Attribute-Based Signatures for Circuits with Optimal Parameter Size from Standard Assumptions

Ryuya Hayashi[1,2], Yusuke Sakai[2], and Shota Yamada[2]

[1] The University of Tokyo, Tokyo, Japan
rhys@iis.u-tokyo.ac.jp
[2] AIST, Tokyo, Japan
{yusuke.sakai, yamada-shota}@aist.go.jp

Abstract. Attribute-based signatures (ABS) allow users to simultaneously sign messages and prove their possession of some attributes while hiding the attributes and revealing only the fact that they satisfy a public policy. In this paper, we propose a generic construction of ABS for circuits of unbounded depth and size with optimal parameter size, meaning that the lengths of public parameters, keys, and signatures are all constant. Our generic construction can be instantiated from various standard assumptions including LWE or DLIN. Only previous ABS construction with optimal parameter size necessitates succinct non-interactive argument of knowledge, which can be only constructed from non-standard assumptions. Our generic construction is based on RAM delegations, which intuitively allows us to compress the evaluation of a circuit when inputs are public. In high level, we find a way to compress the computation of the policy circuit on input a user attribute to achieve overall parameter size, while hiding the user policy at the same time.

1 Introduction

1.1 Backgrounds

Attribute-based signatures (ABS), firstly proposed by Maji, Prabhakaran, and Rosulek [35], allow users to simultaneously sign messages and prove their possession of some attributes while hiding the attributes and revealing only the fact that they satisfy a public policy. In the typical scenario of using ABS, we consider two entities: a key issuing authority and signers. The authority first generates a master secret key together with some public parameter and issues a user secret key associated with the user's attribute. After receiving the user secret key, each signer can generate a signature on a message with a policy. Such a signature is publicly verifiable, and anyone can verify that a signer who generates the signature has some attributes that satisfy the policy if the verification passes. The important feature is that the signature hides the attributes used to satisfy the policy and any information identifying the signer. ABS draws increasing attention for its applications such as anonymous credentials [44], non-transferable access controls [33], electronic medical records [25], etc.

Table 1: Comparison of efficiency among expressive ABS schemes.

	Policy	Param.	Sig.	Key	Assumption
BGI14 [4]	Unbounded circuits*	$O(1)$	$O(1)$	$O(1)$	zk-SNARKs
SAH16 [42]	Unbounded circuits	$O(\|x\|)$	$O(\|C\|)$	$O(1)$	pairings
EK18 [18]	Unbounded circuits	$O(\|x\|)$	$O(\|C\|)$	$O(1)$	lattices in ROM
SKAH18 [43]	TM†	$O(1)$	$O(T^2)$	$O(\|\mathit{\Gamma}\|)$	pairings
DDK23 [12]	TM	$O(1)$	$O(1)$	$O(\|x\|)$	iO
LNP+24 [34]	Unbounded circuits	$O(\|x\|)$	$O(\|C\|)$	$O(1)$	codes in QROM
Ours	Unbounded circuits	$O(1)$	$O(1)$	$O(1)$	pairings or lattices

* Unbounded circuits : Circuits of unbounded depth and size
† TM : Unbounded polynomial-time deterministic Turing machines of unbounded input length and description size
$|x|$: Length of attributes
$|C|$: Length of policy circuits
T : Computational time of Turing machines
$|\mathit{\Gamma}|$: Length of policy descriptions

Expressiveness and Efficiency. After Maji et al. [35] introduced the notion of ABS and proposed ABS schemes for monotone span programs, earlier works [1, 8, 26, 33, 38, 39, 39, 44–46] proposed ABS for limited class of policies. The work by Sakai, Attrapadung, and Hanaoka [42], who proposed an ABS for *circuits with unbounded depth and size*, significantly broadened the class of policies. After this work, several works proposed ABS schemes for quite expressive classes of policies including unbounded circuits [18, 34] and Turing machines [12, 43]. We summarized these schemes in Section 1.1. As shown in the table, no existing ABS scheme that can deal with circuits or more expressive class realizes optimal parameters, i.e., constant size of the public parameters, signatures, and secret keys at the same time. Only exception is the construction proposed by [4], but it requires succinct non-interactive arguments of knowledge (SNARKs) as a building block, whose instantiation is not known from standard assumptions. Given the state of affairs, we pose the following natural question:

Can we construct an ABS for circuits with optimal parameter size from standard assumptions?

1.2 Our Contribution

The main contribution of this paper is to propose a construction of an ABS for circuits with unbounded depths and sizes that has optimal parameter size, answering the above question in the affirmative. As shown in Section 1.1, no existing scheme but BGI14 [4] does not achieve the constant lengths. Moreover, our construction can be constructed from any of the following assumptions: LWE, DLIN over pairing groups, or simultaneously assuming QR and DDH over groups without pairings. Namely, we are the first to propose such an optimal ABS from standard assumptions.

1.3 Technical Overview

We provide a generic construction of ABS with constant parameters. Our generic construction is from a public-key encryption scheme, a signature scheme, a RAM SNARG corresponding to the specific hash family, and a non-interactive zero-knowledge (NIZK) proof system.

For brevity, we consider a simpler variant of an ABS, called a constrained signature (CS) [3, 46], in which a prover only generates a proof to show that it has attributes x to satisfy some policy C, i.e., we do not consider the part of signing a message m in the following.

Naive Construction. The difficulty of constructing a succinct CS for unbounded circuits is how to realize constant lengths of user secret keys and signatures without revealing other information than the fact that each has attributes x to satisfy some policy C, i.e., $C(x) = 1$. If we do not consider the requirement of the succinctness, a naive construction is informally as follows:

- The key issuing authority, given some attribute x, generates a signature σ_x of its attribute and sends the signature to each user as its user secret key.
- Each user generates a NIZK proof Π to show that (i) it has a pair (x, σ_x) such that σ_x is a valid signature of x and (ii) $C(x) = 1$ holds. Then, it publishes Π as its signature.
- Verifiers check if the received proof Π is valid with the given C.

Of course, the proof length in the above naive construction depends both on the attribute and the policy size, which we should avoid.

Our Approach using RAM Delegations. Our first attempt is to make the verification of $C(x) = 1$ succinct by using publicly verifiable RAM delegations [6, 11, 28–30]. A RAM delegation is a succinct non-interactive argument (SNARG) for RAM computations where a prover, given a common reference string crs, a RAM machine $\mathcal{R}$, and an input x, can produce a short proof that the RAM machine $\mathcal{R}$ takes x as input and outputs y. The important feature of publicly verifiable RAM delegations is that any one can check the validity of the proof *with the short digest of the input to the RAM machine.* In addition, the proof length and the verifier runtime is independent of the input length and polylogarithmic in the runtime of the RAM machine $\mathcal{R}$. Therefore, by using a RAM delegation for a RAM machine $\mathcal{R}$ that takes (x, C) as inputs and computes $C(x) = b \in \{0, 1\}$, each user can generate a succinct proof to show that $C(x) = 1$, and the proof can be verified with the short digest of (x, C). We next show a simple construction of ABS using RAM delegations.

Simple Construction by applying RAM Delegations. In a simple construction of a CS using RAM delegations, similar to the naive construction, each user generates a NIZK proof to show that the verification of the RAM delegation passes by including the digest of the input in the witness. More concretely, the simple construction is as follows (we use the $\underline{\text{underline}}$ for the different part from the naive construction for clarity):

- The key issuing authority, given some attribute x, generates a signature σ_x of its attribute and sends the signature to each user as its user secret key.
- Each user proceeds as follows:
  1. $\underline{\text{Computes a digest } \mathsf{d} \text{ of the input } (x, C) \text{ and a proof } \pi \text{ of the RAM delegation to show that } C(x) = 1}$;
  2. Generate a NIZK proof $\varPi$ to show that (i) it has a pair (x, σ_x) such that σ_x is a valid signature of x, (ii) $\underline{\text{the verification of the RAM delegation passes with the digest } \mathsf{d} \text{ and the proof } \pi}$, and (iii) $\underline{\text{the digest } \mathsf{d} \text{ is correctly computed by taking } (x, C) \text{ as input}}$.

  Then, it publishes $\varPi$ as its signature.
- Verifiers check if the received proof $\varPi$ is valid with the given C.

However, the signature size in the above construction still depends on the lengths both of attributes and policies, so we need a little more work.

New Primitive: Circuit Delegations. The reason why the signature size depends on the lengths both of attributes and policies is due to the fact that we need to prove (i) and (iii) using NIZK in the above simple construction. To achieve the succinctness, our key idea is to make the digest value of (x, C) computable from the digest of x and one of C separately. If the digest value can be separately computed as this, the NIZK proof size will be succinct by having the key issuing authority sign the digest of x since verifiers know C and can compute the digest of C by themselves. Fortunately, we can use arbitrary hash family for computing the digest in RAM delegations as long as the hash family can locally open a bit of each position. Kalai et al. [28] introduced the formal definition of such RAM delegations and called them *flexible RAM SNARGs*.

For easily understanding of this technique, we introduce a new notion, a *Circuit SNARG*, that allows us to verify $C(x) = b \in \{0, 1\}$ with the digest values of the input (x, C) and a succinct proof π. More precisely, in Circuit SNARGs, a prover can generate a succinct proof π to show that $C(x) = b$, and verifiers, given the digest value of x, the digest value of C, the output bit b, and the proof π, can check the validity of the given proof. This primitive can be easily constructed from a flexible RAM SNARG for the RAM machine $\mathcal{R}$ that takes x and C as input and outputs $C(x)$. The flexible RAM SNARG corresponds to a specific hash family, in which the digest value of (x, C) consists of the digest of x and one of C. Such a hash family can be also easily constructed from any hash family in a generic way. See Section 3 in detail.

The Overview of Our Construction. Using the above Circuit SNARGs, we can construct a succinct CS as follows (we use the $\underline{\text{underline}}$ for the different part from the simple construction for clarity):

- The key issuing authority, given some attribute x, $\underline{\text{computes the digest } \mathsf{d}_x \text{ of } x\text{, generates a signature } \sigma_{\mathsf{d}_x} \text{ of the digest}}$, and sends $\underline{\text{the digest } \mathsf{d}_x \text{ and the signature } \sigma_{\mathsf{d}_x}}$ to each user as its user secret key.
- Each user proceeds as follows:
  1. Computes a digest d_x of the part x of the input and a proof π of the *Circuit SNARG* to show that $C(x) = 1$;
  2. Generate a NIZK proof Π to show that (i) it has a pair $\underline{(\mathsf{d}_x, \sigma_{\mathsf{d}_x}) \text{ such that } \sigma_{\mathsf{d}_x} \text{ is a valid signature of } \mathsf{d}_x}$ and (ii) the verification $\underline{\text{of the } \textit{Circuit SNARG}}$ passes with the digests $\underline{\mathsf{d}_x \text{ and } \mathsf{d}_C}$, and the proof π, $\underline{\text{where } \mathsf{d}_C \text{ is the digest of } C}$.

  Then, it publishes Π as its signature.

In the above construction, it is easy to see that all lengths of public parameters, user secret keys, and signatures are polylogarithmic in the lengths both of attributes and circuits. To make the above construction of CS apply to ABS, we will use a simulation-sound NIZK and add a ciphertext of a NIZK witness to a signature in ABS. The formal description and security analysis of our scheme will be provided in Section 4. As a result, we obtain the following informal theorem.

Theorem 1.1. *(Informal.) If the public-key encryption scheme, signature scheme, circuit SNARG, and NIZK are secure, then the above construction of ABS with constant parameters is secure.*

In particular, each building block of our construction is known to be constructed from assumptions either pairings or lattices. See Section 1.4 in detail. Thus, we obtain the following corollary.

Corollary 1.1. *(Informal.) There is an ABS for unbounded circuits with constant parameters based either on the DLIN (on pairings) or LWE (on lattices) assumption.*

1.4 Related Works

A Line of Work in ABS. Maji et al. [35] first proposed an ABS for monotone span programs, in which each signature size depends on the policy size. Following their work, several works have studied ABS for various classes of computations including conjunction predicates [33,44], non-monotone span programs [1,39,45], threshold policies [8,26], bounded circuits [46], and deterministic finite automata [38]. The work by Sakai et al. [42], who proposed an ABS for unbounded circuits, significantly broadened the class of policies. After this, several works realized ABS schemes for quite expressive classes of policies as follows. El Kaafarani and Katsumata [18], and Ling et al. [34] proposed an ABS for unbounded circuits, but its signatures size depends on the size of the circuits. Sakai et al. [43] proposed an ABS for Turing machines, but its signatures size is quadratic to the running time of the Turing machines. Datta et al. [12] also proposed an ABS for Turing machines with better parameters, but its length of keys stll depends on the length of attributes.

ABS with Additional Functionalities and Security Requirements. In this paper, we focus on the standard ABS, but there are also some variants of ABS. For example, Escala et al. [19, 34] proposed a *revocable* ABS that allows an external judge to break the anonymity of signatures. Okamoto and Takashima [40] proposed a *decentralized* ABS, in which there are multiple authorities to issue user secret keys and is no central authority. Zhang et al. [48] recently proposed a *registered* ABS that allows any user to generate their own key pairs and register them to the system. In addition to these, some works have considered traceability [14], hierarchical variants [16, 17, 20, 23], and the universal composability [2].

SNARGs and RAM Delegations. Existing works [5, 7, 9–11, 13, 22, 24, 28, 30–32, 37, 47] studied succinct non-interactive arguments (SNARGs) for efficiently verifying computations. Choudhuri, Jain, and Jin [11] and Kalai, Vaikuntanathan, and Zhang [32] proposed a generic compiler from SNARGs for Batch-NP computations (BARGs) with somewhere extractable succinct commitment schemes with local opening (SECOM) to RAM delegations for deterministic polynomial-time computations. Then, Kalai, Lombardi, Vaikuntanathan, and Wichs [28] bootstrapped the efficiency of RAM delegations with succinctness $\mathsf{poly}(\lambda, \log T)$ constructed from any BARG and SECOM, where T is the computational time of the RAM machine. This result implies that succinct RAM delegations can be constructed from various computational assumptions since SECOM can be constructed from any of the following assumptions: LWE, QR, DDH, or DLIN [15], and BARGs can be constructed from any of the following assumptions: LWE [11], QR and DDH [27], or DLIN [47].

2 Preliminaries

In this section, we review basic notations and formal definitions of primitives.

Notation. In this paper, we use the following notations. $x \leftarrow X$ denotes sampling an element x from a finite set X uniformly at random. $y \leftarrow \mathcal{A}(x; r)$ denotes that a probabilistic algorithm $\mathcal{A}$ outputs y for an input x using a randomness r, and we simply denote $y \leftarrow \mathcal{A}(x)$ when we need not write an internal randomness explicitly. For strings x and y, $x||y$ denotes the concatenation of x and y. Also, $x \coloneqq y$ denotes that x is defined by y, and $|x|$ denotes the length of x. λ denotes a security parameter. A function $f(\lambda)$ is a negligible function in λ if $f(\lambda)$ tends to 0 faster than $\frac{1}{\lambda^c}$ for every constant $c > 0$. $\mathsf{negl}(\lambda)$ denotes an unspecified negligible function. PPT stands for probabilistic polynomial time. $\emptyset$ denotes the empty set. If n is a natural number, $[n]$ denotes the set of integers $\{1, \cdots, n\}$. If

x is a n bits string, x_i denotes the i-th bit of the string x for any $i \in [n]$. If $\mathcal{O}$ is a function or an algorithm and $\mathcal{A}$ is an algorithm, $\mathcal{A}^{\mathcal{O}}$ denote that $\mathcal{A}$ has oracle access to $\mathcal{O}$.

2.1 Public-Key Encryption

We recall a definition of a public-key encryption (PKE) scheme.

Definition 2.1 (Public-Key Encryption). *A PKE scheme* PKE *with a plaintext space* $\mathbb{M}$ *consists of the following three PPT algorithms.*

$\mathsf{PKE.KG}(1^\lambda) \to (\mathsf{ek}, \mathsf{dk})$ *: The key generation algorithm, given a security parameter* 1^λ*, outputs an encryption key* ek *and a decryption key* dk.

$\mathsf{PKE.Enc}(\mathsf{ek}, m) \to c$ *: The encryption algorithm, given an encryption key* ek *and a plaintext* m*, outputs a ciphertext* c.

$\mathsf{PKE.Dec}(\mathsf{dk}, c) \to m$ *: The (deterministic) decryption algorithm, given a decryption key* dk*, and a ciphertext* c*, outputs a plaintext* $m \in \{\perp\} \cup \mathbb{M}$.

Furthermore, we require a PKE scheme to satisfy the following standard properties.

Correctness. *For all* $\lambda \in \mathbb{N}$ *and* $m \in \mathbb{M}$*, we have*

$$\Pr[(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{PKE.KG}(1^\lambda) : \mathsf{PKE.Dec}(\mathsf{dk}, \mathsf{PKE.Enc}(\mathsf{ek}, m)) = m] = 1.$$

IND-CPA Security. *For any PPT adversary* $\mathcal{A}$*, the following advantage is negligible:*

$$\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{PKE},\mathcal{A}}(\lambda) := \left| \Pr\left[\begin{array}{c} b \leftarrow \{0,1\}, \\ (\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{PKE.KG}(1^n), \\ (m_0^*, m_1^*, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{ek}), \\ c^* \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}, m_b^*), \\ b' \leftarrow \mathcal{A}(c^*, \mathsf{st}) \end{array} : b = b' \right] - \frac{1}{2} \right|,$$

where $\mathcal{A}$ *is required to output* m_0^* *and* m_1^* *satisfying* $|m_0^*| = |m_1^*|$.

2.2 Signature Scheme

Here we recall the definition of a signature scheme.

Definition 2.2 (Signature). *A signature scheme* SIG *with a message space* $\mathbb{M}$ *consists of the following three PPT algorithms.*

$\mathsf{SIG.KG}(1^\lambda) \to (\mathsf{vk}, \mathsf{sk})$ *: The key generation algorithm, given a security parameter* 1^λ*, outputs a verification key* vk *and a signing key* sk.

$\mathsf{SIG.Sign}(\mathsf{sk}, m) \to \sigma$ *: The signing algorithm, given a signing key* sk *and a message* m*, outputs a signature* σ.

$\mathsf{SIG.Ver}(\mathsf{vk}, m, \sigma) \to 1/0$: *The (deterministic) verification algorithm, given a verification key* vk*, a message* m*, and a signature* σ*, outputs either* 1 *(accept) or* 0 *(reject).*

Furthermore, we require a signature scheme to satisfy the following properties.

Correctness. *For all* $\lambda \in \mathbb{N}$ *and* $m \in \mathbb{M}$*, we have*

$$\Pr[(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{SIG.KG}(1^\lambda) : \mathsf{SIG.Ver}(\mathsf{vk}, m, \mathsf{SIG.Sign}(\mathsf{sk}, m)) = 1] = 1.$$

EUF-CMA Security. *For any PPT adversary* $\mathcal{A}$*, the advantage defined as follows is negligible:*

$$\mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathsf{SIG},\mathcal{A}}(\lambda) := \Pr\left[\begin{array}{c} L_{sig} := \emptyset, \\ (\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{SIG.KG}(1^\lambda), \\ (m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}_{sign}}(\mathsf{vk}) \end{array} : \begin{array}{c} \mathsf{SIG.Ver}(\mathsf{vk}, m^*, \sigma^*) = 1 \\ \wedge\ m^* \notin L_{sig} \end{array}\right],$$

where the signing oracle $\mathcal{O}_{sign}$ *is defined as follows:*

Signing Oracle. *When* $\mathcal{A}$ *accesses the signing oracle* $\mathcal{O}_{sign}$ *by making a query* m*, it computes* $\sigma \leftarrow \mathsf{SIG.Sign}(\mathsf{sk}, m)$*, returns* σ *to* $\mathcal{A}$*, and appends* m *to* L_{sig}*.*

2.3 Non-Interactive Zero-Knowledge Proof

We define a non-interactive zero-knowledge proof (or simply NIZK). We require NIZK to satisfy the *simulation soundness*, which is known to be constructed from standard NIZK [21].

Definition 2.3 (NIZK Proof System). *A non-interactive zero-knowledge (NIZK) proof system* NIZK *for a NP relation* $\rho \subseteq \mathcal{X} \times \mathcal{W}$ *consists of the following three PPT algorithms.*

$\mathsf{NIZK.Setup}(1^\lambda) \to \mathsf{crs}$: *The setup algorithm, given a security parameter* 1^λ*, outputs a common reference string* crs*.*

$\mathsf{NIZK.Prove}(\mathsf{crs}, \mathsf{X}, \mathsf{W}) \to \pi$: *The prove algorithm, given a common reference string* crs *and a pair of statement and witness* $(\mathsf{X}, \mathsf{W}) \in \rho$*, outputs a proof* π*.*

$\mathsf{NIZK.Ver}(\mathsf{crs}, \mathsf{X}, \pi) \to 1/0$: *The verify algorithm, given a common reference* crs*, a statement* X*, and a proof* π*, outputs either* 1 *(accept) or* 0 *(reject).*

Furthermore, we require a NIZK proof system to satisfy the following properties.

Correctness. *For all* $\lambda \in \mathbb{N}$*,* $(\mathsf{X}, \mathsf{W}) \in \rho$*, we have*

$$\Pr\left[\begin{array}{c} \mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda), \\ \pi \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, \mathsf{X}, \mathsf{W}) \end{array} : \mathsf{NIZK.Ver}(\mathsf{crs}, \mathsf{X}, \pi) = 1\right] = 1.$$

Zero-Knowledge. *Let* $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$ *be a zero-knowledge simulator for* NIZK. *For any PPT adversary* $\mathcal{A}$, *the advantage defined as follows is negligible:*

$$\mathsf{Adv}^{\mathsf{zk}}_{\mathsf{NIZK},\mathcal{A}}(\lambda) := \left| \begin{array}{l} \Pr[\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda) : \ \mathcal{A}^{\mathcal{P}(\mathsf{crs},\cdot,\cdot)}(\mathsf{crs}) = 1] \\ \quad - \Pr[(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Sim}_0(1^\lambda) : \mathcal{A}^{\mathcal{S}(\mathsf{crs},\mathsf{td},\cdot,\cdot)}(\mathsf{crs}) = 1] \end{array} \right|,$$

where $\mathcal{P}$ *and* $\mathcal{S}$ *are oracles that on input* (X, W) *return* $\perp$ *if* $(\mathsf{X}, \mathsf{W}) \notin \rho$ *and otherwise return* $\mathsf{NIZK.Prove}(\mathsf{crs}, \mathsf{X}, \mathsf{W})$ *and* $\mathsf{Sim}_1(\mathsf{crs}, \mathsf{td}, \mathsf{X})$, *respectively.*

Simulation Soundness. *Let* $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$ *be a zero-knowledge simulator for* NIZK. *For any PPT adversary* $\mathcal{A}$, *the advantage defined as follows is negligible:*

$$\mathsf{Adv}^{\mathsf{sim\text{-}sound}}_{\mathsf{NIZK},\mathcal{A}}(\lambda) := \Pr \left[\begin{array}{c} L_\pi := \emptyset, \\ (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Sim}_0(1^\lambda), \\ (\mathsf{X}, \pi) \leftarrow \mathcal{A}^{\mathcal{S}}(\mathsf{crs}) \end{array} : \begin{array}{c} \mathsf{NIZK.Ver}(\mathsf{crs}, \mathsf{X}, \pi) = 1 \\ \wedge\ (\mathsf{X}, \pi) \notin L_\pi \\ \wedge\ \mathsf{X} \notin L_\rho \end{array} \right],$$

where $\mathcal{S}$ *is a oracle that on input* (X, W) *return* $\pi \leftarrow \mathsf{Sim}_1(\mathsf{crs}, \mathsf{td}, \mathsf{X})$ *and add* (X, π) *to* L_π, *and* L_ρ *is the NP language such that* $L_\rho := \{x \mid \exists w, (x, w) \in \rho\}$.

2.4 Hash Family with Local Opening

Here we recall the definition of a hash family with local opening [36]. The following definition refers to previous works [5, 28].

Definition 2.4. *A hash tree* HT *consists of the following four PPT algorithms.*

$\mathsf{HT.Gen}(1^\lambda) \rightarrow \mathsf{hk}$: *The key generation algorithm, given a security parameter* 1^λ, *outputs a hash key* hk.

$\mathsf{HT.Hash}(\mathsf{hk}, x) \rightarrow \mathsf{d}$: *The hash algorithm, given a hash key* hk *and a message* x, *outputs a hash value* d.

$\mathsf{HT.Open}(\mathsf{hk}, x, i) \rightarrow (b, \pi)$: *The opening algorithm, given a hash key* hk, *an input* x, *and an index* $i \in [N]$, *outputs a bit* b *and an opening* π.

$\mathsf{HT.Ver}(\mathsf{hk}, \mathsf{d}, i, b, \pi) \rightarrow 1/0$: *The verification algorithm, given a hash key* hk, *a hash value* d, *an index* i, *a bit* b, *and an opening* π, *outputs either* 1 *(accept) or* 0 *(reject).*

Furthermore, we require a hash family with local opening to satisfy the following properties.

Completeness. *For all* $\lambda \in \mathbb{N}$, *all* $x \in \{0,1\}^{\mathsf{poly}(\lambda)}$, *and all* $i \in [|x|]$, *there exists a negligible function* negl *such that*

$$\Pr \left[\begin{array}{c} \mathsf{hk} \leftarrow \mathsf{HT.Gen}(1^\lambda), \\ \mathsf{d} \leftarrow \mathsf{HT.Hash}(\mathsf{hk}, x), \\ (b, \pi) \leftarrow \mathsf{HT.Open}(\mathsf{hk}, x, i) \end{array} : \begin{array}{c} \mathsf{HT.Ver}(\mathsf{hk}, \mathsf{d}, i, b, \pi) = 1, \\ \wedge\ b = x_i \end{array} \right] \geq 1 - \mathsf{negl}(\lambda).$$

Efficiency. *In the completeness experiment above, both the running times of* $\mathsf{HT.Gen}$ *and* $\mathsf{HT.Ver}$ *are at most* $\mathsf{poly}(\lambda)$*.*

Collision Resistance w.r.t. Opening. *For any PPT adversary* $\mathcal{A}$*, the advantage defined as follows is negligible:*

$$\mathsf{Adv}^{\mathsf{col}}_{\mathsf{HT},\mathcal{A}}(\lambda) := \Pr\left[\begin{array}{c} \mathsf{hk} \leftarrow \mathsf{HT.Gen}(1^\lambda), \\ (\mathsf{d}, i, \pi_0, \pi_1) \leftarrow \mathcal{A}(\mathsf{hk}) \end{array} : \begin{array}{c} \mathsf{HT.Ver}(\mathsf{hk}, \mathsf{d}, i, 0, \pi_0) = 1, \\ \wedge\ \mathsf{HT.Ver}(\mathsf{hk}, \mathsf{d}, i, 1, \pi_1) = 1 \end{array}\right].$$

2.5 flexible RAM SNARGs

Here we recall the definition of a flexible RAM SNARG proposed by Kalai, Lombardi, Vaikuntanathan, and Wichs [28], which is a RAM delegation scheme [6, 11, 29, 30] in a specific model. In this scheme, we consider a read-only RAM machine that deterministically runs with random access to an external memory of arbitrary size. It allows us to verify whether the RAM machine accepts an input or not with a digest value of the initial external memory.

Definition 2.5. *A flexible RAM SNARG* RamS *for machine* $\mathcal{R}$ *corresponding to a hash family* $\mathsf{HT} = (\mathsf{HT.Gen}, \mathsf{HT.Hash}, \mathsf{HT.Open}, \mathsf{HT.Ver})$ *consists of the following four PPT algorithms.*

$\mathsf{RamS.Setup}(1^\lambda) \to \mathsf{crs}$: *The setup algorithm, given a security parameter* 1^λ*, outputs a common reference string* crs*.*

$\mathsf{RamS.Dig}(\mathsf{hk}, x_{\mathsf{imp}}) \to \mathsf{d}$: *The digest algorithm, given a hash key* hk *generated by* $\mathsf{HT.Gen}(1^\lambda)$ *and a string* x_{imp}*, outputs a digest* d*.*

$\mathsf{RamS.Prove}(\mathsf{crs}, \mathsf{hk}, (x_{\mathsf{imp}}, x_{\mathsf{exp}})) \to (b, \pi)$: *The prove algorithm, given a common reference string* crs*, a hash key* hk*, and a pair of implicit and explicit input* $(x_{\mathsf{imp}}, x_{\mathsf{exp}})$*, outputs a bit* b *(indicating* $\mathcal{R}(x_{\mathsf{imp}}, x_{\mathsf{exp}})$*) and a proof* π*.*

$\mathsf{RamS.Ver}(\mathsf{crs}, \mathsf{hk}, \mathsf{d}, x_{\mathsf{exp}}, b, \pi) \to 1/0$: *The verification algorithm, given a common reference string* crs*, a hash key* hk*, a digest* d*, an explicit input* x_{exp}*, a bit* b*, and a proof* π*, outputs either* 1 *(accept) or* 0 *(reject).*

Furthermore, we require a RAM SNARG to satisfy the following properties.

Completeness. *For all* $\lambda \in \mathbb{N}$*, all RAM machines* $\mathcal{R}$*, all* $x = (x_{\mathsf{imp}}, x_{\mathsf{exp}})$ *such that* $\mathcal{R}(x)$ *accepts (i.e.,* $\mathcal{R}(x) = 1$*), there exists a negligible function* negl *such that*

$$\Pr\left[\begin{array}{c} \mathsf{crs} \leftarrow \mathsf{RamS.Setup}(1^\lambda), \\ \mathsf{d} \leftarrow \mathsf{RamS.Dig}(\mathsf{hk}, x_{\mathsf{imp}}), \\ (b, \pi) \leftarrow \mathsf{RamS.Prove}(\mathsf{crs}, \mathsf{hk}, x) \end{array} : \begin{array}{c} \mathsf{RamS.Ver}(\mathsf{crs}, \mathsf{hk}, \mathsf{d}, x_{\mathsf{exp}}, b, \pi) = 1 \\ \wedge\ b = \mathcal{R}(x) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

Efficiency. *In the completeness experiment above, the running time of* $\mathsf{RamS.Setup}$ *is at most* $\mathsf{poly}(\lambda, |x_{\mathsf{exp}}|, \log |x_{\mathsf{imp}}|)$*, and the length of a proof* π *is at most* $\mathsf{poly}(\lambda, |x_{\mathsf{exp}}|, \log |x_{\mathsf{imp}}|)$*.*

Soundness. *For any PPT adversary $\mathcal{A}$, the advantage defined as follows is negligible:*

$$\mathsf{Adv}^{\mathsf{sound}}_{\mathsf{RamS},\mathcal{A}}(\lambda) := \Pr\left[\begin{array}{c} \mathsf{crs} \leftarrow \mathsf{RamS.Setup}(1^\lambda), \\ (x = (x_{\mathsf{imp}}, x_{\mathsf{exp}}), b, \pi) \leftarrow \mathcal{A}(\mathsf{crs}), \\ \mathsf{d} \leftarrow \mathsf{RamS.Dig}(\mathsf{hk}, x_{\mathsf{imp}}) \end{array} : \begin{array}{c} \mathsf{RamS.Ver}(\mathsf{crs}, \mathsf{hk}, \mathsf{d}, x_{\mathsf{exp}}, b, \pi) = 1 \\ \wedge\ b \neq \mathcal{R}(x) \end{array}\right].$$

Remark 2.1. Kalai, Lombardi, Vaikuntanathan, and Wichs [28] proposed a RAM SNARG scheme that satisfy a stronger definition of soundness, *partial input soundness*, but a weaker definition defined as above is enough for our construction in the following. We also note that existing constructions [11,30] of a RAM delegation satisfy the weaker soundness.

Remark 2.2. In the above definition, we omit a time bound T from an input to the setup algorithm since we can set T as at most 2^λ in the existing constructions [11,28,30].

2.6 Attribute-Based Signature

Here we recall the definition of a attribute-based signature scheme from [35,41].

Definition 2.6. *An attribute-based signature* ABS *scheme consists of the following four PPT algorithms.*

$\mathsf{ABS.Setup}(1^\lambda, 1^\ell) \to (\mathsf{pp}, \mathsf{msk})$: *The setup algorithm, given a security parameter 1^λ and an attribute length 1^ℓ, outputs a public parameter* pp *and a master secret key* msk.

$\mathsf{ABS.KG}(\mathsf{msk}, x) \to \mathsf{sk}_x$: *The key generation algorithm, given a master secret key* msk *and an attribute $x \in \{0,1\}^\ell$, outputs a user secret key* sk_x.

$\mathsf{ABS.Sign}(\mathsf{pp}, \mathsf{sk}_x, x, C, m) \to \Sigma$: *The signing algorithm, given a public parameter* pp*, a user secret key* sk_x*, an attribute x, a policy C, and a message m, outputs a signature Σ.*

$\mathsf{ABS.Ver}(\mathsf{pp}, C, m, \Sigma) \to 1/0$: *The verification algorithm, given a public parameter* pp*, a policy C, a message m, and a signature Σ, outputs either 1 (accept) or 0 (reject).*

Furthermore, we require an attribute-based signature scheme to satisfy the following properties.

Correctness. *For all $\lambda \in \mathbb{N}$, all $\ell \in \mathsf{poly}(\lambda)$, all $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{ABS.Setup}(1^\lambda, 1^\ell)$, all attributes x, all $\mathsf{sk}_x \leftarrow \mathsf{ABS.KG}(\mathsf{msk}, x)$, all policies C satisfying $C(x) = 1$, all messages m, and all $\Sigma \leftarrow \mathsf{ABS.Sign}(\mathsf{pp}, \mathsf{sk}_x, x, C, m)$, we have $\mathsf{ABS.Ver}(\mathsf{pp}, C, m, \Sigma) = 1$.*

```
Experiment $\mathsf{Expt}^{\mathsf{unf}}_{\mathsf{ABS},\mathcal{A}}(\lambda)$
  $L_{corr}, L_{sig}, L_{key} \leftarrow \emptyset$
  $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{ABS.Setup}(1^\lambda, 1^\ell)$
  $(C^*, m^*, \Sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}_{sig}, \mathcal{O}_{corr}}(\mathsf{pp})$
  if $\exists\, x \in L_{corr}$ s.t. $C^*(x) = 1$ then return $0$
  if $(C^*, m^*) \in L_{sig}$ then return $0$
  if $\mathsf{ABS.Ver}(\mathsf{pp}, C^*, m^*, \Sigma^*) = 1$ then return $1$
  return $0$

Oracle $\mathcal{O}_{sig}(x, C, m)$
  if $C(x) = 0$ then return $\perp$
  $L_{sig} \leftarrow L_{sig} \cup \{(C, m)\}$
  if $(x, \cdot) \notin L_{key}$ then
      $\mathsf{sk}_x \leftarrow \mathsf{ABS.KG}(\mathsf{msk}, x)$
      $L_{key} \leftarrow L_{key} \cup \{(x, \mathsf{sk}_x)\}$
  otherwise find $(x, \mathsf{sk}_x) \in L_{key}$
  $\Sigma \leftarrow \mathsf{ABS.Sign}(\mathsf{pp}, \mathsf{sk}_x, x, C, m)$
  return $\Sigma$

Oracle $\mathcal{O}_{corr}(x)$
  $L_{corr} \leftarrow L_{corr} \cup \{x\}$
  if $(x, \mathsf{sk}_x) \in L_{key}$ then return $\mathsf{sk}_x$
  $\mathsf{sk}_x \leftarrow \mathsf{ABS.KG}(\mathsf{msk}, x)$
  $L_{key} \leftarrow L_{key} \cup \{(x, \mathsf{sk}_x)\}$
  return $\mathsf{sk}_x$
```

Fig. 1: The experiment for defining *unforgeability* of ABS.

Privacy. *For any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, the advantage defined as follows is negligible:*

$$\mathsf{Adv}^{\mathsf{priv}}_{\mathsf{ABS},\mathcal{A}}(\lambda) := \left| \Pr\left[\begin{array}{c} b \leftarrow \{0,1\}, \\ (\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{ABS.Setup}(1^\lambda, 1^\ell), \\ (\mathsf{st}, x_0, x_1, C, m) \leftarrow \mathcal{A}_1(\mathsf{pp}, \mathsf{msk}), \\ \forall i \in \{0,1\}, \mathsf{sk}_i \leftarrow \mathsf{ABS.KG}(\mathsf{msk}, x_i)\ , \\ \Sigma \leftarrow \mathsf{ABS.Sign}(\mathsf{pp}, \mathsf{sk}_b, x_b, C, m), \\ b' \leftarrow \mathcal{A}_2(\mathsf{st}, \mathsf{sk}_0, \mathsf{sk}_1, \Sigma) \end{array} : b = b' \right] - \frac{1}{2} \right|,$$

where $\mathcal{A}_1$ is required to output x_0, x_1, and C satisfying $C(x_0) = C(x_1) = 1$.

Unforgeability. *For any PPT adversary $\mathcal{A}$, the advantage $\mathsf{Adv}^{\mathsf{unf}}_{\mathsf{ABS},\mathcal{A}}(\lambda) := \Pr[\mathsf{Expt}^{\mathsf{unf}}_{\mathsf{ABS},\mathcal{A}}(\lambda) = 1]$ is negligible, where the experiment is defined in Figure 1.*

3 Circuit SNARGs

In this section, we introduce a new notion, *Circuit SNARG*, which will be a useful tool for describing our construction in Section 4. Intuitively, this primitive allows us to verify $C(x) = b \in \{0,1\}$ with digest values of an input x and a circuit C, and a succinct proof π.

Definition 3.1. *A Circuit SNARG* CirS *for circuits $\mathcal{C}$ consists of the following five PPT algorithms.*

$\mathsf{CirS.Setup}(1^\lambda, 1^\ell) \to \mathsf{crs}$: *The setup algorithm, given a security parameter* 1^λ *and an input length* 1^ℓ*, outputs a common reference string* crs.

$\mathsf{CirS.DStr}(\mathsf{crs}, x) \to \mathsf{d}_x$: *The string digest algorithm, given a common reference string* crs *and a string* $x \in \{0,1\}^\ell$*, outputs a string digest* d_x.

$\mathsf{CirS.DCir}(\mathsf{crs}, C) \to \mathsf{d}_C$: *The circuit digest algorithm, given a common reference string* crs *and a circuit* C*, outputs a circuit digest* d_C.

$\mathsf{CirS.Prove}(\mathsf{crs}, x, C) \to (b, \pi)$: *The prove algorithm, given a common reference string* crs*, a string* x*, and a circuit* C*, outputs a bit* b *a proof* π.

$\mathsf{CirS.Ver}(\mathsf{crs}, \mathsf{d}_x, \mathsf{d}_C, b, \pi) \to 1/0$: *The verification algorithm, given a common reference string* crs*, a string digest* d_x*, a circuit digest* d_C*, and a proof* π*, outputs either* 1 *(accept) or* 0 *(reject).*

Furthermore, we require a Circuit SNARG to satisfy the following properties.

Completeness. *For all* $\lambda \in \mathbb{N}$*, all strings* $x \in \{0,1\}^\ell$*, all circuits* C *such that* $C(x) = b$*, we have*

$$\Pr\left[\begin{array}{c} \mathsf{crs} \leftarrow \mathsf{CirS.Setup}(1^\lambda, 1^\ell), \\ \mathsf{d}_x \leftarrow \mathsf{CirS.DStr}(\mathsf{crs}, x), \\ \mathsf{d}_C \leftarrow \mathsf{CirS.DCir}(\mathsf{crs}, C), \\ (b, \pi) \leftarrow \mathsf{CirS.Prove}(\mathsf{crs}, x, C) \end{array} : \mathsf{CirS.Ver}(\mathsf{crs}, \mathsf{d}_x, \mathsf{d}_C, b, \pi) = 1 \right] = 1.$$

Efficiency. *In the completeness experiment above, the running time of* $\mathsf{CirS.Ver}$ *is at most* $\mathsf{poly}(\lambda, \log(|x| + |C|))$*, and the lengths of both digests* d_x *and* d_C *are* $O(\lambda)$*, and the length of a proof* π *is at most* $\mathsf{poly}(\lambda, \log(|x| + |C|))$.

Collision Resistance w.r.t. the String Digest. *For any PPT adversary* $\mathcal{A}$*, the advantages defined as follows is negligible:*

$$\mathsf{Adv}^{\mathsf{col\text{-}str}}_{\mathsf{CirS},\mathcal{A}}(\lambda) := \Pr\left[\begin{array}{c} \mathsf{crs} \leftarrow \mathsf{CirS.Setup}(1^\lambda, 1^\ell), \\ (x_0, x_1) \leftarrow \mathcal{A}(\mathsf{crs}), \\ \mathsf{d}_{x_0} \leftarrow \mathsf{CirS.DStr}(\mathsf{crs}, x_0), \\ \mathsf{d}_{x_1} \leftarrow \mathsf{CirS.DStr}(\mathsf{crs}, x_1) \end{array} : \mathsf{d}_{x_0} \neq \mathsf{d}_{x_1} \right].$$

Collision Resistance w.r.t. the Circuit Digest. *For any PPT adversary* $\mathcal{A}$*, the advantages defined as follows is negligible:*

$$\mathsf{Adv}^{\mathsf{col\text{-}cir}}_{\mathsf{CirS},\mathcal{A}}(\lambda) := \Pr\left[\begin{array}{c} \mathsf{crs} \leftarrow \mathsf{CirS.Setup}(1^\lambda, 1^\ell), \\ (C_0, C_1) \leftarrow \mathcal{A}(\mathsf{crs}), \\ \mathsf{d}_{C_0} \leftarrow \mathsf{CirS.DCir}(\mathsf{crs}, C_0), \\ \mathsf{d}_{C_1} \leftarrow \mathsf{CirS.DCir}(\mathsf{crs}, C_1) \end{array} : \mathsf{d}_{C_0} \neq \mathsf{d}_{C_1} \right].$$

Soundness. *For any PPT adversary $\mathcal{A}$, the advantage defined as follows is negligible:*

$$\mathsf{Adv}^{\mathsf{sound}}_{\mathsf{CirS},\mathcal{A}}(\lambda) := \Pr\left[\begin{array}{l} \mathsf{crs} \leftarrow \mathsf{CirS.Setup}(1^\lambda, 1^\ell), \\ (x^*, C^*, b^*, \pi^*) \leftarrow \mathcal{A}(\mathsf{crs}), \\ \mathsf{d}_{x^*} \leftarrow \mathsf{CirS.DStr}(\mathsf{crs}, x^*), \\ \mathsf{d}_{C^*} \leftarrow \mathsf{CirS.DCir}(\mathsf{crs}, C^*) \end{array} : \begin{array}{c} \mathsf{CirS.Ver}(\mathsf{crs}, \mathsf{d}_{x^*}, \mathsf{d}_{C^*}, b^*, \pi^*) = 1 \\ \wedge\ b^* \neq C^*(x^*) \end{array}\right].$$

3.1 Construction From flexible RAM SNARGs

We propose a construction of a Circuit SNARG for a circuit class $\mathcal{C}$ with input length ℓ. Our construction is directly from a flexible RAM SNARG RamS corresponding to a hash family HT'. Below we define a hash family HT' and a RAM machine $\mathcal{R}'$.

Hash Family HT'. Let $\mathsf{HT} = (\mathsf{HT.Gen}, \mathsf{HT.Hash}, \mathsf{HT.Open}, \mathsf{HT.Ver})$ be a hash family (Definition 2.4). We use a hash family $\mathsf{HT}' = (\mathsf{HT}'.\mathsf{Gen}, \mathsf{HT}'.\mathsf{Hash}, \mathsf{HT}'.\mathsf{Open}, \mathsf{HT}'.\mathsf{Ver})$ defined as follows.

$\mathsf{HT}'.\mathsf{Gen}(1^\lambda)$: It computes $\mathsf{hk} \leftarrow \mathsf{HT.Gen}(1^\lambda)$ and outputs $\mathsf{hk}' = \mathsf{hk}$.

$\mathsf{HT}'.\mathsf{Hash}(\mathsf{hk}', x)$: It first parses $x \in \{0,1\}^N$ as (x_0, x_1), where $x_0 \in \{0,1\}^\ell$ and $x_1 \in \{0,1\}^{N-\ell}$. Then, it computes $\mathsf{d}_{x_0} \leftarrow \mathsf{HT.Hash}(\mathsf{hk}, x_0)$ and $\mathsf{d}_{x_1} \leftarrow \mathsf{HT.Hash}(\mathsf{hk}, x_1)$. Finally, it outputs $\mathsf{d} = (\mathsf{d}_{x_0}, \mathsf{d}_{x_1})$.

$\mathsf{HT}'.\mathsf{Open}(\mathsf{hk}', x, j)$: If $j \in [\ell]$, then it sets $\rho \leftarrow \mathsf{HT.Open}(\mathsf{hk}, x_0, j)$. Otherwise, it computes $\rho \leftarrow \mathsf{HT.Open}(\mathsf{hk}, x_1, j - \ell)$. Finally, it outputs ρ.

$\mathsf{HT}'.\mathsf{Ver}(\mathsf{hk}', \mathsf{d}, j, b, \rho)$: It first parses d as $(\mathsf{d}_{x_0}, \mathsf{d}_{x_1})$. If $j \in [\ell]$, then it outputs $\mathsf{HT.Ver}(\mathsf{hk}, \mathsf{d}_{x_0}, j, b, \rho)$. Otherwise, it outputs $\mathsf{HT.Ver}(\mathsf{hk}, \mathsf{d}_{x_1}, j, b, \rho)$.

It is easy to see that the above hash family HT' satisfies all properties in Definition 2.4.

Theorem 3.1. *If HT is a hash family with local opening, then the above HT' is a hash family with local opening.*

flexible RAM SNARG RamS for $\mathcal{R}$. We use a flexible RAM SNARG $\mathsf{RamS} = (\mathsf{RamS.Setup}, \mathsf{RamS.Dig}, \mathsf{RamS.Prove}, \mathsf{RamS.Ver})$ for a RAM machine $\mathcal{R}$, which is corresponding to the above hash family HT'. A RAM machine $\mathcal{R}$ takes as input $(x_{\mathsf{imp}}, x_{\mathsf{exp}}) = (x||C, \bot)$ and outputs 1 if and only if $C(x) = 1$, where $x \in \{0,1\}^\ell$ and $C \in \{0,1\}^{N-\ell}$.

Our Construction. We show the construction of Circuit SNARG. We note that the $\mathsf{RamS.Dig}$ algorithm is deterministic and fixed by the corresponding HT'. Let hk' be a hash key generated by $\mathsf{HT}'.\mathsf{Gen}(1^\lambda)$. From the construction of HT', two digests $(\mathsf{d}_{x_0}, \mathsf{d}_{x_1})$ are separately computable. More precisely, if a string $x_{\mathsf{imp}} \in \{0,1\}^N$ stored in the random access memory can be divided into two strings $x \in \{0,1\}^{\ell}$ and $C \in \{0,1\}^{N-\ell}$, then the digest value $\mathsf{d} = (\mathsf{d}_x, \mathsf{d}_C) \leftarrow \mathsf{RamS.Dig}(\mathsf{hk}', x_{\mathsf{imp}} = x||C)$ can be computed separately, i.e., there exist two algorithms $\mathsf{RamS.Dig}_1$ and $\mathsf{RamS.Dig}_2$ such that $\mathsf{d}_x \leftarrow \mathsf{RamS.Dig}_1(\mathsf{hk}', x)$ and $\mathsf{d}_C \leftarrow \mathsf{RamS.Dig}_2(\mathsf{hk}', C)$, respectively.

$\mathsf{CirS.Setup}(1^{\lambda})$: It generates a hash key $\mathsf{hk}' \leftarrow \mathsf{HT}'.\mathsf{Gen}(1^{\lambda})$ and a common reference string for RAM SNARG $\mathsf{crs}_{\mathcal{R}} \leftarrow \mathsf{RamS.Setup}(1^{\lambda})$ and outputs $\mathsf{crs} = (\mathsf{hk}', \mathsf{crs}_{\mathcal{R}})$.

$\mathsf{CirS.DStr}(\mathsf{crs}, x)$: It computes $\mathsf{d}_x \leftarrow \mathsf{RamS.Dig}_1(\mathsf{hk}', x)$ and outputs d_x.

$\mathsf{CirS.DStr}(\mathsf{crs}, C)$: It computes $\mathsf{d}_C \leftarrow \mathsf{RamS.Dig}_2(\mathsf{hk}', C)$ and outputs d_C.

$\mathsf{CirS.Prove}(\mathsf{crs}, x, C)$: It generates a RAM SNARG proof $(b, \pi) \leftarrow \mathsf{RamS.Prove}(\mathsf{crs}_{\mathcal{R}}, \mathsf{hk}', (x||C, \perp))$ and outputs (b, π).

$\mathsf{CirS.Ver}(\mathsf{crs}, \mathsf{d}_x, \mathsf{d}_C, b, \pi) \to 1/0$: It computes $b' \leftarrow \mathsf{RamS.Ver}(\mathsf{crs}, \mathsf{hk}', (\mathsf{d}_x, \mathsf{d}_C), \perp, b, \pi)$ and outputs b'.

It is easy to see that the above construction satisfies completeness and efficiency requirement if the RAM SNARG satisfies completeness and is efficient.

3.2 Security Analysis

In this section, we provide a security proof to show that our construction of a Circuit SNARG satisfies the collision resistance w.r.t. the circuit digest and the soundness. Although we omit a proof to show that our construction satisfies the collision resistance w.r.t. the string digest, it is easy to see that ours also satisfies it in the same way as proof of Theorem 3.2.

Theorem 3.2. *If the hash family* HT' *is collision-resistant w.r.t. opening, then the above Circuit SNARG is collision-resistant w.r.t. the circuit digest.*

Proof. Assume that there exists a PPT adversary $\mathcal{A}$ which breaks the collision resistance w.r.t. the circuit digest of the Circuit SNARG with non-negligible probability. Then, we can construct another PPT adversary $\mathcal{B}$ that breaks the collision-resistant w.r.t. opening of the hash family with the same probability. The description of $\mathcal{B}$ is as follows.

- $\mathcal{B}$ initially receives hk', computes $\mathsf{crs}_{\mathcal{R}} \leftarrow \mathsf{RamS.Setup}(1^{\lambda})$, sets $\mathsf{crs} := (\mathsf{hk}', \mathsf{crs}_{\mathcal{R}})$ and runs $\mathcal{A}(\mathsf{crs})$.
- When $\mathcal{A}$ outputs (C_0^*, C_1^*) and terminates, $\mathcal{B}$ finds an index i such that $C_{0,i}^* \neq C_{1,i}^*$, where $C_{b,i}^*$ denotes the i-th bit of C_b^* for $b \in \{0,1\}$. Then, $\mathcal{B}$ computes $\mathsf{d} \leftarrow \mathsf{HT}'.\mathsf{Hash}(\mathsf{hk}', C_0^*)$, $\pi_0 \leftarrow \mathsf{HT}'.\mathsf{Open}(\mathsf{hk}', C_0^*, i)$, and $\pi_1 \leftarrow \mathsf{HT}'.\mathsf{Open}(\mathsf{hk}', C_1^*, i)$, outputs $(\mathsf{d}, i, \pi_0, \pi_1)$ if $C_{0,i}^* = 0$; otherwise, it outputs $(\mathsf{d}, i, \pi_1, \pi_0)$.

The above completes the description of $\mathcal{B}$. Since $\mathcal{A}$ breaks the collision resistance w.r.t. the circuit digest, we have $\mathsf{d} = \mathsf{HT}'.\mathsf{Hash}(\mathsf{hk}', C_0^*) = \mathsf{HT}'.\mathsf{Hash}(\mathsf{hk}', C_1^*)$. In addition, since each opening is correctly generated, we have $\mathsf{HT}'.\mathsf{Ver}(\mathsf{hk}', \mathsf{d}, i, 0, \pi_0) =$

1 and $\mathsf{HT}'.\mathsf{Ver}(\mathsf{hk}', \mathsf{d}, i, 1, \pi_1) = 1$ if $C^*_{0,i} = 0$; otherwise, we have $\mathsf{HT}'.\mathsf{Ver}(\mathsf{hk}', \mathsf{d}, i, 0, \pi_1) = 1$ and $\mathsf{HT}'.\mathsf{Ver}(\mathsf{hk}', \mathsf{d}, i, 1, \pi_0) = 1$. In both cases, $\mathcal{B}$ breaks the collision-resistant w.r.t. opening of the hash family. Therefore, we have $\mathsf{Adv}^{\mathsf{col\text{-}cir}}_{\mathsf{CirS},\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathsf{col}}_{\mathsf{HT}',\mathcal{B}}(\lambda)$. □ (**Theorem 3.2**)

Theorem 3.3. *If the flexible RAM SNARG* RamS *is sound, then the above Circuit SNARG is sound.*

Proof. Assume that there exists a PPT adversary $\mathcal{A}$ which breaks the soundness of the Circuit SNARG with non-negligible probability. Then, we can construct another PPT adversary $\mathcal{B}$ that breaks the soundness of the flexible RAM SNARG with the same probability. Let us fix a hash key hk' generated by $\mathsf{HT}'.\mathsf{Gen}(1^\lambda)$ corresponding to the flexible RAM SNARG. The description of $\mathcal{B}$ is as follows.

- $\mathcal{B}$ initially receives $\mathsf{crs}_\mathcal{R}$, sets $\mathsf{crs} := (\mathsf{hk}', \mathsf{crs}_\mathcal{R})$ and runs $\mathcal{A}(\mathsf{crs})$.
- When $\mathcal{A}$ outputs (x^*, C^*, b^*, π^*) and terminates, $\mathcal{B}$ sets $x^*_{\mathsf{imp}} = x^*||C^*$ and $x^*_{\mathsf{exp}} = \perp$, outputs $((x^*_{\mathsf{imp}}, x^*_{\mathsf{exp}}), b^*, \pi^*)$, and terminates.

The above completes the description of $\mathcal{B}$. Let $\mathsf{d}_{x^*} = \mathsf{RamS}.\mathsf{Dig}_1(\mathsf{hk}', x^*)$ and $\mathsf{d}_{C^*} = \mathsf{RamS}.\mathsf{Dig}_2(\mathsf{hk}', C^*)$. Since $\mathcal{A}$ breaks the soundness of the Circuit SNARG, we have $\mathsf{CirS}.\mathsf{Ver}(\mathsf{crs}, \mathsf{d}_{x^*}, \mathsf{d}_{C^*}, b^*, \pi^*) = 1$ and $b^* \neq C^*(x^*)$. Thus, we now have $b^* = \mathsf{RamS}.\mathsf{Ver}(\mathsf{crs}_\mathcal{R}, \mathsf{hk}', (\mathsf{d}_{x^*}, \mathsf{d}_{C^*}), \perp, b^*, \pi^*)$ while $b^* \neq \mathcal{R}(x^*||C^*, \perp)$ due to the definition of the RAM machine $\mathcal{R}$. Therefore, we have $\mathsf{Adv}^{\mathsf{sound}}_{\mathsf{CirS},\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathsf{sound}}_{\mathsf{RamS},\mathcal{B}}(\lambda)$. □ (**Theorem 3.3**)

4 Attribute-Based Signatures for General Circuits from Circuit Delegations

In this section, we provide a construction of an attribute-based signature scheme for every polynomial-size circuits with input length ℓ. Before showing our detailed construction, we provide an intuition of the construction. To issue a user signing key, the key issuer computes a digest d_x of the user's attribute x and signs the digest. Each user receives a signing key that consists of an attribute digest d_x and its signature σ_{d_x} and generates a proof π to show that his attribute satisfies some policy C. In addition, for completing security proof, we require each user to encrypt a witness consisting of a message m to be signed, the digest d_x, its signature σ_{d_x}, and the proof π, and include a calculated ciphertext to a signature. It also computes a NIZK proof to show that it has (i) an attribute x such that $C(x) = 1$, (ii) a valid signature of its digest value, and (iii) a ciphertext of the witness.

4.1 Our Construction

Here we provide a construction based on following building blocks:

- a PKE scheme $\mathsf{PKE} = (\mathsf{PKE.KG}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$;

– a signature scheme $\mathsf{SIG} = (\mathsf{SIG.KG}, \mathsf{SIG.Sign}, \mathsf{SIG.Ver})$;
– a NIZK $\mathsf{NIZK} = (\mathsf{NIZK.Setup}, \mathsf{NIZK.Prove}, \mathsf{NIZK.Ver})$ where the NIZK relation ρ is defined as follows:

$$\begin{aligned}\rho := \{((\mathsf{ek}, \mathsf{vk}, \mathsf{crs}_{\mathsf{CirS}}, \mathsf{d}_C, m, \mathsf{ctx}), (\mathsf{d}_x, \sigma_{\mathsf{d}_x}, \pi, \mathsf{rand})) \mid \\ \mathsf{CirS.Ver}(\mathsf{crs}_{\mathsf{CirS}}, \mathsf{d}_x, \mathsf{d}_C, 1, \pi) = 1 \\ \wedge\ \mathsf{SIG.Ver}(\mathsf{vk}, \mathsf{d}_x, \sigma_{\mathsf{d}_x}) = 1 \\ \wedge\ \mathsf{PKE.Enc}(\mathsf{ek}, (m, \mathsf{d}_x, \sigma_{\mathsf{d}_x}, \pi); \mathsf{rand}) = \mathsf{ctx}\};\end{aligned}$$

– a Circuit SNARG $\mathsf{CirS} = (\mathsf{CirS.Setup}, \mathsf{CirS.DStr}, \mathsf{CirS.DCir}, \mathsf{CirS.Prove}, \mathsf{CirS.Ver})$.

Our Construction. Our construction is as follows.

$\mathsf{ABS.Setup}(1^\lambda, 1^\ell)$: It computes the followings:
- a key pair for PKE $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{PKE}(1^\lambda)$,
- a key pair for SIG $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{SIG.KG}(1^\lambda)$,
- a common reference string for NIZK $\mathsf{crs}_{\mathsf{NIZK}} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$, and
- a common reference string for Circuit SNARG $\mathsf{crs}_{\mathsf{CirS}} \leftarrow \mathsf{CirS.Setup}(1^\lambda, 1^\ell)$.

Then, it outputs $\mathsf{pp} := (\mathsf{ek}, \mathsf{vk}, \mathsf{crs}_{\mathsf{NIZK}}, \mathsf{crs}_{\mathsf{CirS}})$ and $\mathsf{msk} := (\mathsf{sk}, \mathsf{crs}_{\mathsf{CirS}})$.

$\mathsf{ABS.KG}(\mathsf{msk}, x)$: It computes in the following steps:
1. Parse $\mathsf{msk} = (\mathsf{sk}, \mathsf{crs}_{\mathsf{CirS}})$;
2. Compute $\mathsf{d}_x \leftarrow \mathsf{CirS.DStr}(\mathsf{crs}_{\mathsf{CirS}}, x)$ and $\sigma_{\mathsf{d}_x} \leftarrow \mathsf{SIG.Sign}(\mathsf{sk}, \mathsf{d}_x)$.

Then, it outputs $\mathsf{sk}_x := (\mathsf{d}_x, \sigma_{\mathsf{d}_x})$.

$\mathsf{ABS.Sign}(\mathsf{pp}, \mathsf{sk}_x, x, C, m)$: It computes in the following steps:
1. Parse $\mathsf{pp} = (\mathsf{ek}, \mathsf{vk}, \mathsf{crs}_{\mathsf{NIZK}}, \mathsf{crs}_{\mathsf{CirS}})$ and $\mathsf{sk}_x = (\mathsf{d}_x, \sigma_{\mathsf{d}_x})$;
2. Compute $\mathsf{d}_C \leftarrow \mathsf{CirS.DCir}(\mathsf{crs}_{\mathsf{CirS}}, C)$ and $(b, \pi) \leftarrow \mathsf{CirS.Prove}(\mathsf{crs}_{\mathsf{CirS}}, x, C)$;
3. Randomly choose $\mathsf{rand} \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$ and compute $\mathsf{ctx} \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}, (m, \mathsf{d}_x, \sigma_{\mathsf{d}_x}, \pi); \mathsf{rand})$;
4. Compute $\Pi \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_{\mathsf{NIZK}}, (\mathsf{ek}, \mathsf{vk}, \mathsf{crs}_{\mathsf{CirS}}, \mathsf{d}_C, m, \mathsf{ctx}), (\mathsf{d}_x, \sigma_{\mathsf{d}_x}, \pi, \mathsf{rand}))$.

Finally, it outputs $\Sigma := (\mathsf{ctx}, \Pi)$.

$\mathsf{ABS.Ver}(\mathsf{pp}, C, m, \Sigma) \to 1/0$: It computes in the following steps:
1. Parse $\mathsf{pp} = (\mathsf{ek}, \mathsf{vk}, \mathsf{crs}_{\mathsf{NIZK}}, \mathsf{crs}_{\mathsf{CirS}})$ and $\Sigma = (\mathsf{ctx}, \Pi)$;
2. Compute $\mathsf{d}_C \leftarrow \mathsf{CirS.DCir}(\mathsf{crs}_{\mathsf{CirS}}, C)$ and $b \leftarrow \mathsf{NIZK.Ver}(\mathsf{crs}_{\mathsf{NIZK}}, (\mathsf{ek}, \mathsf{vk}, \mathsf{crs}_{\mathsf{CirS}}, \mathsf{d}_C, m, \mathsf{ctx}), \Pi)$.

Finally, it outputs 1 if $b = 1$; otherwise, it outputs 0.

It is easy to see that the above construction satisfies the correctness if each building block is correct.

Efficiency. We show that the above construction achieves optimal parameter sizes as follows: if the underlying Circuit SNARG is efficient,

– the length of the public parameter is $\mathsf{poly}(\lambda, \log(|x| + |C|))$;

– the length of the key is $\mathsf{poly}(\lambda)$ since $|\mathsf{d}_x| = O(\lambda)$;
– the length of the signature is $\mathsf{poly}(\lambda, \log(|x|+|C|))$ since we have $|\mathsf{d}_x|, |\mathsf{d}_C| = O(\lambda)$ and $|\pi| = \mathsf{poly}(\lambda, \log(|x|+|C|))$, so the verification circuit of NIZK and its proof are of lengths $\mathsf{poly}(\lambda, \log(|x|+|C|))$.

4.2 Security Analysis

Here we provide security proofs to show that our construction satisfies the privacy and unforgeability in Theorem 4.1 and Theorem 4.2, respectively. In the following proofs of theorems and lemmata, we will use the underline to explicitly show the parts where each reduction accesses to its challenge oracle for clarity.

Theorem 4.1. *If the PKE scheme* PKE *is IND-CPA secure and the NIZK proof system* NIZK *is zero-knowledge, then the above ABS scheme is private.*

Proof. Let us fix a PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ attacking the privacy of the ABS, the security parameter λ, and the attribute length ℓ. The attack game used to define the privacy is in Definition 2.6. We define two games as follows:

Game_0 : This game is the original attack game.

Game_1 : This game is the game identical with Game_0 except that we use modified algorithms, in which some steps are replaced as follows:
- In the $\mathsf{ABS.Setup}(1^\lambda, 1^\ell)$ algorithm, $\mathsf{crs}_{\mathsf{NIZK}} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$ is replaced by $(\widetilde{\mathsf{crs}}, \mathsf{td}) \leftarrow \mathsf{Sim}_0(1^\lambda)$.
- In the $\mathsf{ABS.Sign}(\mathsf{pp}, \mathsf{sk} = (\mathsf{d}_x, \sigma), x, C, m)$ algorithm, $\mathit{\Pi} \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_{\mathsf{NIZK}}, (\mathsf{ek}, \mathsf{vk}, \mathsf{crs}_{\mathsf{CirS}}, \mathsf{d}_C, m, \mathsf{ctx}), (\mathsf{d}_x, \sigma, \pi, \mathsf{rand}))$ is replaced by $\widetilde{\mathit{\Pi}} \leftarrow \mathsf{Sim}_1(\widetilde{\mathsf{crs}}, \mathsf{td}, (\mathsf{ek}, \mathsf{vk}, \mathsf{crs}_{\mathsf{CirS}}, \mathsf{d}_C, m, \mathsf{ctx}))$.

For $i = 0, 1$, let T_i be the event that $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ wins the privacy experiment in the game Game_i. We will show that the probability $|\Pr[T_0] - \Pr[T_1]| \leq \mathsf{negl}(\lambda)$ and $\Pr[T_1] \leq \mathsf{negl}(\lambda)$ in the following lemmas.

We first show to have $|\Pr[T_0] - \Pr[T_1]| \leq \mathsf{negl}(\lambda)$. Intuitively, any difference between these two games Game_0 and Game_1 yields a PPT algorithm that distinguishes the real proof from the simulated proof.

Lemma 4.1. *There exists a PPT algorithm $\mathcal{B}$ such that*

$$|\Pr[T_0] - \Pr[T_1]| = \mathsf{Adv}^{\mathsf{zk}}_{\mathsf{NIZK},\mathcal{B}}(\lambda).$$

Proof. Assume that there exists a PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ which makes the probability $|\Pr[T_0] - \Pr[T_1]|$ non-negligible. Then, we can construct another PPT adversary $\mathcal{B}$ that breaks the zero-knowledge property of $\mathit{\Pi}$ with non-negligible probability, which implies the lemma. The description of $\mathcal{B}$ is as follows.

1. $\mathcal{B}$ initially receives crs, computes $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{PKE}(1^\lambda)$, $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{SIG.KG}(1^\lambda)$, and $\mathsf{crs}_{\mathsf{CirS}} \leftarrow \mathsf{CirS.Setup}(1^\lambda, 1^\ell)$, sets $\mathsf{pp} := (\mathsf{ek}, \mathsf{vk}, \mathsf{crs}, \mathsf{crs}_{\mathsf{CirS}})$ and $\mathsf{msk} := (\mathsf{sk}, \mathsf{crs}_{\mathsf{CirS}})$, and runs $\mathcal{A}_1(\mathsf{pp}, \mathsf{msk})$.

2. When $\mathcal{A}_1$ outputs $(\mathsf{st}, x_0, x_1, C, m)$ and terminates, $\mathcal{B}$ randomly chooses $b \leftarrow \{0,1\}$ and proceeds as follows, where $\star$ denotes that some value exists but is being ignored:
   (i) Compute $\mathsf{d}_{x_i} \leftarrow \mathsf{CirS.DStr}(\mathsf{crs}_{\mathsf{CirS}}, x_i)$ for both $i \in \{0,1\}$, $\mathsf{d}_C \leftarrow \mathsf{CirS.DCir}(\mathsf{crs}_{\mathsf{CirS}}, C)$, and $(\star, \pi_b) \leftarrow \mathsf{CirS.Prove}(\mathsf{crs}_{\mathsf{CirS}}, x_b, C)$;
   (ii) Compute $\sigma_{\mathsf{d}_{x_i}} \leftarrow \mathsf{SIG.Sign}(\mathsf{sk}, \mathsf{d}_{x_i})$ for both $i \in \{0,1\}$;
   (iii) Randomly choose $\mathsf{rand} \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$ and compute $\mathsf{ctx} \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}, (m, \mathsf{d}_{x_b}, \sigma_{\mathsf{d}_{x_b}}, \pi); \mathsf{rand})$;
   (iv) $\underline{\text{Query } ((\mathsf{ek}, \mathsf{vk}, \mathsf{crs}_{\mathsf{CirS}}, \mathsf{d}_C, m, \mathsf{ctx}), (\mathsf{d}_{x_b}, \sigma_{\mathsf{d}_{x_b}}, \pi, \mathsf{rand})) \text{ to the challenge oracle, and receive } \Pi}$;
3. $\mathcal{B}$ sets $\Sigma := (\mathsf{ctx}, \Pi)$ and runs $\mathcal{A}_2(\mathsf{st}, (\mathsf{d}_{x_0}, \sigma_{\mathsf{d}_{x_0}}), (\mathsf{d}_{x_1}, \sigma_{\mathsf{d}_{x_1}}), \Sigma)$;
4. When $\mathcal{A}_2$ outputs b' and terminates, $\mathcal{B}$ outputs 1 if and only if $b' = b$; otherwise, it outputs 0.

The above completes the description of $\mathcal{B}$. If crs is generated by the $\mathsf{NIZK.Setup}$ (resp., Sim_0) algorithm and $\mathcal{B}$ accesses the $\mathsf{NIZK.Prove}$ (resp., Sim_1) oracle, then $\mathcal{B}$ perfectly simulates Game_0 (resp., Game_1) for $\mathcal{A}$. Therefore, we have $|\Pr[T_0] - \Pr[T_1]| = |\Pr[b = b' \text{ in } \mathsf{Game}_0] - \Pr[b = b' \text{ in } \mathsf{Game}_1]| = \mathsf{Adv}^{\mathsf{zk}}_{\mathsf{NIZK},\mathcal{B}}(\lambda)$.
$\square$ (**Lemma 4.1**)

Next, we show to have $\Pr[T_1] \leq \mathsf{negl}(\lambda)$. Intuitively, any algorithm that makes the probability $\Pr[T_1]$ non-negligible can distinguish two ciphertexts with different messages.

Lemma 4.2. *There exists a PPT algorithm $\mathcal{B}$ such that*

$$\Pr[T_1] = \mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{PKE},\mathcal{B}}(\lambda).$$

Proof. Assume that there exists a PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ which makes the probability $\Pr[T_1]$ non-negligible. Then, we can construct another PPT adversary $\mathcal{B}$ that breaks the IND-CPA security of the PKE with the same probability, which implies the lemma. The description of $\mathcal{B}$ is as follows.

1. $\mathcal{B}$ initially receives ek, computes $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{SIG.KG}(1^\lambda)$, $(\widetilde{\mathsf{crs}}, \mathsf{td}) \leftarrow \mathsf{Sim}_0(1^\lambda)$, and $\mathsf{crs}_{\mathsf{CirS}} \leftarrow \mathsf{CirS.Setup}(1^\lambda, 1^\ell)$, sets $\mathsf{pp} := (\mathsf{ek}, \mathsf{vk}, \widetilde{\mathsf{crs}}, \mathsf{crs}_{\mathsf{CirS}})$ and $\mathsf{msk} := (\mathsf{sk}, \mathsf{crs}_{\mathsf{CirS}})$, and runs $\mathcal{A}_1(\mathsf{pp}, \mathsf{msk})$.
2. When $\mathcal{A}_1$ outputs $(\mathsf{st}, x_0, x_1, C, m)$ and terminates, $\mathcal{B}$ proceeds as follows:
   (i) Compute $\mathsf{d}_C \leftarrow \mathsf{CirS.DCir}(\mathsf{crs}_{\mathsf{CirS}}, C)$, $(\cdot, \pi_i) \leftarrow \mathsf{CirS.Prove}(\mathsf{crs}_{\mathsf{CirS}}, x_i, C)$, and $\mathsf{d}_{x_i} \leftarrow \mathsf{CirS.DStr}(\mathsf{crs}_{\mathsf{CirS}}, x_i)$ for both $i \in \{0,1\}$;
   (ii) Compute $\sigma_{\mathsf{d}_{x_i}} \leftarrow \mathsf{SIG.Sign}(\mathsf{sk}, \mathsf{d}_{x_i})$ for both $i \in \{0,1\}$;
   (iii) $\underline{\text{Query } ((m, \mathsf{d}_{x_0}, \sigma_{\mathsf{d}_{x_0}}, \pi_0), (m, \mathsf{d}_{x_1}, \sigma_{\mathsf{d}_{x_1}}, \pi_1)) \text{ to the challenge oracle, and receive } \mathsf{ctx}}$;
   (iv) Compute $\Pi \leftarrow \mathsf{Sim}_1(\widetilde{\mathsf{crs}}, \mathsf{td}, (\mathsf{ek}, \mathsf{vk}, \mathsf{crs}_{\mathsf{CirS}}, \mathsf{d}_C, m, \mathsf{ctx})$;
3. $\mathcal{B}$ sets $\Sigma := (\mathsf{ctx}, \Pi)$ and runs $\mathcal{A}_2(\mathsf{st}, (\mathsf{d}_{x_0}, \sigma_{\mathsf{d}_{x_0}}), (\mathsf{d}_{x_1}, \sigma_{\mathsf{d}_{x_1}}), \Sigma)$;

4. When $\mathcal{A}_2$ outputs b' and terminates, $\mathcal{B}$ outputs b' and terminates.

The above completes the description of $\mathcal{B}$. It is easy to see that $\mathcal{B}$ perfectly simulates Game_1 for $\mathcal{A}$. Therefore, we have $\Pr[T_1] = \mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{PKE},\mathcal{B}}(\lambda)$.

$\square$ (**Lemma 4.2**)

Theorem 4.1 now follows immediately from Lemmata 4.1 and 4.2.

$\square$ (**Theorem 4.1**)

Theorem 4.2. *If the PKE scheme* PKE *is IND-CPA secure, the signature scheme* SIG *is EUF-CMA secure, the NIZK proof system* NIZK *is simulation sound and zero-knowledge, and the Circuit SNARG* CirS *is collision-resistant of the circuit digest and sound, then the above ABS scheme is unforgeable.*

Proof. Let us fix a PPT adversary $\mathcal{A}$ attacking the unforgeability of the ABS, the security parameter λ, and the attribute length ℓ. The attack game used to define the unforgeability is in Definition 2.6. We define three games as follows:

Game_0 : This game is the original attack game.

Game_1 : This game is the game identical with Game_0 except that we use modified algorithms, in which some steps are replaced as follows:

- In the $\mathsf{ABS.Setup}(1^\lambda, 1^\ell)$ algorithm, $\mathsf{crs}_{\mathsf{NIZK}} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$ is replaced by $(\widetilde{\mathsf{crs}}, \mathsf{td}) \leftarrow \mathsf{Sim}_0(1^\lambda)$.
- In the $\mathsf{ABS.Sign}(\mathsf{pp}, \mathsf{sk} = (\mathsf{d}_x, \sigma), x, C, m)$ algorithm, $\mathit{\Pi} \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_{\mathsf{NIZK}}, (\mathsf{ek}, \mathsf{vk}, \mathsf{crs}_{\mathsf{CirS}}, \mathsf{d}_C, m, \mathsf{ctx}), (\mathsf{d}_x, \sigma, \pi, \mathsf{rand}))$ is replaced by $\widetilde{\mathit{\Pi}} \leftarrow \mathsf{Sim}_1(\widetilde{\mathsf{crs}}, \mathsf{td}, (\mathsf{ek}, \mathsf{vk}, \mathsf{crs}_{\mathsf{CirS}}, \mathsf{d}_C, m, \mathsf{ctx}))$.

Game_2 : This game is the game identical with Game_1 except that the signing algorithm is modified in some step as follows:

- In the $\mathsf{ABS.Sign}(\mathsf{pp}, \mathsf{sk} = (\mathsf{d}_{x_b}, \sigma), x_b, C, m)$ algorithm, $\mathsf{ctx} \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}, (m, \mathsf{d}_x, \sigma_{\mathsf{d}_x}, \pi))$ is replaced by $\widetilde{\mathsf{ctx}} \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}, 0^t)$, where t is the total length of $(m, \mathsf{d}_x, \sigma_{\mathsf{d}_x}, \pi)$.

For $i = 0, 1, 2$, let T_i be the event that $\mathcal{A}$ wins the unforgeability experiment in the game Game_i. We will show that the probability $|\Pr[T_0] - \Pr[T_1]| \leq \mathsf{negl}(\lambda)$, $|\Pr[T_0] - \Pr[T_1]| \leq \mathsf{negl}(\lambda)$, and $\Pr[T_1] \leq \mathsf{negl}(\lambda)$ in turn.

We first show to have $|\Pr[T_0] - \Pr[T_1]| \leq \mathsf{negl}(\lambda)$. Similar to Lemma 4.1, any difference between these two games Game_0 and Game_1 yields a PPT algorithm that distinguishes the real proof from the simulated proof.

Lemma 4.3. *There exists a PPT algorithm* $\mathcal{B}$ *such that*

$$|\Pr[T_0] - \Pr[T_1]| = \mathsf{Adv}^{\mathsf{zk}}_{\mathsf{NIZK},\mathcal{B}}(\lambda).$$

Proof. Assume that there exists a PPT adversary $\mathcal{A}$ which makes the probability $|\Pr[T_0] - \Pr[T_1]|$ non-negligible. Then, we can construct another PPT adversary $\mathcal{B}$ that breaks the zero-knowledge property of NIZK with non-negligible probability, which implies the lemma. The description of $\mathcal{B}$ is as follows.

1. $\mathcal{B}$ initially receives crs, computes $(\mathsf{ek},\mathsf{dk}) \leftarrow \mathsf{PKE}(1^\lambda)$, $(\mathsf{sk},\mathsf{vk}) \leftarrow \mathsf{SIG.KG}(1^\lambda)$, and $\mathsf{crs}_{\mathsf{CirS}} \leftarrow \mathsf{CirS.Setup}(1^\lambda, 1^\ell)$, sets $L_{sig}, L_{corr}, L_{key} := \emptyset$, and $\mathsf{pp} := (\mathsf{ek}, \mathsf{vk}, \mathsf{crs}, \mathsf{crs}_{\mathsf{CirS}})$, and runs $\mathcal{A}(\mathsf{pp})$.
2. $\mathcal{B}$ responds to each of queries from $\mathcal{A}$ as follows:
   - For each query to $\mathcal{O}_{sig}(x, C, m)$, $\mathcal{B}$ returns $\perp$ if $C(x) = 0$; otherwise, it proceeds as follows:
     (i) If $\exists(x,\cdot) \notin L_{key}$, then
       (a) compute $\mathsf{d}_x \leftarrow \mathsf{CirS.DStr}(\mathsf{crs}_{\mathsf{CirS}}, x)$ and $\sigma_{\mathsf{d}_x} \leftarrow \mathsf{SIG.Sign}(\mathsf{sk}, \mathsf{d}_x)$;
       (b) add $(x, (\mathsf{d}_x, \sigma_{\mathsf{d}_x}))$ to L_{key};
       Otherwise, find $(x, \mathsf{sk}_x) \in L_{key}$ and parse $\mathsf{sk}_x = (\mathsf{d}_x, \sigma_{\mathsf{d}_x})$;
     (ii) Compute $\mathsf{d}_C \leftarrow \mathsf{CirS.DCir}(\mathsf{crs}_{\mathsf{CirS}}, C)$ and $(\cdot, \pi) \leftarrow \mathsf{CirS.Prove}(\mathsf{crs}_{\mathsf{CirS}}, x, C)$;
     (iii) Randomly choose $\mathsf{rand} \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$ and compute $\mathsf{ctx} \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}, (m, \mathsf{d}_x, \sigma_{\mathsf{d}_x}, \pi); \mathsf{rand})$;
     (iv) $\underline{\text{Query } ((\mathsf{ek}, \mathsf{vk}, \mathsf{crs}_{\mathsf{CirS}}, \mathsf{d}_C, m, \mathsf{ctx}), (\mathsf{d}_x, \sigma_{\mathsf{d}_x}, \pi, \mathsf{rand})) \text{ to the challenge oracle, and receive } \varPi}$.
     Then, $\mathcal{B}$ adds (C, m) to L_{sig} and returns $(\mathsf{ctx}, \varPi)$.
   - For each query to $\mathcal{O}_{corr}(x)$, $\mathcal{B}$ computes as follows:
     - If $\exists(x,\cdot) \notin L_{key}$, then $\mathcal{B}$ computes $\mathsf{d}_x \leftarrow \mathsf{CirS.DStr}(\mathsf{crs}_{\mathsf{CirS}}, x)$ and $\sigma_{\mathsf{d}_x} \leftarrow \mathsf{SIG.Sign}(\mathsf{sk}, \mathsf{d}_x)$, sets $\mathsf{sk}_x := (\mathsf{d}_x, \sigma_{\mathsf{d}_x})$, and adds (x, sk_x) to L_{key};
     - Otherwise, $\mathcal{B}$ finds $(x, \mathsf{sk}_x) \in L_{key}$.
     Finally, $\mathcal{B}$ adds x to L_{corr} and returns sk_x.
3. When $\mathcal{A}$ outputs $(C^*, m^*, \varSigma^* = (\mathsf{ctx}^*, \varPi^*))$ and terminates, $\mathcal{B}$ computes $\mathsf{d}_{C^*} \leftarrow \mathsf{CirS.DCir}(\mathsf{crs}_{\mathsf{CirS}}, C^*)$ and outputs 1 if $C^*(x) = 0$ for all $x \in L_{corr}$, $(C^*, m^*) \notin L_{sig}$, and $\mathsf{NIZK.Ver}(\mathsf{crs}_{\mathsf{NIZK}}, (\mathsf{ek}, \mathsf{vk}, \mathsf{crs}_{\mathsf{CirS}}, \mathsf{d}_{C^*}, m^*, \mathsf{ctx}^*), \varPi^*) = 1$; otherwise, it outputs 0.

The above completes the description of $\mathcal{B}$. If crs is generated by the $\mathsf{NIZK.Setup}$ (resp., Sim_0) algorithm and $\mathcal{B}$ accesses the $\mathsf{NIZK.Prove}$ (resp., Sim_1) oracle, then $\mathcal{B}$ perfectly simulates Game_0 (resp., Game_1) for $\mathcal{A}$. Therefore, we have $|\Pr[T_0] - \Pr[T_1]| = \mathsf{Adv}^{\mathsf{zk}}_{\mathsf{NIZK},\mathcal{B}}(\lambda)$.

□ (**Lemma 4.3**)

Secondly, we will show to have $|\Pr[T_1] - \Pr[T_2]| \leq \mathsf{negl}(\lambda)$. Intuitively, any difference between these two games Game_1 and Game_2 yields a PPT algorithm that distinguishes two ciphertexts with different messages.

Lemma 4.4. *There exists a PPT algorithm $\mathcal{B}$ such that*

$$|\Pr[T_1] - \Pr[T_2]| = \mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{PKE},\mathcal{B}}(\lambda).$$

Proof. Assume that there exists a PPT adversary $\mathcal{A}$ which makes the probability $|\Pr[T_1] - \Pr[T_2]|$ non-negligible. Then, we can construct another PPT adversary $\mathcal{B}$ that breaks the IND-CPA security of the PKE with non-negligible probability, which implies the lemma. The description of $\mathcal{B}$ is as follows:

1. $\mathcal{B}$ initially receives ek, computes $(\mathsf{sk},\mathsf{vk}) \leftarrow \mathsf{SIG.KG}(1^\lambda)$, $(\widetilde{\mathsf{crs}},\mathsf{td}) \leftarrow \mathsf{Sim}_0(1^\lambda)$, and $\mathsf{crs}_{\mathsf{CirS}} \leftarrow \mathsf{CirS.Setup}(1^\lambda, 1^\ell)$, sets $L_{sig}, L_{corr}, L_{key} := \emptyset$, and $\mathsf{pp} := (\mathsf{ek},\mathsf{vk},\widetilde{\mathsf{crs}},\mathsf{crs}_{\mathsf{CirS}})$, and runs $\mathcal{A}(\mathsf{pp})$.
2. $\mathcal{B}$ responds to each of queries from $\mathcal{A}$ as follows:
   – For each query to $\mathcal{O}_{sig}(x,C,m)$, $\mathcal{B}$ returns $\perp$ if $C(x)=0$; otherwise, it proceeds as follows:
      (i) If $\exists(x,\cdot)\notin L_{key}$, then
         (a) compute $\mathsf{d}_x \leftarrow \mathsf{CirS.DStr}(\mathsf{crs}_{\mathsf{CirS}},x)$ and $\sigma_{\mathsf{d}_x} \leftarrow \mathsf{SIG.Sign}(\mathsf{sk},\mathsf{d}_x)$;
         (b) add $(x,(\mathsf{d}_x,\sigma_{\mathsf{d}_x}))$ to L_{key};
         Otherwise, find $(x,\mathsf{sk}_x)\in L_{key}$ and parse $\mathsf{sk}_x=(\mathsf{d}_x,\sigma_{\mathsf{d}_x})$;
      (ii) Compute $\mathsf{d}_C \leftarrow \mathsf{CirS.DCir}(\mathsf{crs}_{\mathsf{CirS}},C)$ and $(\cdot,\pi)\leftarrow \mathsf{CirS.Prove}(\mathsf{crs}_{\mathsf{CirS}},x,C)$;
      (iii) $\underline{\text{Query } ((m,\mathsf{d}_x,\sigma_{\mathsf{d}_x},\pi),0^t) \text{ to the challenge oracle, where } t \text{ is the}}$ $\underline{\text{length of message } (m,\mathsf{d}_x,\sigma_{\mathsf{d}_x},\pi)\text{, and receive } \mathsf{ctx}}$;
      (iv) Compute $\Pi \leftarrow \mathsf{Sim}_1(\widetilde{\mathsf{crs}},\mathsf{td},(\mathsf{ek},\mathsf{vk},\mathsf{crs}_{\mathsf{CirS}},\mathsf{d}_C,m,\mathsf{ctx}))$;
      Then, $\mathcal{B}$ adds (C,m) to L_{sig} and returns (ctx,Π).
   – For each query to $\mathcal{O}_{corr}(x)$, $\mathcal{B}$ computes in the same way as described in the proof of Lemma 4.3.
3. When $\mathcal{A}$ outputs $(C^*,m^*,\Sigma^*=(\mathsf{ctx}^*,\Pi^*))$ and terminates, $\mathcal{B}$ computes $\mathsf{d}_{C^*}\leftarrow \mathsf{CirS.DCir}(\mathsf{crs}_{\mathsf{CirS}},C^*)$ and outputs 1 if $C^*(x)=0$ for all $x\in L_{corr}$, $(C^*,m^*)\notin L_{sig}$, and $\mathsf{NIZK.Ver}(\mathsf{crs}_{\mathsf{NIZK}},(\mathsf{ek},\mathsf{vk},\mathsf{crs}_{\mathsf{CirS}},\mathsf{d}_{C^*},m^*,\mathsf{ctx}^*),\Pi^*)=1$; otherwise, it outputs 0.

The above completes the description of $\mathcal{B}$. If ctx is an encryption of a message $(m,\mathsf{d}_x,\sigma_{\mathsf{d}_x},\pi)$ (resp., 0^t), then $\mathcal{B}$ perfectly simulates Game_1 (resp., Game_2) for $\mathcal{A}$. Therefore, we have $|\Pr[T_1]-\Pr[T_2]| = 2\cdot \mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{PKE},\mathcal{B}}(\lambda)$.

$\square$ (**Lemma 4.4**)

Thirdly, we will show $\Pr[T_2]\le \mathsf{negl}(\lambda)$, so we focus on the game Game_2 only. We consider the winning condition for $\mathcal{A}$ in the game Game_2. In the following, let t be the fixed total length of a message m, an attribute digest d_x, its signature σ_{d_x}, and a Circuit SNARG proof π. In addition, let $(C^*,m^*,\Sigma^*=(\mathsf{ctx}^*,\Pi^*))$ be the $\mathcal{A}$'s output of the game.

In the following, we will use the notation $\star$ denoting that some value exists but is being ignored. We consider two events in the game as follows:

E_{sig} : is the event that there exists $(C,\star)\in L_{sig}$ such that $\mathsf{CirS.DCir}(\mathsf{crs}_{\mathsf{CirS}},C^*)=\mathsf{CirS.DCir}(\mathsf{crs}_{\mathsf{CirS}},C)$ and $C^*\neq C$.
$\mathsf{E}_{\bar{\rho}}$: is the event that the statement $(\mathsf{ek},\mathsf{vk},\mathsf{crs}_{\mathsf{CirS}},\mathsf{d}_{C^*},m^*,\mathsf{ctx}^*)$ is not in the language corresponding to the NIZK relation ρ, where $\mathsf{d}_{C^*}=\mathsf{CirS.DCir}(\mathsf{crs}_{\mathsf{CirS}},C^*)$.
E_{key} : is the event that there exists $x\in L_{corr}$ such that $\mathsf{d}^*=\mathsf{CirS.DStr}(\mathsf{crs}_{\mathsf{CirS}},x)$, where $(\star,\mathsf{d}^*,\star,\star)=\mathsf{PKE.Dec}(\mathsf{dk},\mathsf{ctx}^*)$.

We clearly have

$$\Pr[T_2] \le \Pr[T_2\wedge \mathsf{E}_{sig}] + \Pr[T_2\wedge\neg\mathsf{E}_{sig}\wedge\mathsf{E}_{\bar{\rho}}]$$

$$+ \Pr[T_2 \wedge \neg \mathsf{E}_{sig} \wedge \neg \mathsf{E}_{\bar{\rho}} \wedge \mathsf{E}_{key}] + \Pr[T_2 \wedge \neg \mathsf{E}_{sig} \wedge \neg \mathsf{E}_{\bar{\rho}} \wedge \neg \mathsf{E}_{key}].$$

In the following, we separate the winning condition in the game into four cases.

First, if the event E_{sig} occurs, it is easy to see that if $\mathcal{A}$ wins in the game Game_2, then it breaks the collision resistance of the circuit digest of the Circuit SNARG scheme.

Lemma 4.5. *There exists a PPT algorithm $\mathcal{B}$ such that*

$$\Pr[T_2 \wedge \mathsf{E}_{sig}] \leq \mathsf{Adv}^{\mathsf{col\text{-}cir}}_{\mathsf{CirS},\mathcal{B}}(\lambda).$$

Proof. Assume that there exists a PPT adversary $\mathcal{A}$ which makes the probability $\Pr[T_2 \wedge \mathsf{E}_{sig}]$ non-negligible. Then, we can construct another PPT adversary $\mathcal{B}$ that breaks the collision resistance of the circuit digest of CirS with non-negligible probability, which implies the lemma. The description of $\mathcal{B}$ is as follows, where $\star$ denotes that some value exists but is being ignored.

1. $\mathcal{B}$ initially receives $\mathsf{crs}_{\mathsf{CirS}}$, computes $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{PKE}(1^\lambda)$, $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{SIG.KG}(1^\lambda)$, and $(\widetilde{\mathsf{crs}}, \mathsf{td}) \leftarrow \mathsf{Sim}_0(1^\lambda)$, sets $L_{sig}, L_{corr}, L_{key} := \emptyset$, and $\mathsf{pp} := (\mathsf{ek}, \mathsf{vk}, \widetilde{\mathsf{crs}}, \mathsf{crs}_{\mathsf{CirS}})$, and runs $\mathcal{A}(\mathsf{pp})$.
2. $\mathcal{B}$ responds to each of queries from $\mathcal{A}$ as follows:
   - For each query to $\mathcal{O}_{sig}(x, C, m)$, $\mathcal{B}$ returns $\perp$ if $C(x) = 0$; otherwise, it proceeds as follows:
     - (i) Compute $\mathsf{d}_C \leftarrow \mathsf{CirS.DCir}(\mathsf{crs}_{\mathsf{CirS}}, C)$;
     - (ii) Compute $\widetilde{\mathsf{ctx}} \leftarrow \mathsf{PKE.Enc}(0^t)$, where t is the fixed length in Game_2;
     - (iii) Compute $\Pi \leftarrow \mathsf{Sim}_1(\widetilde{\mathsf{crs}}, \mathsf{td}, (\mathsf{ek}, \mathsf{vk}, \mathsf{crs}_{\mathsf{CirS}}, \mathsf{d}_C, m, \widetilde{\mathsf{ctx}}))$.

     Then, $\mathcal{B}$ adds (C, m) to L_{sig} and returns (ctx, Π).
   - For each query to $\mathcal{O}_{corr}(x)$, $\mathcal{B}$ computes in the same way as described in the proof of Lemma 4.3.
3. When $\mathcal{A}$ outputs $(C^*, m^*, \Sigma^* = (\mathsf{ctx}^*, \Pi^*))$ and terminates, $\mathcal{B}$ finds C such that $(C, \star) \in L_{sig}$ and $\mathsf{CirS.DCir}(\mathsf{crs}_{\mathsf{CirS}}, C) = \mathsf{CirS.DCir}(\mathsf{crs}_{\mathsf{CirS}}, C^*)$. If $\mathcal{B}$ cannot find such C, then it outputs $\perp$; otherwise, it outputs (C, C^*).

The above completes the description of $\mathcal{B}$. It is easy to see that $\mathcal{B}$ perfectly simulates the game Game_2 for $\mathcal{A}$. Let $\mathsf{X}^* = (\mathsf{ek}, \mathsf{vk}, \mathsf{crs}_{\mathsf{CirS}}, \mathsf{d}_{C^*}, m^*, \mathsf{ctx}^*)$. When the event E_{sig} occurs, $\mathcal{B}$ can always find C such that $(C, \star) \in L_{sig}$, $C \neq C^*$, and $\mathsf{CirS.DCir}(\mathsf{crs}_{\mathsf{CirS}}, C) = \mathsf{CirS.DCir}(\mathsf{crs}_{\mathsf{CirS}}, C^*)$. Therefore, we have $\Pr[T_2 \wedge \mathsf{E}_{sig}] \leq \mathsf{Adv}^{\mathsf{col\text{-}cir}}_{\mathsf{CirS},\mathcal{B}}(\lambda)$.

□ (**Lemma 4.5**)

Second, if the event E_{sig} never occurs but the event $\mathsf{E}_{\bar{\rho}}$ occurs, it is easy to see that if $\mathcal{A}$ wins in the game Game_2, then it breaks the simulation soundness of the NIZK scheme.

Lemma 4.6. *There exists a PPT algorithm $\mathcal{B}$ such that*

$$\Pr[T_2 \wedge \neg \mathsf{E}_{sig} \wedge \mathsf{E}_{\bar{\rho}}] \leq \mathsf{Adv}^{\mathsf{sim\text{-}sound}}_{\mathsf{NIZK},\mathcal{B}}(\lambda).$$

Proof. Assume that there exists a PPT adversary $\mathcal{A}$ which makes the probability $\Pr[T_2 \wedge \neg \mathsf{E}_{sig} \wedge \mathsf{E}_{\bar{\rho}}]$ non-negligible. Then, we can construct another PPT adversary $\mathcal{B}$ that breaks the simulation soundness of NIZK with non-negligible probability, which implies the lemma. The description of $\mathcal{B}$ is as follows.

1. $\mathcal{B}$ initially receives $\widetilde{\mathsf{crs}}$, computes $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{PKE}(1^\lambda)$, $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{SIG.KG}(1^\lambda)$, and $\mathsf{crs}_{\mathsf{CirS}} \leftarrow \mathsf{CirS.Setup}(1^\lambda, 1^\ell)$, sets $L_{sig}, L_{corr}, L_{key} := \emptyset$, and $\mathsf{pp} := (\mathsf{ek}, \mathsf{vk}, \widetilde{\mathsf{crs}}, \mathsf{crs}_{\mathsf{CirS}})$, and runs $\mathcal{A}(\mathsf{pp})$.
2. $\mathcal{B}$ responds to each of queries from $\mathcal{A}$ as follows:
   - For each query to $\mathcal{O}_{sig}(x, C, m)$, $\mathcal{B}$ returns $\perp$ if $C(x) = 0$; otherwise, it proceeds as follows:
     (i) Compute $\mathsf{d}_C \leftarrow \mathsf{CirS.DCir}(\mathsf{crs}_{\mathsf{CirS}}, C)$;
     (ii) Compute $\widetilde{\mathsf{ctx}} \leftarrow \mathsf{PKE.Enc}(0^t)$, where t is the fixed length in Game_2;
     (iii) $\underline{\text{Query } (\mathsf{ek}, \mathsf{vk}, \mathsf{crs}_{\mathsf{CirS}}, \mathsf{d}_C, m, \widetilde{\mathsf{ctx}}) \text{ to the simulation oracle, and receive } \Pi}$.

     Then, $\mathcal{B}$ adds (C, m) to L_{sig} and returns (ctx, Π).
   - For each query to $\mathcal{O}_{corr}(x)$, $\mathcal{B}$ computes in the same way as described in the proof of Lemma 4.3.
3. When $\mathcal{A}$ outputs $(C^*, m^*, \Sigma^* = (\mathsf{ctx}^*, \Pi^*))$ and terminates, $\mathcal{B}$ computes $\mathsf{d}_{C^*} \leftarrow \mathsf{CirS.DCir}(\mathsf{crs}_{\mathsf{CirS}}, C^*)$ and outputs $((\mathsf{ek}, \mathsf{vk}, \mathsf{crs}_{\mathsf{CirS}}, \mathsf{d}_{C^*}, m^*, \mathsf{ctx}^*), \Pi^*)$.

The above completes the description of $\mathcal{B}$. It is easy to see that $\mathcal{B}$ perfectly simulates the game Game_2 for $\mathcal{A}$. Let $\mathsf{X}^* = (\mathsf{ek}, \mathsf{vk}, \mathsf{crs}_{\mathsf{CirS}}, \mathsf{d}_{C^*}, m^*, \mathsf{ctx}^*)$. When the event $\mathsf{E}_{\bar{\rho}}$ occurs, we have $\mathsf{X}^* \notin L_\rho$, where $L_\rho := \{x \mid \exists w \text{ s.t. } (x, w) \in \rho\}$. On the other hand, when $\mathcal{A}$ wins, we have $\mathsf{NIZK.Ver}(\widetilde{\mathsf{crs}}, \mathsf{X}^*, \Pi^*) = 1$. In addition, X^* is never queried to the simulation oracle from the following reason: if $(\star, m^*) \notin L_{sig}$, then X^* is never queried; otherwise, there is no C such that $(C, m^*) \in L_{sig}$ and $\mathsf{CirS.DCir}(\mathsf{crs}_{\mathsf{CirS}}, C) = \mathsf{CirS.DCir}(\mathsf{crs}_{\mathsf{CirS}}, C^*)$ since the event E_{sig} never occurs and we have $(C^*, m^*) \notin L_{sig}$ due to the winning condition for $\mathcal{A}$. Therefore, we have $\Pr[T_2 \wedge \neg \mathsf{E}_{sig} \wedge \mathsf{E}_{\bar{\rho}}] \leq \mathsf{Adv}^{\mathsf{sim\text{-}sound}}_{\mathsf{NIZK},\mathcal{B}}(\lambda)$.

$\square$ (**Lemma 4.6**)

Third, if the events E_{sig} and $\mathsf{E}_{\bar{\rho}}$ never occur but the event E_{key} occurs, $\mathcal{A}$ must generate a circuit SNARG proof that passes the verification of the Circuit SNARG. However, we have $C^*(x) = 0$ for all $x \in L_{corr}$ when $\mathcal{A}$ wins the game. Therefore, to win the game, $\mathcal{A}$ has to break the soundness of the CirS scheme.

Lemma 4.7. *There exist PPT algorithms $\mathcal{B}$ such that*

$$\Pr[T_2 \wedge \neg \mathsf{E}_{sig} \wedge \neg \mathsf{E}_{\bar{\rho}} \wedge \mathsf{E}_{key}] \leq \mathsf{Adv}^{\mathsf{sound}}_{\mathsf{CirS},\mathcal{B}}(\lambda).$$

Proof. Assume that there exists a PPT adversary $\mathcal{A}$ which makes the probability $\Pr[T_2 \wedge \neg \mathsf{E}_{sig} \wedge \neg \mathsf{E}_{\bar{\rho}} \wedge \mathsf{E}_{key}]$ non-negligible. Then, we can construct another PPT adversary $\mathcal{B}$ that breaks the soundness of CirS with non-negligible probability, which implies the lemma. The description of $\mathcal{B}$ is as follows, where $\star$ denotes that some value exists but is being ignored.

1. $\mathcal{B}$ initially receives $\mathsf{crs}_{\mathsf{CirS}}$, computes $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{PKE}(1^\lambda)$, $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{SIG.KG}(1^\lambda)$, and $(\widetilde{\mathsf{crs}}, \mathsf{td}) \leftarrow \mathsf{Sim}_0(1^\lambda)$, and sets $L_{sig}, L_{corr}, L_{key} = \emptyset$, and $\mathsf{pp} := (\mathsf{ek}, \mathsf{vk}, \widetilde{\mathsf{crs}}, \mathsf{crs}_{\mathsf{CirS}})$, and runs $\mathcal{A}(\mathsf{pp})$.
2. $\mathcal{B}$ responds to each of queries from $\mathcal{A}$ in the same way as described in the proof of Lemma 4.5.
3. When $\mathcal{A}$ outputs $(C^*, m^*, \Sigma^* = (\mathsf{ctx}^*, \Pi^*))$ and terminates, $\mathcal{B}$ computes $(m^*, \mathsf{d}^*, \sigma_{\mathsf{d}^*}, \pi^*) \leftarrow \mathsf{PKE.Dec}(\mathsf{dk}, \mathsf{ctx}^*)$ and finds x^* such that $(x^*, (\mathsf{d}^*, \star)) \in L_{key}$. If $\mathcal{B}$ cannot find such x^*, then it outputs $\perp$; otherwise, it outputs $(x^*, C^*, 1, \pi^*)$.

The above completes the description of $\mathcal{B}$. It is easy to see that $\mathcal{B}$ perfectly simulates the game Game_2 for $\mathcal{A}$. Let $\mathsf{X}^* = (\mathsf{ek}, \mathsf{vk}, \mathsf{crs}_{\mathsf{CirS}}, \mathsf{d}_{C^*}, m^*, \mathsf{ctx}^*)$. When the event E_{key} occurs, $\mathcal{B}$ can always find x^* such that $(x^*, (\mathsf{d}^*, \star)) \in L_{key}$. On the other hand, when $\mathcal{A}$ wins, we have $C^*(x) = 0$ for all $(x, \star) \in L_{key}$ since $\{x \mid x \in L_{corr}\} = \{x \mid (x, \star) \in L_{key}\}$ from the above description of $\mathcal{B}$. We also have $\mathsf{CirS.Ver}(\mathsf{crs}_{\mathsf{CirS}}, \mathsf{d}^*, \mathsf{d}_{C^*}, 1, \pi^*) = 1$ since we have $\mathsf{X} \in L_\rho$ and both d^* and d_{C^*} are calculated deterministically. Therefore, we have $\Pr[T_2 \wedge \neg \mathsf{E}_{sig} \neg \mathsf{E}_{\bar{\rho}} \wedge \mathsf{E}_{key}] \leq \mathsf{Adv}^{\mathsf{sound}}_{\mathsf{CirS},\mathcal{B}}(\lambda)$.

□ (**Lemma 4.7**)

Finally, if all the events E_{sig}, $\mathsf{E}_{\bar{\rho}}$, and E_{key} never occur, $\mathcal{A}$ must generate a valid signature for d_{x^*}. Thus, if $\mathcal{A}$ wins, then it breaks the EUF-CMA security of the signature scheme.

Lemma 4.8. *There exists a PPT algorithm $\mathcal{B}$ such that*

$$\Pr[T_2 \wedge \neg \mathsf{E}_{sig} \wedge \neg \mathsf{E}_{\bar{\rho}} \wedge \neg \mathsf{E}_{key}] \leq \mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathsf{SIG},\mathcal{B}}(\lambda).$$

Proof. Assume that there exists a PPT adversary $\mathcal{A}$ which makes the probability $\Pr[T_2 \wedge \neg \mathsf{E}_{sig} \wedge \neg \mathsf{E}_{\bar{\rho}} \wedge \neg \mathsf{E}_{key}]$ non-negligible. Then, we can construct another PPT adversary $\mathcal{B}$ that breaks the EUF-CMA security of SIG with non-negligible probability, which implies the lemma. The description of $\mathcal{B}$ is as follows.

1. $\mathcal{B}$ initially receives vk, computes $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{PKE}(1^\lambda)$, $(\widetilde{\mathsf{crs}}, \mathsf{td}) \leftarrow \mathsf{Sim}_0(1^\lambda)$, and $\mathsf{crs}_{\mathsf{CirS}} \leftarrow \mathsf{CirS.Setup}(1^\lambda, 1^\ell)$, and sets $L_{sig}, L_{corr}, L_{key} := \emptyset$, and $\mathsf{pp} := (\mathsf{ek}, \mathsf{vk}, \widetilde{\mathsf{crs}}, \mathsf{crs}_{\mathsf{CirS}})$, and runs $\mathcal{A}(\mathsf{pp})$.
2. $\mathcal{B}$ responds to each of queries from $\mathcal{A}$ as follows, where $\star$ denotes that some value exists but is being ignored:
   – For each query to $\mathcal{O}_{sig}(x, C, m)$, $\mathcal{B}$ computes in the same way as described in the proof of Lemma 4.5.
   – For each query to $\mathcal{O}_{corr}(x)$, $\mathcal{B}$ computes as follows:
     - If $\exists (x, \star) \notin L_{key}$, then $\mathcal{B}$ proceeds as follows:
       (i) Compute $\mathsf{d}_x \leftarrow \mathsf{CirS.DStr}(\mathsf{crs}_{\mathsf{CirS}}, x)$;
       (ii) $\underline{\text{Query } \mathsf{d}_x \text{ to the signing oracle and receive } \sigma_{\mathsf{d}_x}}$;
       (iii) Set $\mathsf{sk}_x := (\mathsf{d}_x, \sigma_{\mathsf{d}_x})$, and add (x, sk_x) to L_{key}.

- Otherwise, $\mathcal{B}$ finds $(x, \mathsf{sk}_x) \in L_{key}$.

  Then, $\mathcal{B}$ adds x to L_{corr} and returns $(\mathsf{d}_x, \sigma_{\mathsf{d}_x})$.

3. When $\mathcal{A}$ outputs $(C^*, m^*, \Sigma^* = (\mathsf{ctx}^*, \Pi^*))$ and terminates, $\mathcal{B}$ computes $(m^*, \mathsf{d}^*, \sigma_{\mathsf{d}^*}, \pi^*) \leftarrow \mathsf{PKE.Dec}(\mathsf{dk}, \mathsf{ctx}^*)$ and outputs $(\mathsf{d}^*, \sigma_{\mathsf{d}^*})$.

The above completes the description of $\mathcal{B}$. It is easy to see that $\mathcal{B}$ perfectly simulates the game Game_2 for $\mathcal{A}$. Let $\mathsf{X}^* = (\mathsf{ek}, \mathsf{vk}, \mathsf{crs}_{\mathsf{CirS}}, \mathsf{d}_{C^*}, m^*, \mathsf{ctx}^*)$. Since the event E_{key} never occurs, $\mathcal{B}$ never queries d^* to the signing oracle in the EUF-CMA security experiment. On the other hand, when $\mathcal{A}$ wins, $\mathsf{SIG.Ver}(\mathsf{vk}, \mathsf{d}^*, \sigma_{\mathsf{d}^*}) = 1$ since we have $\mathsf{X} \in L_\rho$. Therefore, we have $\Pr[T_2 \wedge \neg \mathsf{E}_{sig} \wedge \neg \mathsf{E}_{\bar{\rho}} \wedge \neg \mathsf{E}_{key}] \leq \mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathsf{SIG},\mathcal{B}}(\lambda)$.

□ (**Lemma 4.8**)

Theorem 4.2 now follows immediately from Lemmata 4.3 to 4.8.

□ (**Theorem 4.2**)

Acknowledgement. This work was partially supported by JSPS KAKENHI Grant Number JP23KJ0548.

References

1. Attrapadung, N., Hanaoka, G., Yamada, S.: Conversions among several classes of predicate encryption and applications to ABE with various compactness tradeoffs. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015, Part I. LNCS, vol. 9452, pp. 575–601. Springer, Heidelberg (Nov / Dec 2015) 2, 5
2. Blömer, J., Eidens, F., Juhnke, J.: Enhanced security of attribute-based signatures. In: Camenisch, J., Papadimitratos, P. (eds.) CANS 18. LNCS, vol. 11124, pp. 235–255. Springer, Heidelberg (Sep / Oct 2018) 6
3. Boneh, D., Zhandry, M.: Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 480–499. Springer, Heidelberg (Aug 2014) 3
4. Boyle, E., Goldwasser, S., Ivan, I.: Functional signatures and pseudorandom functions. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 501–519. Springer, Heidelberg (Mar 2014) 2
5. Brakerski, Z., Brodsky, M.F., Kalai, Y.T., Lombardi, A., Paneth, O.: SNARGs for monotone policy batch NP. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part II. LNCS, vol. 14082, pp. 252–283. Springer, Heidelberg (Aug 2023) 6, 9
6. Brakerski, Z., Holmgren, J., Kalai, Y.T.: Non-interactive delegation and batch NP verification from standard computational assumptions. In: Hatami, H., McKenzie, P., King, V. (eds.) 49th ACM STOC. pp. 474–482. ACM Press (Jun 2017) 3, 10
7. Campanelli, M., Ganesh, C., Khoshakhlagh, H., Siim, J.: Impossibilities in succinct arguments: Black-box extraction and more. In: El Mrabet, N., De Feo, L., Duquesne, S. (eds.) AFRICACRYPT 23. LNCS, vol. 14064, pp. 465–489. Springer Nature (Jul 2023) 6

8. Chen, C., Chen, J., Lim, H.W., Zhang, Z., Feng, D., Ling, S., Wang, H.: Fully secure attribute-based systems with short ciphertexts/signatures and threshold access structures. In: Dawson, E. (ed.) CT-RSA 2013. LNCS, vol. 7779, pp. 50–67. Springer, Heidelberg (Feb / Mar 2013) 2, 5
9. Choudhuri, A.R., Garg, S., Jain, A., Jin, Z., Zhang, J.: Correlation intractability and SNARGs from sub-exponential DDH. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part IV. LNCS, vol. 14084, pp. 635–668. Springer, Heidelberg (Aug 2023) 6
10. Choudhuri, A.R., Jain, A., Jin, Z.: Non-interactive batch arguments for NP from standard assumptions. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part IV. LNCS, vol. 12828, pp. 394–423. Springer, Heidelberg, Virtual Event (Aug 2021) 6
11. Choudhuri, A.R., Jain, A., Jin, Z.: SNARGs for $\mathcal{P}$ from LWE. In: 62nd FOCS. pp. 68–79. IEEE Computer Society Press (Feb 2022) 3, 6, 10, 11
12. Datta, P., Dutta, R., Mukhopadhyay, S.: Short attribute-based signatures for arbitrary turing machines from standard assumptions. DCC **91**(5), 1845–1872 (2023) 2, 6
13. Devadas, L., Goyal, R., Kalai, Y., Vaikuntanathan, V.: Rate-1 non-interactive arguments for batch-NP and applications. In: 63rd FOCS. pp. 1057–1068. IEEE Computer Society Press (Oct / Nov 2022) 6
14. Ding, S., Zhao, Y., Liu, Y.: Efficient traceable attribute-based signature. In: 2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications. pp. 582–589 (2014) 6
15. Döttling, N., Garg, S., Ishai, Y., Malavolta, G., Mour, T., Ostrovsky, R.: Trapdoor hash functions and their applications. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part III. LNCS, vol. 11694, pp. 3–32. Springer, Heidelberg (Aug 2019) 6
16. Dragan, C.C., Gardham, D., Manulis, M.: Hierarchical attribute-based signatures. In: Camenisch, J., Papadimitratos, P. (eds.) CANS 18. LNCS, vol. 11124, pp. 213–234. Springer, Heidelberg (Sep / Oct 2018) 6
17. El Kaafarani, A., Ghadafi, E., Khader, D.: Decentralized traceable attribute-based signatures. In: Benaloh, J. (ed.) CT-RSA 2014. LNCS, vol. 8366, pp. 327–348. Springer, Heidelberg (Feb 2014) 6
18. El Kaafarani, A., Katsumata, S.: Attribute-based signatures for unbounded circuits in the ROM and efficient instantiations from lattices. In: Abdalla, M., Dahab, R. (eds.) PKC 2018, Part II. LNCS, vol. 10770, pp. 89–119. Springer, Heidelberg (Mar 2018) 2, 5
19. Escala, A., Herranz, J., Morillo, P.: Revocable attribute-based signatures with adaptive security in the standard model. In: Nitaj, A., Pointcheval, D. (eds.) AFRICACRYPT 11. LNCS, vol. 6737, pp. 224–241. Springer, Heidelberg (Jul 2011) 6
20. Gardham, D., Manulis, M.: Hierarchical attribute-based signatures: Short keys and optimal signature length. In: Deng, R.H., Gauthier-Umaña, V., Ochoa, M., Yung, M. (eds.) ACNS 19. LNCS, vol. 11464, pp. 89–109. Springer, Heidelberg (Jun 2019) 6
21. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: 54th FOCS. pp. 40–49. IEEE Computer Society Press (Oct 2013) 8
22. Gentry, C., Wichs, D.: Separating succinct non-interactive arguments from all falsifiable assumptions. In: Fortnow, L., Vadhan, S.P. (eds.) 43rd ACM STOC. pp. 99–108. ACM Press (Jun 2011) 6

23. Ghadafi, E.: Stronger security notions for decentralized traceable attribute-based signatures and more efficient constructions. In: Nyberg, K. (ed.) CT-RSA 2015. LNCS, vol. 9048, pp. 391–409. Springer, Heidelberg (Apr 2015) 6
24. González, A., Zacharakis, A.: Fully-succinct publicly verifiable delegation from constant-size assumptions. In: Nissim, K., Waters, B. (eds.) TCC 2021, Part I. LNCS, vol. 13042, pp. 529–557. Springer, Heidelberg (Nov 2021) 6
25. Guo, H., Li, W., Meamari, E., Shen, C.C., Nejad, M.: Attribute-based multi-signature and encryption for ehr management: A blockchain-based solution. In: 2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC). pp. 1–5 (2020) 1
26. Herranz, J., Laguillaumie, F., Libert, B., Ràfols, C.: Short attribute-based signatures for threshold predicates. In: Dunkelman, O. (ed.) CT-RSA 2012. LNCS, vol. 7178, pp. 51–67. Springer, Heidelberg (Feb / Mar 2012) 2, 5
27. Hulett, J., Jawale, R., Khurana, D., Srinivasan, A.: SNARGs for P from sub-exponential DDH and QR. In: Dunkelman, O., Dziembowski, S. (eds.) EURO-CRYPT 2022, Part II. LNCS, vol. 13276, pp. 520–549. Springer, Heidelberg (May / Jun 2022) 6
28. Kalai, Y., Lombardi, A., Vaikuntanathan, V., Wichs, D.: Boosting batch arguments and RAM delegation. In: Saha, B., Servedio, R.A. (eds.) 55th ACM STOC. pp. 1545–1552. ACM Press (Jun 2023) 3, 4, 6, 9, 10, 11
29. Kalai, Y.T., Paneth, O.: Delegating RAM computations. In: Hirt, M., Smith, A.D. (eds.) TCC 2016-B, Part II. LNCS, vol. 9986, pp. 91–118. Springer, Heidelberg (Oct / Nov 2016) 3, 10
30. Kalai, Y.T., Paneth, O., Yang, L.: How to delegate computations publicly. In: Charikar, M., Cohen, E. (eds.) 51st ACM STOC. pp. 1115–1124. ACM Press (Jun 2019) 3, 6, 10, 11
31. Kalai, Y.T., Raz, R., Rothblum, R.D.: How to delegate computations: The power of no-signaling proofs. J. ACM **69**(1) (nov 2021), https://doi.org/10.1145/3456867 6
32. Kalai, Y.T., Vaikuntanathan, V., Zhang, R.Y.: Somewhere statistical soundness, post-quantum security, and SNARGs. In: Nissim, K., Waters, B. (eds.) TCC 2021, Part I. LNCS, vol. 13042, pp. 330–368. Springer, Heidelberg (Nov 2021) 6
33. Li, J., Au, M.H., Susilo, W., Xie, D., Ren, K.: Attribute-based signature and its applications. In: Feng, D., Basin, D.A., Liu, P. (eds.) ASIACCS 10. pp. 60–69. ACM Press (Apr 2010) 1, 2, 5
34. Ling, S., Nguyen, K., Phan, D.H., Tang, K.H., Wang, H., Xu, Y.: Fully dynamic attribute-based signatures for circuits from codes. In: Tang, Q., Teague, V. (eds.) PKC 2024, Part I. LNCS, vol. 14601, pp. 37–73. Springer, Heidelberg (Apr 2024) 2, 5, 6
35. Maji, H.K., Prabhakaran, M., Rosulek, M.: Attribute-based signatures. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 376–392. Springer, Heidelberg (Feb 2011) 1, 2, 5, 11
36. Merkle, R.C.: A digital signature based on a conventional encryption function. In: Pomerance, C. (ed.) CRYPTO'87. LNCS, vol. 293, pp. 369–378. Springer, Heidelberg (Aug 1988) 9
37. Micali, S.: CS proofs (extended abstracts). In: 35th FOCS. pp. 436–453. IEEE Computer Society Press (Nov 1994) 6
38. Nandi, M., Pandit, T.: On the power of pair encodings: Frameworks for predicate cryptographic primitives. Cryptology ePrint Archive, Paper 2015/955 (2015), https://eprint.iacr.org/2015/955 2, 5

39. Okamoto, T., Takashima, K.: Efficient attribute-based signatures for non-monotone predicates in the standard model. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 35–52. Springer, Heidelberg (Mar 2011) 2, 5
40. Okamoto, T., Takashima, K.: Decentralized attribute-based signatures. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 125–142. Springer, Heidelberg (Feb / Mar 2013) 6
41. Sakai, Y.: Succinct attribute-based signatures for bounded-size circuits by combining algebraic and arithmetic proofs. In: Galdi, C., Jarecki, S. (eds.) Security and Cryptography for Networks. pp. 711–734. Springer International Publishing, Cham (2022) 11
42. Sakai, Y., Attrapadung, N., Hanaoka, G.: Attribute-based signatures for circuits from bilinear map. In: Cheng, C.M., Chung, K.M., Persiano, G., Yang, B.Y. (eds.) PKC 2016, Part I. LNCS, vol. 9614, pp. 283–300. Springer, Heidelberg (Mar 2016) 2, 5
43. Sakai, Y., Katsumata, S., Attrapadung, N., Hanaoka, G.: Attribute-based signatures for unbounded languages from standard assumptions. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part II. LNCS, vol. 11273, pp. 493–522. Springer, Heidelberg (Dec 2018) 2, 6
44. Shahandashti, S.F., Safavi-Naini, R.: Threshold attribute-based signatures and their application to anonymous credential systems. In: Preneel, B. (ed.) AFRICACRYPT 09. LNCS, vol. 5580, pp. 198–216. Springer, Heidelberg (Jun 2009) 1, 2, 5
45. Tang, F., Li, H., Liang, B.: Attribute-based signatures for circuits from multilinear maps. In: Chow, S.S.M., Camenisch, J., Hui, L.C.K., Yiu, S.M. (eds.) ISC 2014. LNCS, vol. 8783, pp. 54–71. Springer, Heidelberg (Oct 2014) 2, 5
46. Tsabary, R.: An equivalence between attribute-based signatures and homomorphic signatures, and new constructions for both. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017, Part II. LNCS, vol. 10678, pp. 489–518. Springer, Heidelberg (Nov 2017) 2, 3, 5
47. Waters, B., Wu, D.J.: Batch arguments for NP and more from standard bilinear group assumptions. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part II. LNCS, vol. 13508, pp. 433–463. Springer, Heidelberg (Aug 2022) 6
48. Zhang, Y., Zhao, J., Zhu, Z., Gong, J., Chen, J.: Registered attribute-based signature. In: Tang, Q., Teague, V. (eds.) PKC 2024, Part I. LNCS, vol. 14601, pp. 133–162. Springer, Heidelberg (Apr 2024) 6