNSW13. San Ling, Khoa Nguyen, Damien Stehlé, and Huaxiong Wang. Improved zero-knowledge proofs of knowledge for the isis problem, and applications. In *PKC*, pages 107–124, 2013.

LNW15. San Ling, Khoa Nguyen, and Huaxiong Wang. Group signatures from lattices: simpler, tighter, shorter, ring-based. In *PKC*, pages 427–449. Springer, 2015.

Lyu09. Vadim Lyubashevsky. Fiat-shamir with aborts: applications to lattice and factoring-based signatures. In *ASIACRYPT*, pages 598 – 616. Springer, 2009.

Lyu12. Vadim Lyubashevsky. Lattice signatures without trapdoors. In *EUROCRYPT*, pages 738–755. Springer, 2012.

MP12. Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT*, pages 700–718. Springer, 2012.

MPR11. Hemanta K Maji, Manoj Prabhakaran, and Mike Rosulek. Attribute-based signatures. In *CT-RSA*, pages 376–392. Springer, 2011.

Nao91. Moni Naor. Bit commitment using pseudorandomness. *Journal of cryptology*, pages 151–158, 1991.

OT11. Tatsuaki Okamoto and Katsuyuki Takashima. Efficient attribute-based signatures for non-monotone predicates in the standard model. In *PKC*, pages 35–52. Springer, 2011.

OT13. Tatsuaki Okamoto and Katsuyuki Takashima. Decentralized attribute-based signatures. In *PKC*, pages 125–142. Springer, 2013.

PS00. David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of cryptology*, pages 361–396, 2000.

PSV06. Rafael Pass, Abhi Shelat, and Vinod Vaikuntanathan. Construction of a non-malleable encryption scheme from any semantically secure one. In *CRYPTO*, pages 271–289. Springer, 2006.

Rom90. John Rompel. One-way functions are necessary and sufficient for secure signatures. In *STOCS*, pages 387–394. ACM, 1990.

SAH16. Yusuke Sakai, Nuttapong Attrapadung, and Goichiro Hanaoka. Attribute-based signatures for circuits from bilinear map. In *PKC*, pages 283–300. Springer, 2016.

Tsa17. Rotem Tsabary. An equivalence between attribute-based signatures and homomorphic signatures, and new constructions for both. Cryptology ePrint Archive, Report 2017/723, to appear in TCC 2017.

Wee09. Hoeteck Wee. Zero knowledge in the random oracle model, revisited. In *ASIACRYPT*, pages 417–434. Springer, 2009.

XXW13. Xiang Xie, Rui Xue, and Minqian Wang. Zero knowledge proofs from ring-lwe. In *CANS*, pages 57–73. Springer, 2013.

YAL+17. Rupeng Yang, Man Ho Au, Junzuo Lai, Qiuliang Xu, and Zuoxia Yu. Lattice-based techniques for accountable anonymity: Composition of abstract stern 's protocols and weak prf with efficient protocols from lwr. Cryptology ePrint Archive, Report 2017/781, 2017.