Versatile ABS: Usage Limited, Revocable, Threshold Traceable, Authority Hiding, Decentralized Attribute Based Signatures

Osman Biçer
Koç University

Alptekin Küpçü
Koç University

March 13, 2019

Abstract

In this work, we revisit multi-authority attribute based signatures (MA-ABS), and elaborate on the limitations of the current MA-ABS schemes to provide a hard to achieve (yet very useful) combination of features, i.e., decentralization, periodic usage limitation, dynamic revocation of users and attributes, reliable threshold traceability, and authority hiding. In contrast to previous work, we disallow *even the authorities* to de-anonymize an ABS, and only allow joint tracing by threshold-many tracing authorities. Moreover, in our solution, the authorities cannot sign on behalf of users. In this context, first we define a useful and practical attribute based signature scheme (versatile ABS or VABS) along with the necessary operations and security games to accomplish our targeted functionalities. Second, we provide the first VABS scheme in a modular design such that any application can utilize a subset of the features endowed by our VABS, while omitting the computation and communication overhead of the features that are not needed. Third, we prove the security of our VABS scheme based on standard assumptions, i.e., Strong RSA, DDH, and SDDHI, in the random oracle model. Fourth, we implement our signature generation and verification algorithms, and show that they are practical (for a VABS with 20 attributes, Sign and Verify times are below 1.2 seconds, and the generated signature size is below 0.5 MB).

Keywords— attribute based signature, anonymous credentials, threshold cryptography.

1 Introduction

Attribute based signatures (ABS) [1, 2, 3, 4, 5, 6, 7, 8] are developed for non-interactively proving satisfiability of a Boolean attribute policy as a signer of a message using the attribute tokens that the user has obtained from an attribute authority. The goal is to disclose only suitability to the given policy while protecting the privacy of the rest (e.g., identity of the signer, her other attributes, her other generated signatures). An ABS should also protect against collusion of users to combine their attributes, and defend against forging attempts of signatures without having the required attribute tokens in the policy. Although the initially considered application of ABS was attribute based messaging [9, 10], it has emerged to be a useful primitive applicable to trust negotiation [11], etc. To compare it with similar constructions (e.g., mesh signatures [12] and anonymous credentials [13]), we refer to [1].

Due to the lack of support for multiple attribute authorities in the initially developed ABS schemes, multi-authority ABS (MA-ABS) scheme of [2] was proposed. However, this scheme required a central master authority, so the decentralized ABS scheme of [3], which does not require a master authority and is more scalable, was developed. As a further contribution, [4] proposed the first decentralized traceable ABS scheme that allows a tracing authority to detect and "prove" to a judge the signer of an ABS. Unfortunately, [4] allows each attribute authority that a signer has interacted with to obtain her secret token, and thus forge a signature traceable to her. [7] elaborates on this issue, and provides "non-frameability" and "tracing soundness" notions (the latter has been previously defined in the context of group signatures by [14]), as well as a generic construction satisfying these notions. However, the proposed constructions still allow any malicious tracing authority to disclose an honest signer, whereas our solution requires threshold-many authorities to be corrupted to launch such an attack.

In this paper, we confront the shortcomings of existing multi-authority and decentralized ABS schemes (altogether we refer by MA-ABS schemes), and achieve the following properties:

- *Periodic usage limitation.* The ability to limit the number of verifying signatures of a user (per verifier) in a given time period. Note this notion differs from controllable linkability of [5, 8, 15], as the former

MA-ABS	Decentralization	Usage Limitation	Traceability	Threshold Traceability	Reliable Traceability	Attribute Revocation	User Revocation	Authority Hiding
[2]	✗	✗	✗	✗	✗	✗	✗	✗
[3]	✓	✗	✗	✗	✗	✗	✗	✗
[4]	✓	✗	✓	✗	✗	✗	✗	✗
[7]	✓	✗	✓	✗	✓	✗	✗	✗
VABS	✓*	✓	✓	✓	✓	✓	✓	✓

Table (1) Comparison of our VABS with the existing MA-ABS schemes. ✓ and ✗ denote whether the corresponding scheme has this feature or not, respectively. By ✓*, we refer to the fact that our scheme is somewhat decentralized (or scalable multi-authority) by allowing many identity providers (e.g., master authorities).

merely limits the number of signatures that gets verified by a verifier to a verifier-determined bound, while the latter provides opportunity for linking between signatures generated by the same user.

- *Threshold traceability.* The requirement that a predefined number of tracing authorities should collaborate to output the identity of the signer of a signature. This is important to eliminate the possibility of a corrupted tracing authority de-anonymizing a signer without a rightful purpose.
- *Reliable traceability.* This includes inability to forge a person's signature (even by the authorities) and ability of threshold-many honest tracing authorities to always find the original signer. Inability to forge a person's signature also supports our periodic usage limitation goal, since if a malicious party forges a signer's signature, it consumes her rights to honestly generate verifying signatures. Note that this notion also covers "traceability" of [4] and "non-frameability" of [7].
- *Dynamic revocation of attributes.* A useful property for dynamically revoking the attributes of signers that no longer deserves them by the attribute issuing authorities (e.g., a student graduates).
- *Dynamic revocation of users.* A useful property, especially to dynamically detach detected malicious users or criminals from the ability of generating verifiable ABSs by a specialized authority. Similarly, this may be employed when a user's key is stolen.
- *Authority hiding.* This feature is important for further anonymity of a signer by hiding the authorities that she has interacted with, in applications where an attribute can be issued by multiple authorities (e.g., if each school can provide a student attribute, just by learning the authority that provided the attribute to the user, her privacy is partly invaded).

We named our scheme that satisfies these novel, non-trivial, and seemingly conflicting aspects as versatile ABS, or shortly VABS. Table 1 compares the features of existing MA-ABS schemes and our VABS.

1.1 Related Work

Anonymous credential schemes. Anonymous credential schemes [16, 17, 18, 19, 20, 21] can be utilized for proving the conformation to some attribute policy while keeping the anonymity. Schemes providing n-times unlinkability seem especially suitable [16, 21]. In our solution, we build on top of [16] and improve it on many aspects to get a VABS that satisfies our requirements. However, we could not use [21] for this purpose, due to the fact that it allows that the authorities to obtain the secret keys of the users (and hence sign on their behalf).

Attribute based signature (ABS) schemes. We already mentioned ABS schemes [1, 2, 3, 4, 5, 7, 6, 8, 15], which are non-interactive signature solutions for anonymous attribute policy proving as a signature. The existing multi-authority and decentralized ABS schemes of [2, 3, 4, 7] lacks our mentioned goals. Although there exists some standard credential revocation techniques [22, 23] that can possibly be applied to these schemes, to modify these schemes for the requirements of a VABS is a not trivial task.

Functional credential schemes. The functional credential scheme of [24] can be utilized for anonymously proving conformity to an attribute policy to the third parties. Unfortunately, the number of authentication attempts in this scheme cannot be bounded by a fixed value for limited use. Moreover, there seems no effective way of multi-authority attribute issuing.

Group signatures. Group signature schemes [25, 26, 27, 28, 29, 30, 31, 32] are non-interactive constructions for proving that the signer of a message belongs to some group (sharing a particular attribute). In particular, revocable [31, 32], or distributed traceable [30], or fully dynamic model of [33] may seem useful in construction of a VABS. On the other hand, again, the use of keys in those schemes cannot easily be bounded by a fixed value.

1.2 Our Contributions

In Section 3, we define the VABS scheme along with its authority architecture, operations and security requirements (i.e., anonymity, signature unforgeability, soundness, and reliable traceability). Our security definitions are novel by including the different authority architecture and allowing proposed operations, yet they are inline with previous definitions in similar paradigms.

In Section 4, we provide the first VABS protocol with modular design. In our construction, the desired features can be turned on/off to efficiently achieve the requirements of the application.

In Section 5, we prove the security of our VABS based on Strong RSA (via reduction to the security of CL signature [34]), DDH, and SDDHI (via reduction to the pseudo random function construction of [16]) assumptions in the random oracle model. In Section 6, we implement the basic features of our VABS scheme (without traceability, revocation, and authority hiding), and show their efficiency (i.e., for a VABS with upto 20 attributes to be proven, Sign and Verify spend below 1.2 second, and the generated signature size is below 0.5 MB).

2 Preliminaries

Notation. Throughout this paper,

- $a \twoheadleftarrow B$ denotes that the value of a is picked from the set B uniformly at random,
- $a \leftarrow \mathsf{B}$ denotes that a is set as the output returned by a probabilistic polynomial time (PPT) B,
- $a := b$ denotes that the value of a is set as the value of b,
- $\mathsf{A}(a) \rightarrow b$ denotes that a PPT A is takes as input a and its output is called b.
- $\mathsf{A}\{B_1(b_1), B_2(b_2)\} \rightarrow (c_1), (c_2)$ denotes that A is a protocol executed between parties B_1 with input b_1 and B_2 with input b_2. At the end, party B_1 obtains output c_1 and party B_2 obtains output c_2.
- $\boxed{a}$ denotes that a is an optional step only for achieving the user/attribute revocation feature.
- $\textcircled{a}$ denotes that a is an optional step only for achieving the reliable traceability feature.
- λ denotes a security parameter, n denotes the number of uses allowed in a time period, and $\mathcal{QR}_{\mathsf{N}}$ denotes the set of quadratic residues modulo N.

Zero-Knowledge Proofs. In our protocols, we utilize non-interactive zero knowledge proofs of knowledge (NIZKPoK) obtained via the Fiat-Shamir transformation [35] of zero knowledge proof of knowledge (ZKPoK) protocols. The Fiat-Shamir transformation of ZKPoK protocols can also be utilized to sign messages, in which case we call them signatures of knowledge (SoK). For NIZKPoKs and SoKs, we utilize the notation of Camenisch and Stadler [36]: $\text{NIZKPoK}\{(a, b) : C = g^a h^b\}$ denotes a NIZKPoK of private values a and b that satisfy $C = g^a h^b$ against public C, g, h, and $\text{SoK}[m]\{(a, b) : C = g^a h^b\}$ denotes SoK of a and b that satisfy $C = g^a h^b$ on public C, g, h and a public message m.

For OR proofs, we make use of zero-knowledge OR proofs realizable by the generic scheme of [37]. This scheme provides an efficient method for proving knowledge of solutions for a-out-of-d problems, without revealing the subset of the problems whose solutions are known. The other example ZKPoK schemes that can be used in the realization of our constructions are [38] (for ZKPoK of factorization of a strong RSA modulus), [13] (for ZKPoK that some numbers are quadratic residue of a strong RSA modulus), [39] (for ZKPoK that a committed value is in a given range), [34] (for ZKPoK of a CL signature), [40] (for ZKPoK of a discrete logarithm and opening values of Pedersen commitments [41]).

Pseudo Random Functions (PRFs). In the PRF security game $\mathsf{DistPRF}$, the adversary is given a unary security parameter 1^λ and oracle access to either $F_s(\cdot)$ or $f(\cdot)$ (chosen fairly at random), where f is a random function and F is a PRF family. The adversary wins by correctly guessing whether the oracle is $F_s(\cdot)$ or $f(\cdot)$. We say F is a PRF family, if for all PPT adversaries $\mathcal{A}$, for randomly chosen s, there exists a negligible function $\mathsf{n}(\cdot)$ such that

$$\mathsf{Pr}[\mathcal{A} \text{ wins } \mathsf{DistPRF}] \leq \frac{1}{2} + \mathsf{n}(\lambda)$$

For simplicity, we call $F_s(\cdot)$ a PRF. Note that [16] shows that the function $F_{g,s}(\cdot) = \mathbf{g}^{1/(s+\cdot)}$ is a PRF (where g is a random generator of a generic group $\mathbb{G}$ of prime order q and $s \twoheadleftarrow \mathbb{Z}_q^*$), if SDDHI assumption [16] holds in $\mathbb{G}$. We refer the reader for the further details to [16].

Periodic n-Times Unlinkable Anonymous Credential Scheme of [16] We now briefly describe the n-times unlinkable anonymous credential scheme of [16] that we build upon. Let $\ell_q \in \Theta(\lambda)$, ℓ_x, ℓ_{time}, and ℓ_{cnt} be system parameters satisfying $\ell_q \geq \ell_x \geq \ell_{time} + \ell_{cnt} + 2$ and $2\ell_{cnt} - 1 > n$ as in [16]. The attribute issuer generates a cyclic group $\langle g \rangle = \mathbb{G}$ of prime order q such that $2^{\ell_q - 1} < q < 2^{\ell_q}$. It also generates another generator h of $\mathbb{G}$ and a cyclic group $\langle \mathbf{g} \rangle = \langle \mathbf{h} \rangle = \mathbf{G}$ of composite order $\mathsf{p'q'}$ where $\mathbf{g}$ and $\mathbf{h}$ are quadratic residues modulo $\mathsf{N} = (2\mathsf{p'} + 1)(2\mathsf{q'} + 1)$. Moreover, the issuer also generates a CL signature [34] key pair (p, s) within the group $\mathbf{G}$. It publishes its public key $(g, h, \mathbf{g}, \mathbf{h}, \mathbf{G}, p)$, and the zero-knowledge proof that N is a special RSA modulus

[38] and that $\langle \mathbf{g} \rangle = \langle \mathbf{h} \rangle$ are quadratic residues modulo N [13]. To obtain a credential, a user first interacts with the issuer, and they together run the following protocol in a mutually authentic channel:

1. The user generates a key pair $(s_1, p_1 := g^{s_1})$.
2. The user picks $s'_2 \twoheadleftarrow \mathbb{Z}_q$ and computes the Pedersen commitment $\mathsf{C}_{s_1+s'_2}$ to $s_1 + s'_2$. [41]. She then sends $\mathsf{C}_{s_1+s'_2}$ to the issuer and proves that it is correctly formed via [40].
3. The issuer picks $\rho \twoheadleftarrow \mathbb{Z}_q$ and sends it to the user. Both parties compute $\mathsf{C}_{s_1+s'_2+\rho} := \mathsf{C}_{s_1+s'_2} g^{\rho}$. The user sets $s_2 := s'_2 + \rho$.
4. The parties run the signature on a committed value protocol proposed in [34] using $\mathsf{C}_{s_1+s_2}$. At the end, the user obtains a CL signature σ of the issuer on the user secret keys s_1 and s_2.

The user can then prove the issued credential anonymously to a verifier n-times in a given time period t via the following protocol.

1. The verifier sends to the user a value $R \twoheadleftarrow \mathbb{Z}_q^*$.
2. The user computes and sends to the verifier the serial number $S = g^{1/(s_1+t2^{\ell_{cnt}}+J))}$ and the double spending tag $E = p_1 g^{R/(s_2+2^{\ell_{cnt}+\ell_{time}}+t2^{\ell_{cnt}}+J))}$ where J is the number of times that the user has authenticated her credential in the current time period $t \geq 1$.
3. The user and the verifier run ZKPoK protocols for s_1, s_2, σ, and J such that $0 \leq J < n$, $S = g^{1/(s_1+t2^{\ell_{cnt}}+J))}$, $E = p_1 g^{R/(s_2+2^{\ell_{cnt}+\ell_{time}}+t2^{\ell_{cnt}}+J))}$, and σ verifies on the user secret s with the issuer public key p.

3 Definition of Versatile ABS

3.1 System Model

In our system, there are *users*, who can be signers or verifiers, and *authorities*. We have three type of authorities: *identity providers*, *attribute authorities*, and *tracing authorities*, as explained below.

The system model and authority architecture of our VABS definition differs from existing multi-authority and decentralized ABS schemes [2, 3, 4], as we assume each state (or region) has an *identity provider* that is an authority with the top role in hierarchy (e.g., civil registry authority providing national identity numbers), and is responsible for issuing/revoking global ids and ensuring that a user picks her secret key randomly (to prevent attribute combination between users). In our scheme, if an identity provider issues the same secret key (and hence the same public key) to multiple users, this will be detected by tracing authorities, who are described below. When joining the system, each user obtains her global identity from a single identity provider.

There are arbitrary number of attribute authorities, each responsible for issuing various attributes to the deserving parties. We only trust these authorities for proper conduct of the above-mentioned tasks and for non-disclosure of the users that have interacted with them. Yet, even with the information resulting from those interactions, none of the identity providers or attribute authorities can de-anonymize a signer from a given signature or can sign on behalf of a user. The only malicious actions that those authorities may take are issuing tokens to the undeserving users and revealing the identity of the users that applied to them for tokens (which are concerns in any such scheme, and not related to our construction).

Also, there exist ϕ tracing authorities in our model, θ of which can de-anonymize the signer of a given VABS. This is provided to ensure lawful de-anonymization purposes, and to resist corruption of tracing authorities, as long as the adversary corrupts at most $\theta - 1$ of them. Note that we also require $\phi > 2(\theta - 1)$, so the majority of the tracing authorities are assumed to be honest. As long as this assumption holds, they can always trace and output the correct signer of a VABS in lawful cases. Even when all the tracing authorities are corrupt, they cannot sign on behalf of a user. As an additional duty, tracing authorities can detect malicious behaviour of corrupt identity providers colluding with users for issuing the same secret key to multiple of them.

3.2 Operations

In what follows, we define the components (GlobalSetup, TraceSetup, IdPJoin, AuthJoin, UserJoin, TraceIssue, AttrIssue, Sign, Verify, UserRevoke, AttrRevoke, Trace) of a VABS scheme. Note that a basic scheme should define (GlobalSetup, IdPJoin, AuthJoin, UserJoin, AttrIssue, Sign, Verify), whereas (TraceSetup, TraceIssue, Trace, UserRevoke, AttrRevoke) are optional components of a VABS.

GlobalSetup$(1^{\lambda}) \rightarrow params$: This is an operation that takes place once in the setup phase of the scheme. It takes as input a unary security parameter λ. It outputs global setup parameters $params$. Note that depending on the application, this can be run as an algorithm by a trusted party or as a multi-party protocol to avoid single point of failures.

$\mathsf{TraceSetup}(params) \rightarrow ((tp_1, ts_1), \ldots, (tp_\phi, ts_\phi))$: This is an operation that takes place once in the setup phase. It takes as input the global setup parameters $params$. It outputs the tracing key pair (tp_i, ts_i) for each tracing authority i, such that at least θ authorities are needed to trace a signer of a given VABS.

$\mathsf{IdPJoin}(params) \rightarrow (iis, iip, irs, irp_0, sk_{id}, pk_{id})$: This is an algorithm that is executed by each identity provider when it joins to the system. The algorithm takes as input the global setup parameters $params$. It outputs an identity provider issuing secret and public key pair (iis, iip), its revocation secret key irs, its initial revocation public key irp_0, and its digital signature key pair (sk_{id}, pk_{id}).

$\mathsf{AuthJoin}(params) \rightarrow (ais, aip, ars, arp_0)$: This is an algorithm that is executed by each attribute authority when it joins to the system. The algorithm takes as input the global setup parameters $params$. It outputs an authority issuing secret and public key pair (ais, aip), its revocation secret key ars, and initial revocation public key arp_0.

$\mathsf{UserJoin}\{\text{User}(iip, irp_i, params), \text{Identity provider}(iis, iip, irp_i, sk_{id}, pk_{id}, params)\} \rightarrow (s, \sigma_{id}, cert, w_{id}), (irp_{i+1}, \tilde{g\sigma}_{id})$: This is a two-party protocol between a user and an identity provider that takes place when the user joins the system. The identity provider starts by knowing the global setup parameters $params$, its issuing secret and public key pair (iis, iip), its current revocation public key, and its digital signature key pair (sk_{id}, pk_{id}), while the user knows iip, irp_i and $params$. The protocol outputs to the user her secret key s, her global id σ_{id}, a certificate $cert$ that includes her personal user identity uid and some additional information (e.g., validity period), and a witness w_{id} for non-revocation, and to the identity provider its next revocation public key irp_{i+1} and a revocation handle $\tilde{\sigma}_{id}$ for σ_{id}.

$\mathsf{TraceIssue}\{\text{User}(s, cert, tp, w, irp, params), \text{Tracing Authority}(ts, tp, pk_{id}, irp, params)\} \rightarrow (\sigma_T), (\perp)$: This is the tracing issue protocol that a user should run with each of θ tracing authorities before being able generate traceable signatures. The tracing authority starts by knowing the global setup parameters $params$, its tracing secret and public key pair (ts, tp), the identity provider's (i.e., the one that the user interacted with) digital signature public key pk_{id} and current revocation public key irp, while the user knows her secret key s, her certificate $cert$, tp, the non-revocation witness w_{id} of her σ_{id}, irp, and $params$. The protocol outputs to the user her tracing token σ_T.

$\mathsf{AttrIssue}\{\text{User}(s, cert, \omega, aip, arp_j, w_{id}, irp, params), \text{Attribute Authority}(ais, aip, \omega, pk_{id}, arp_j, irp, params)\} \rightarrow (\sigma_\omega, w), (arp_{j+1}, \tilde{\sigma}_\omega)$: This is the attribute issuing operation executed by an authority and a user for each attribute token that will be issued to a user. As inputs, the authority starts by knowing the global setup parameters $params$, its issuing secret and public key pair (ais, aip), the attribute ω to be given to the user, its current revocation public key arp_j, the identity provider's (i.e., the one that the user interacted with) digital signature public key pk_{id} and its current revocation public key irp, while the user knows a user secret key s, her user certificate $cert$, ω, aip, arp_j, the non-revocation witness w_{id} of her global id σ_{id}, irp, and the global setup parameters $params$. The protocol outputs to the user an attribute token σ_ω and the non-revocation witness w_ω for the attribute, and to the authority its next revocation public key arp_{j+1} and a revocation handle $\tilde{\sigma}_\omega$ for the issued attribute token.

$\mathsf{Sign}(m, s, \beta, \sigma_{id}, w_{id}, \Sigma_\beta, W_\beta, \Sigma_T, IIP, IRP, AIP, ARP, TP_\phi, t, J, n, params) \rightarrow \mu$: This is the signing algorithm that is executed by a user. It takes as input a message m, a secret key s, an attribute policy β, a global id σ_{id}, a non-revocation witness w_{id} for σ_{id}, an attribute token set Σ_β that proves that the owner of s conforms to the attribute policy β, a set W_β of non-revocation witness of those attributes, a set Σ_T of the tracing tokens, the issuing public key set IIP and the revocation public key set IRP of all identity providers, the issuing public key set AIP and the revocation public key set ARP of all authorities that can issue the attributes in β, the tracing public key set TP_ϕ of all tracing authorities, the a time period indicator t, a signature counter J, the allowed number n of the legitimate signatures by a user in a time period, and the global setup parameters $params$. The output of the algorithm is a VABS μ (including the tracing tag Φ) on m. At the end of each algorithm run, the user increments the counter J for validness of her next signature.

$\mathsf{Verify}(m, \mu, \beta, IIP, IRP, AIP, ARP, TP_\phi, t, SDB_j, n, params) \rightarrow (b, SDB_{j+1})$: This is the verification algorithm that is executed by a verifier for a given signature. It takes as input a message m, a VABS μ, a policy β, the issuing public key set IIP and the revocation public key set IRP of all identity providers, the issuing public key set AIP and the revocation public key set ARP of all authorities that can issue the attributes in β, the tracing public key set TP_ϕ of all tracing authorities, a time period indicator t, the current signature database SDB_j, the allowed number n of the legitimate signatures by a user in a time period, and the global setup parameters $params$. The output of the algorithm is a bit b. b is defined as 1, if the user's global id is still valid (not revoked), the user has executed the $\mathsf{TraceIssue}$ operation with θ tracing authorities, signed attributes in μ conform to β under AIP and has not been revoked in ARP, and there are no more than $n-1$ signatures produced with the same key that is used to sign m in the time period according to the SDB_j. Otherwise, it is defined as 0. If the algorithm output is 1, the signature database is updated by the addition of μ (i.e., $SDB_{j+1} := SDB_j || \mu$). We usually make the database update output implicit.

$\mathsf{UserRevoke}(irs, irp_j, \tilde{gid}, params) \rightarrow irp_{j+1}$: This is the user revocation algorithm that is executed by an identity provider for revoking a user from the system completely by making her global id invalid. It takes as input the revocation secret key irs and the current revocation public key irp_i, the user's revocation handle $\tilde{\sigma}_{id}$, and the global setup parameters $params$. It outputs the new revocation public key irp_{j+1}.

$\mathsf{AttrRevoke}(ars, arp_j, \tilde{\sigma}_\omega, params) \rightarrow arp_{j+1}$: This is the revocation algorithm that is executed by an authority for revoking an attribute of a user. It takes as input the revocation secret key ars and the current revocation public key arp_j, the user's revocation handle $\tilde{\sigma}_\omega$ for the attribute ω, and the global setup parameters $params$. It outputs the new revocation public key arp_{j+1} of the authority.

$\mathsf{Trace}\{\textbf{for } i = 1, \ldots, 2\theta - 1, \text{ Tracing Authority}_i(\mu, ts_i, tp_i, TDB, params)\} \rightarrow uid$: This is the tracing protocol that is executed by $2\theta - 1$ tracing authorities to reveal the signer of a VABS. Each tracing authority i starts by knowing the global setup parameters $params$, a VABS μ, its own tracing secret and public key pair (ts_i, tp_i), and the shared tracing database TDB. The protocol outputs to each honest tracing authority the uid of the signer of μ.

3.3 Security Definitions

A VABS scheme $\Pi = (\mathsf{GlobalSetup}, \mathsf{TraceSetup}, \mathsf{IdPJoin}, \mathsf{AuthJoin}, \mathsf{UserJoin}, \mathsf{TraceIssu}, \mathsf{AttrIssue}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{UserRevoke}, \mathsf{AttrRevoke}, \mathsf{Trace})$ must satisfy anonymity, signature unforgeability, signature soundness, and tracing reliability as security requirements, all of which we define in this section.

In all of the game-based definitions in this section, given a VABS scheme Π, a PPT adversary $\mathcal{A}$, a challenger $\mathcal{C}$, a unary security parameter 1^λ, and a limit n for a time period duration δ, the first 2 steps of the security games are as follows.

1. $\mathcal{C}$ runs $\mathsf{GlobalSetup}$ and obtains the global setup parameters $params$. $\mathcal{C}$ then gives 1^λ, n, δ, and $params$ to $\mathcal{A}$. $\mathcal{A}$ is allowed to generate $\theta - 1$ fully malicious tracing authorities but is required to give their public keys to $\mathcal{C}$. $\mathcal{C}$ generates the rest of the tracing authorities so that $\phi - \theta + 1$ of them honestly follow the protocol. All tracing authorities together run $\mathsf{TraceSetup}$. If $\mathcal{C}$ can detect any malicious behaviour by the tracing authorities under $\mathcal{A}$'s control during this setup phase, the game terminates and the game's output is defined as 0. The time period counter t is initialized as $t := 1$, and is started.
2. In any step,
   (a) $\mathcal{A}$ can generate polynomially-many identity provider and attribute authority keys, but is required to give the public keys to $\mathcal{C}$. $\mathcal{A}$ can also ask $\mathcal{C}$ to generate identity providers or attribute authorities. Then, $\mathcal{C}$ would honestly generate them, and share their public keys with $\mathcal{A}$.
   (b) $\mathcal{A}$ can generate polynomially-many users under its control or can ask $\mathcal{C}$ to generate honest users, and has the ability to run for those users $\mathsf{UserJoin}$ with any identity provider or $\mathsf{AttrIssue}$ with any attribute authority. For all of the users generated, $\mathcal{A}$ can request $\mathsf{TraceIssue}$ to be run. $\mathcal{A}$ can demand revocation of identity or any attribute belonging to any of the generated users.
   (c) $\mathcal{A}$ can adaptively request $\mathcal{C}$ to sign any message as any user under her control with any attribute policy that the user satisfies.
   (d) $\mathcal{A}$ can adaptively request jointly running the Trace protocol with tracing auhorities under the control of $\mathcal{C}$ given any VABS as input.
   (e) If $\mathcal{A}$ outputs any string to $\mathcal{C}$ other than what is explicitly expected from it in the protocol, then $\mathcal{C}$ just ignores it.

Our signature unforgeability and anonymity definitions are as strong as the ones of [4] (i.e., "strong full unforgeability" and "anonymity"), and the differences are due to having a different authority architecture and adding the revocation feature.

Anonymity of a user is defined using, the following anonymity game $\mathsf{VABSAnonym}_{\mathcal{A},\Pi}(\lambda)$:

3. $\mathcal{A}$ gives $\mathcal{C}$ an attribute policy β.
4. $\mathcal{C}$ generates two secret keys s_0 and s_1. $\mathcal{C}$ runs $\mathsf{UserJoin}$, $\mathsf{TraceIssue}$, and $\mathsf{AttrIssue}$ for s_0 and s_1 with authorities of $\mathcal{A}$'s choice. Note that $\mathcal{C}$ should obtain attribute tokens on s_0 and s_1 such that both users conform to β.
5. $\mathcal{A}$ is given access to the signature oracles of both users:
   $\mathsf{Sign}(\tilde{m}, s_0, \tilde{\beta}, \sigma_{id,0}, w_{id,0}, \tilde{\Sigma}_{\tilde{\beta},0}, \tilde{W}_{\tilde{\beta},0}, \Sigma_{T,0}, IIP, IRP, \tilde{AIP}, \tilde{ARP}, TP_\phi, t, J_0, n, params)$ and
   $\mathsf{Sign}(\tilde{m}, s_1, \tilde{\beta}, \sigma_{id,1}, w_{id,1}, \tilde{\Sigma}_{\tilde{\beta},1}, \tilde{W}_{\tilde{\beta},1}, \tilde{\Sigma}_{T,1}, IIP, IRP, \tilde{AIP}, \tilde{ARP}, TP_\phi, t, J_1, n, params)$; where $(\tilde{m}, \tilde{\beta})$ is chosen by $\mathcal{A}$ as part of its queries; $\sigma_{id,b}$, $w_{id,b}$, and $\Sigma_{T,b}$ are the global id, its non-revocation witness, and the tracing tokens for s_b; IIP, IRP, and TP_ϕ are issuing and revocation public keys of identity providers and the public keys of the tracing authorities (respectively); the value t is the current time, and $0 \leq J_0, J_1 < n$ are the signature counters (incremented with each signing within t and reset between time periods) for the corresponding signing key. Other values with tilde sign are chosen by $\mathcal{C}$ as an honest signer would do.
   Essentially, $\mathcal{C}$ sets $\tilde{AIP}$ and $\tilde{ARP}$ as the public keys of the authorities that can issue attributes in $\tilde{\beta}$, $\tilde{\Sigma}_{\beta,0}$ and $\tilde{W}_{\beta,0}$ as the attribute tokens and non-revocation witnesses that are issued to s_0 related to β, $\tilde{AIP}$ and $\tilde{ARP}$ as issuing and revocation public keys of all authorities known to $\mathcal{C}$ that can issue the attributes

in $\tilde{\beta}$. Each oracle stops responding to further queries during that time period when its counter, J_0 or J_1, reach $n-1$.

6. $\mathcal{A}$ gives two messages m_0 and m_1 to $\mathcal{C}$.
7. $\mathcal{C}$ picks a random bit b. $\mathcal{C}$ signs both messages as s_b by running $\mu_0 \leftarrow \mathsf{Sign}(m_0, s_b, \beta, \sigma_{id,b}, w_{id,b}, \Sigma_{\beta,b}, W_{\beta,b}, \Sigma_{T,b}, IIP, IRP, AIP, ARP, TP_\phi, t, J_b, n, params)$ and $\mu_1 \leftarrow \mathsf{Sign}(m_1, s_{\bar{b}}, \beta, \sigma_{id,\bar{b}}, w_{id,\bar{b}}, \Sigma_{\beta,\bar{b}}, W_{\beta,\bar{b}}, \Sigma_{T,\bar{b}}, IIP, IRP, AIP, ARP, TP_\phi, t, J_{\bar{b}}, n, params)$, where AIP and ARP are issuing and revocation public keys of all authorities known to $\mathcal{C}$ that can issue the attributes in β, and Σ_β and $W_{\beta,b}$ are the set of attribute tokens their non-revocation witnesses for s_b for proving β. The other values are set as before. $\mathcal{C}$ then gives μ_0 and μ_1 to $\mathcal{A}$.
8. $\mathcal{A}$ eventually returns a bit b'. The output of the game is defined as 1 (i.e., $\mathcal{A}$ wins), if $b = b'$ and $\mathcal{A}$ has not asked any tracing authority under $\mathcal{C}$'s control for participation in Trace of μ_0 or μ_1. Otherwise, the output of the game is defined as 0 (i.e., $\mathcal{A}$ loses).

Definition 1 (Anonymity). *A VABS scheme Π provides* anonymity*, if $\forall n, \delta \in poly(\lambda)$, for each PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{n}(\cdot)$ such that*

$$\mathsf{Pr}[\mathsf{VABSAnonym}_{\mathcal{A},\Pi}(\lambda) = 1] \leq \frac{1}{2} + \mathsf{n}(\lambda)$$

For our **signature unforgeability** definition, consider the following signature unforgeability game $\mathsf{VABSForge}_{\mathcal{A},\Pi}(\lambda)$:

3. $\mathcal{A}$ is allowed to fully corrupt all ϕ tracing authorities.
4. $\mathcal{A}$ returns an attribute policy an attribute ω to $\mathcal{C}$. ω must not be queried before as $\mathsf{AttrIssue}$ to the authorities under $\mathcal{C}$'s control for users under $\mathcal{A}$'s control.
5. $\mathcal{A}$ is restricted in that $\mathcal{A}$'s queries to an authority oracle as $\mathsf{AttrIssue}$ with ω for users under $\mathcal{A}$'s control, the attribute is issued but is *immediately revoked*[1].
6. $\mathcal{A}$ eventually generates a message m (not queried for signing to the users under $\mathcal{C}$'s control for the attribute policy $\beta \wedge \omega$) and a VABS μ where for any policy β. The output of the game is defined as 1 (i.e., $\mathcal{A}$ wins), if $\mathsf{Verify}(m, \mu, \beta \wedge \omega, IIP, IRP, AIP, ARP, TP_\phi, t, SDB, n, params) = 1$, where IIP, IRP, AIP, ARP, and TP_ϕ are the issuing and revocation public keys of all identity providers, the issuing and revocation public keys of all attribute authorities that can issue the attributes in $\beta \wedge \omega$, and the public keys of all tracing authorities known to $\mathcal{C}$. Otherwise, the output of the game is defined as 0 (i.e., $\mathcal{A}$ loses). Note that this step enforces $\mathcal{A}$ to ask to $\mathcal{C}$ for generation of at least one authority oracle to check for ω.

Definition 2 (Signature Unforgeability). *A VABS scheme Π provides* signature unforgeability*, if $\forall n, \delta \in poly(\lambda)$, for each PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{n}(\cdot)$ such that*

$$\mathsf{Pr}[\mathsf{VABSForge}_{\mathcal{A},\Pi}(\lambda) = 1] \leq \mathsf{n}(\lambda)$$

The **soundness** definition that we provide is similar to the soundness definition of [16], but differs slightly due to its usage, i.e., ours limits the total number of signatures by a party that gets verified, while the latter limits credential proofs *per attribute*. Formally, consider the following soundness game $\mathsf{VABSSound}_{\mathcal{A},\Pi}(\lambda)$:

3. $\mathcal{A}$ is allowed to fully corrupt all ϕ tracing authorities.
4. The output of the game is defined as 1 (i.e., $\mathcal{A}$ wins), if $\mathcal{A}$ generates at least $n+1$ message and VABS pairs within some time period for the same user s such that honest executions of the Verify algorithm on at least $n+1$ of those pairs within the same time period output 1. Otherwise, the output of the game is defined as 0 (i.e., $\mathcal{A}$ loses).

Definition 3 (Soundness). *A VABS scheme Π provides* soundness*, if $\forall n, \delta \in poly(\lambda)$, for each PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{n}(\cdot)$ such that*

$$\mathsf{Pr}[\mathsf{VABSSound}_{\mathcal{A},\Pi}(\lambda) = 1] \leq \mathsf{n}(\lambda)$$

For our **reliable traceability** definition, the idea is to ensure that tracing is correctly done as long as the adversary controls at most $\theta - 1$ tracing authorities. The reliable traceability notion covers "traceability" of [4] and "non-frameability" of [7], as long as $\theta - 1$ tracing authorities are honest. Our notion implies "tracing soundness" of [7] by requiring an exactly single original signer per VABS that will be deterministically traced. We note that this is the first work that provides a compact definition in the context of MA-ABS in the presence of multiple tracing authorities, although in the context of group signatures "traceability", "non-frameability", and "tracing soundness" definitions have been previously provided for distributed tracing[2] by [30]. Consider the following game $\mathsf{VABSTrace}_{\mathcal{A},\Pi}(\lambda)$:

[1] This is required to make sure that a user cannot use her revoked attributes

[2] Distributed tracing of [30] and our threshold tracing are different in that the former enforces all tracing authorities to join the Trace operation, while the latter enables a settable threshold-many of them to execute Trace

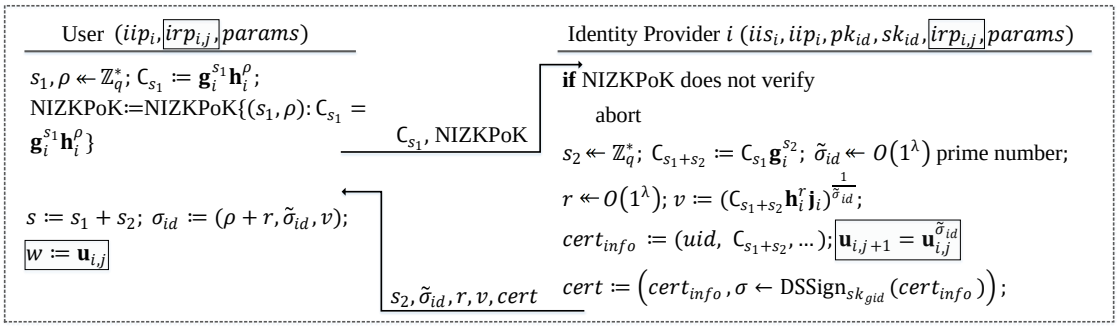

Figure (1) The UserJoin protocol between a new user and an identity provider.

3. The output of the game is defined as 1 (i.e., $\mathcal{A}$ wins), if for any VABS that verifies, at least θ tracing authorities (therefore necessarily including honest ones) do *not* output the *uid* that belongs to the original signer of the comprising VABS during any run of the Trace protocol. Otherwise, the output of the game is defined as 0 (i.e., $\mathcal{A}$ loses).

Definition 4 (Reliable Traceability)**.** *A VABS scheme* Π *provides* reliable traceability*, if* $\forall n, \delta \in poly(\lambda)$*, for each PPT adversary* $\mathcal{A}$*, there exists a negligible function* $\mathsf{n}(\cdot)$ *such that*

$$\mathsf{Pr}[\mathsf{VABSTrace}_{\mathcal{A},\Pi}(\lambda) = 1] \leq \mathsf{n}(\lambda)$$

4 Our Modular VABS

In this section, we present our VABS construction in a modular manner. We first start by a basic design that provides periodic usage limitation. Then, we add reliable threshold traceability, user/attribute revocation, and authority hiding features one by one. Depending on the application, any combination of these features can be used as part of our VABS construction.

4.1 Our Basic Decentralized ABS Scheme with Periodic Usage Limitation

In this subsection, we propose our basic solution for obtaining a usage-limited decentralized ABS scheme, which will be extended in later subsections to satisfy other VABS requirements. We build our scheme by proposing modifications, additions, and optimizations to [16]. However, note that [16] is only an n-times unlinkable anonymous authentication scheme utilizable for individual attributes, and targets cases where only one authority is present. In contrast, our scheme is an ABS scheme without double spending tags, and supports AND and OR operations in attribute policies.[3] In the following subsections, we further add many novel VABS properties to this basic solution.

We note that in the protocol and algorithm descriptions, parts that are inside rectangles are only utilized for user/attribute revocation, and parts that are within *rounded* rectangles are employed for reliable threshold traceability. Therefore, the reader may skip those for our basic solution.

Setup. In GlobalSetup, a cyclic group $\langle g \rangle = \mathbb{G}$ of prime order q such that $2^{\ell_q - 1} < q < 2^{\ell_q}$ is generated. Another generator h of $\mathbb{G}$ is also generated in a distributed computation [42, 43] so that $\log_g h$ would be intractable. Essentially, each party i (a user or an authority) that wants to join the generation process of h picks a random value $x_i \in \mathbb{Z}_q$. It then publishes $h_i := g^{x_i}$ and NIZKPoK$\{(x_i) : h_i = g^{x_i}\}$ with authentication tags.[4] At the end of the setup, all of the k parties involved compute $h := \prod_{i \in \{1,\ldots,k\}} g^{x_i}$ and output the parameters $(q, \mathbb{G}, g, h)$.

Authority Join. To join the system, each identity provider or attribute authority runs Algorithm 1. The proofs of knowledge for NIZKPoK$_1$ and NIZKPoK$_2$ can be instantiated as in [38] and [13], respectively, showing that the authority generated its parameters honestly. Upon executing the algorithm, the authority publishes its public key iip_i/aip_i together with the proofs NIZKPoK$_1$ and NIZKPoK$_2$, while keeping its secret key iis_i/ais_i. For efficiency in key management, we require that each attribute authority only has a fixed length public key, no matter how many different attributes it can issue. Additionally, each identity provider runs the

[3] In many cases, a NOT operation can easily be obtained via simple conversions (e.g., "Birth year NOT before 2000", could be converted to "Birth year after 1999") or can be separately obtained as an attribute from an authority. We highlight that existing decentralized schemes of [2, 3, 4] also do not have direct NOT operation support.

[4] The parties may be required to publish them on a public ledger, to maintain the consistency among parties and to thwart equivocation attempts.

Algorithm 1 Our IdPJoin/AuthJoin algorithm for the identity provider/attribute authority i

input: a unary security parameter 1^λ and global setup parameters $params = (q, \mathbb{G}, g, h)$.
output: an identity provider/attribute authority issuing public key $iip_i/aip_i = (\mathbf{g}_i, \mathbf{h}_i, \mathbf{j}_i, \mathsf{N}_i)$, its issuing secret key $iis_i/ais_i = (\mathsf{p}'_i, \mathsf{q}'_i)$, its initial revocation public key $irp_{i,0}/arp_{i,0} = (\mathbf{u}_{i,0}, \mathbf{g}'_i, \mathbf{h}'_i, M_i)$, its revocation secret key $irs_i/ars_i = (\mathsf{p}''_i, \mathsf{q}''_i)$, and the related proofs (NIZKPoK_1, NIZKPoK_2, NIZKPoK_3, NIZKPoK_4).

$\mathsf{p}'_i, \mathsf{q}'_i \twoheadleftarrow O(1^\lambda)$ Sophie Germain primes; $\mathsf{N}_i := (2\mathsf{p}'_i + 1)(2\mathsf{q}'_i + 1)$; $\mathbf{g}_i, \mathbf{h}_i, \mathbf{j}_i \leftarrow \mathcal{QR}_{\mathsf{N}_i}$
$\mathsf{p}''_i, \mathsf{q}''_i \twoheadleftarrow O(1^\lambda)$ Sophie Germain primes; $\mathsf{M}_i := (2\mathsf{p}''_i + 1)(2\mathsf{q}''_i + 1)$;
$\mathbf{u}_{i,0}, \mathbf{g}'_i, \mathbf{h}'_i \leftarrow \mathcal{QR}_{\mathsf{M}_i}$
$\text{NIZKPoK}_1 := \text{NIZKPoK}\{(\mathsf{q}'_i, \mathsf{p}'_i) : \mathsf{N}_i = (2\mathsf{p}'_i + 1)(2\mathsf{q}'_i + 1)\}$
$\text{NIZKPoK}_2 := \text{NIZKPoK}\{\mathbf{g}_i, \mathbf{h}_i, \mathbf{j}_i \in \mathcal{QR}_{\mathsf{N}_i}\}$
$\text{NIZKPoK}_3 := \text{NIZKPoK}\{(\mathsf{q}''_i, \mathsf{p}''_i) : \mathsf{M}_i = (2\mathsf{p}''_i + 1)(2\mathsf{q}''_i + 1)\}$
$\text{NIZKPoK}_4 := \text{NIZKPoK}\{\mathbf{u}_{0,i}, \mathbf{g}'_i, \mathbf{h}'_i \in \mathcal{QR}_{\mathsf{M}_i}\}$

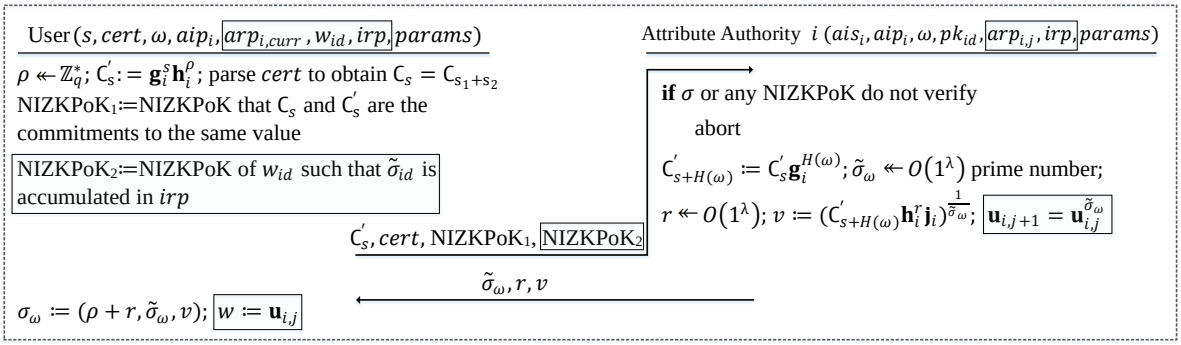

Figure (2) The AttrIssue protocol between a user and an attribute authority.

DSKeyGen algorithm of a conventional digital signature scheme (DSKeyGen, DSSign, DSVerify) to obtain a key pair (pk_{id}, sk_{id}) and publishes the associated public key.

User Join. To join the system, a user interacts with the identity provider in her state through a secure and authenticated channel, where they run the UserJoin protocol in Figure 1 (i.e., an extended version of the "Signature on a Committed Value" protocol of [34]). At the end of the protocol, the user obtains a certificate $cert$, which is composed of user identification information, the commitment C_s to her secret key, possibly expiration date and other information, together with the signature of the identity provider on them. The user uses the certificate $cert$ given by the identity provider to obtain tokens using the same secret key from all attribute authorities. The user also obtains σ_{id} as a CL signature on her secret key s, which is utilized each time she signs a message for the purpose of limiting the number of signatures within a time period. In contrast to [16], in our solution, a user has only one secret key s due to the removal of the double spending tag.

Attribute Issue. To obtain the token for an attribute ω from an authority i, through a secure and authenticated channel the user and the authority runs the protocol in Figure 2 (i.e., again an extended version of the "Signature on a Committed Value" protocol of [34]), where $H : \{0,1\}^* \to \mathbb{Z}_q$ is modeled as a random oracle. At a high level, the user obtains a CL signature on $s + H(\omega)$, after proving in zero knowledge that she holds the identity s to tie her attributes to her identity blindly. We enforce randomization of the attributes by $H(\omega)$ to prevent malicious collusion of users (see Lemma 2).[5] Note that in our basic scheme we trust identity providers for randomization, but if the tracing function is added as in Subsection 4.2, then the tracing authorities can detect this type of malicious behaviour.

Sign. We present our Sign algorithm with AND policy $\omega_1 \wedge \ldots \wedge \omega_k$ in Algorithm 2. The algorithm outputs a serial number S for the VABS, commitments C_J and C_s to the number of generated signatures in the current time period and the user's secret key, and a number of proofs for the correct construction of the signature and knowledge of CL signatures on s plus the randomized attribute. The proof of knowledge

[5] Unlike the ABS schemes of [2, 3, 4], we employ a hash function for randomization, instead of trusting authorities for that purpose. This is reasonable in scenarios where there exist multiple authorities and multiple attributes, and each attribute is not known from the start but rather dynamically established.

Algorithm 2 Our Sign algorithm with AND policy

input: a message m, a secret key s, an AND policy $\beta = \omega_1 \wedge \ldots \wedge \omega_k$, a global id σ_{id}, a non-revocation witness witness w_{id} for σ_{id}, a set $\Sigma_\beta = (\sigma_{\omega_1}, \ldots, \sigma_{\omega_k})$ of the attribute tokens for β, a set $W_\beta = (w_1, \ldots, w_k)$ of non-revocation witness of those attributes, a set $\Sigma_T = (\sigma_{T,1}, \ldots, \sigma_{T,\theta})$ of the tracing tokens, the issuing public key iip and the revocation public key irp of the provider of the σ_{id}, the issuing public key set AIP and the revocation public key set ARP of the authorities that have issued Σ_β, the tracing public key set TP of the tracing authorities that have issued Σ_T, the current time period $t \geq 1$, the number J of ABSs generated by the signer in the current period, the allowed number n of the legitimate signatures by a user in a time period, the signature and global setup parameters $params = (q, \mathbb{G}, g, h)$.

output: an ABS $\sigma = (S, \mathsf{C}_J, \mathsf{C}_s, \Phi, \mathrm{SoK}_1, \ldots, \mathrm{SoK}_{3+k}, \mathrm{SoK}_{4+k}, \ldots, \mathrm{SoK}_{5+k+\theta})$.

$\rho_1, \rho_2, \rho_3 \leftarrow \mathbb{Z}_q$; $S := g^{1/(s+t2^{\ell_{cnt}}+J)}$; $\mathsf{C}_J := g^J h^{\rho_1}$; $\mathsf{C}_s := g^s h^{\rho_2}$

$\Phi := (g_1^{\rho_3}, g_2^{\rho_3}, g_1^s h_1{}^{\rho_3}, c^{\rho_3} d^{\rho_3 \kappa})$ where $\kappa := H(g_1^{\rho_3}, g_2^{\rho_3}, g_1^s h_1{}^{\rho_3})$

$\mathrm{SoK}_1 := \mathrm{SoK}[m]\{(J, \rho_1) : J \in \{0, \ldots, n-1\} \wedge \mathsf{C}_J = g^J h^{\rho_1}\}$

$\mathrm{SoK}_2 := \mathrm{SoK}[m]\{(\alpha, \gamma) : S = g^\alpha \wedge g = (\mathsf{C}_s g^{t2^{\ell_{cnt}}} \mathsf{C}_J)^\alpha h^\gamma\}$

$\mathrm{SoK}_3 := \mathrm{SoK}[m]\{(s, \rho_2, \sigma_{id}, w_{id}) : \sigma_{id}$ is a σ_{CL} on s committed in C_s verifiable with iip, and w_{id} is a witness that $\tilde{\sigma}_{id}$ is accumulated in $irp\}$

for $i = 1, \ldots, k$ **do**

$\mathrm{SoK}_{3+i} := \mathrm{SoK}[m]\{(s, \rho_2, \sigma_{\omega_i}, w_i) : \sigma_{\omega_i}$ is a σ_{CL} on $s + H(\omega_i)$ committed in $\mathsf{C}_{s+H(\omega_i)} = \mathsf{C}_s g^{H(\omega_i)}$ verifiable with $aip_i \in AIP$, and w_i is a witness that $\tilde{\sigma}_{\omega_i}$ is accumulated in $arp_i \in ARP\}$

$\mathrm{SoK}_{4+k} := \mathrm{SoK}[m]$ that Φ is constructed correctly

$\mathrm{SoK}_{5+k} := \mathrm{SoK}[m]$ that C_s and $\dot{g}_1^s \dot{h}^{\rho_3}$ are commitments to the same value

for $i = 1, \ldots, \theta$ **do**

$\mathrm{SoK}_{5+k+i} := \mathrm{SoK}[m]\{(s, \rho_2, \sigma_{T,i}) : \sigma_{T,i}$ is a σ_{CL} on s committed in C_s verifiable with $tp_i \in TP\}$

schemes for SoK_1 and SoK_2 can be realized using [40] and [39] and converting them into non-interactive signatures on m, thereby showing that the signature was not produced more than n times in the current time period (also to be verified against a database to ensure the serial number is not used multiplicatively). For SoK_3, the signer does as follows. She computes the commitment $C_s := g^s h^{\rho_1}$ and the commitments, $C_{\rho+r} := g^{\rho+r} h^{\rho_2}$, $C_{\tilde{\sigma}_{id}} := g^{\tilde{\sigma}_{id}} h^{\rho_3}$, $C_v := v g^{\rho_4}$, $C_{\rho_4} := g^{\rho_4} h^{\rho_5}$, $C_{\rho_4 \tilde{\sigma}_{id}} := g^{\rho_4 \tilde{\sigma}_{id}} h^{\rho_6}$, and $C := (C_v)^{\tilde{\sigma}_{id}} h^{\rho_7}$ using $\sigma_{id} = (\rho + r, \tilde{\sigma}_{id}, v)$ and picking $\rho_1, \ldots, \rho_7$ at random as in "Proof of Knowledge of a Signature" protocol of [34].[6] She then generates zero-knowledge proofs of all listed proofs in the mentioned proof scheme of [34] as SoKs on m. For $\mathrm{SoK}_{3+1}, \ldots, \mathrm{SoK}_{3+k}$, the signer also follows the same method as SoK_3 by replacing s with $s + H(\omega)$ and id with ω.

Note that our basic solutions has some improvements over what would have been obtained if one had just converted [16] to a non-interactive signature scheme. First, we provide a method for attribute authorities generate a signature on the same secret key of the user to eliminate collusion among the users. Second, we require computation of the serial number S and the commitments C_J and C_s only once per signature, since they are utilized in counting the number of signatures a person has generated (but not how many times a particular one of her attributes is utilized), and the commitments $C_{s+H(\omega)}$ can be obtained utilizing C_s. Third, due to the fact that we only require the more than n-times signing attempts to be detectable but not de-anonymized, we removed the double spending tags of [16] for efficiency.

For the Sign algorithm with OR policy, the subpolicies can be combined in SoK using [37]. For example, to sign with a policy $(\omega_1 \wedge \omega_2) \vee (\omega_3 \wedge \omega_4)$, the subpolicies $(\omega_1 \wedge \omega_2)$ and $(\omega_3 \wedge \omega_4)$ can be combined with an OR proof.

Verify. Algorithm 3 shows our Verify algorithm for an AND policy. The Verify algorithm with OR policies is obtained via check of OR proofs.

[6] Note that the commitments that we provide here are the commitments listed in that protocol of [34] in the same order, with different notation.

Algorithm 3 Our Verify algorithm for AND policy

input: a message m, an ABS μ, an AND policy $\beta = \omega_1 \wedge \ldots \wedge \omega_k$, the issuing public key iip and the revocation public key irp of the provider of the σ_{id}, the issuing public key set AIP and the revocation public key set ARP of the authorities that have issued Σ_β, the tracing public key set TP of the tracing authorities that have issued Σ_T, the current time period t, a signature database SDB_j, the allowed number n of the legitimate signatures by a user in a time period, and global setup parameters $params = (q, \mathbb{G}, g, h)$.
output: a bit b and implicitly an updated signature database SDB_{j+1}.

$(S, \mathsf{C}_J, \mathsf{C}_s, \Phi, \mathrm{SoK}_1, \ldots, \mathrm{SoK}_{3+k}, \mathrm{SoK}_{4+k}, \ldots, \mathrm{SoK}_{5+k+\theta}) := \mu$
$b := 1$; $SDB_{j+1} := SDB_j || \mu$
if S is a part of any signature in SDB **then**
  // check for any other use of S to ensure n times usage
  $b := 0$; $SDB_{j+1} := SDB_j$
else
  for $i = 1, \ldots, k$ **do** $\mathsf{C}_{s+H(\omega_i)} := \mathsf{C}_s g^{H(\omega_i)}$
  for $i = 1, \ldots, 3+k$ **do**
    // check for n times usage is completed in the first 3
    // iterations of this loop
    if SoK_i does not verify **then** $b := 0$; $SDB_{j+1} := SDB_j$
  for $i = 4+k, \ldots, 5+k+\theta$ **do**
    if SoK_i does not verify **then** $b := 0$; $SDB_{j+1} := SDB_j$

Algorithm 4 CL signature key generation algorithm for the tracing authority i

input: a unary security parameter 1^λ and global setup parameters $params = (q, \mathbb{G}, g, h)$.
output: a tracing authority CL public key $(\mathbf{g}_i'', \mathbf{h}_i'', \mathbf{j}_i'', \mathsf{O}_i)$, its CL secret key $(\mathsf{p}_i''', \mathsf{q}_i''')$, and the related proofs ($\mathrm{NIZKPoK}_{t,1}$, $\mathrm{NIZKPoK}_{t,2}$).

$\mathsf{p}_i''', \mathsf{q}_i''' \leftarrow O(1^\lambda)$ Sophie Germain primes; $\mathsf{O}_i := (2\mathsf{p}_i''' + 1)(2\mathsf{q}_i''' + 1)$; $\mathbf{g}_i'', \mathbf{h}_i'', \mathbf{j}_i'' \leftarrow \mathcal{QR}_{\mathsf{O}_i}$
$\mathrm{NIZKPoK}_{t,1}$:=$\mathrm{NIZKPoK}\{(\mathsf{q}_i'', \mathsf{p}_i'') : \mathsf{O}_i = (2\mathsf{p}_i''' + 1)(2\mathsf{q}_i''' + 1)\}$
$\mathrm{NIZKPoK}_{t,2}$:=$\mathrm{NIZKPoK}\{\mathbf{g}_i'', \mathbf{h}_i'', \mathbf{j}_i'' \in \mathcal{QR}_{\mathsf{O}_i}\}$

4.2 Adding Threshold Traceability

In this subsection, we explain how to add traceability to our basic solution. To protect against malicious tracing authorities, our solution employs a threshold scheme, where an adversary who controls fewer than threshold-many tracing authorities cannot attack the system. Only at least threshold-many (θ) tracing authorities can de-anonymize a user from a given signature. Moreover, no authority can blame a user by forging a signature on her behalf. Below, we only explain the changes needed on top of our basic scheme (round cornered rectangle parts in the pseudocodes).

Our technique utilizes the threshold public key encryption (TPKE) scheme of [44]. Note that we have chosen this TPKE scheme for efficiency, yet it may also be possible to build a similar construction via other non-broadcast TPKE schemes (e.g., [45, 46, 47, 48, 49, 50], as long as they allow efficient zero-knowledge proof of equality of encrypted value to a committed value and efficient validation of correct construction of a ciphertext by a third party. On the other hand, the broadcast TPKE schemes [51, 52] are not suitable in our scheme for traceability, due to the fact that they require the signer to provide the public-secret key pairs to the tracing authorities herself, which results in disclosure of her identity even by the verifier that needs to validate tracing transcripts.

Setup. In TraceSetup protocol, first a group $\mathbb{G}_T$ of prime order q with generators g_1 and g_2 is generated in a distributed fashion by all ϕ tracing authorities. Then, the authorities together generate the encryption public key $tep := (p, q, g_1, g_2, c, d, h_1)$ and their decryption secret key shares tes_i (such that θ of them can decrypt a ciphertext) as described in [44]. Each tracing authority i runs Algorithm 4 for CL signature key generation, sets its tracing public key as $tp_i := (tep, \mathbf{g}_i'', \mathbf{h}_i'', \mathbf{j}_i'', \mathsf{O}_i)$ and secret key as $ts_i = (tes_i, \mathsf{p}_i''', \mathsf{q}_i''')$, and publishes tp_i, $\mathrm{NIZKPoK}_{t,1}$, and $\mathrm{NIZKPoK}_{t,2}$, showing correct generation of the values. In this phase, each honest authority checks the transcripts of every other tracing authority, and terminates the procedure in case of detection of any malicious behaviour.

User Join. When a user joins the system, upon running the protocol in Figure 1 with an identity provider,

she interacts with at least θ tracing authorities, with each of which she runs the TraceIssue protocol given in Figure 3 to obtain her tracing tokens $\sigma_{T,i}$, i.e., a CL signature on s. The tracing authorities together register the tuple (uid, g_1^s) into the shared tracing database TDB of tracing authorities, which only permits update by the consensus of θ tracing authorities. Note that there are generic ways (e.g., Byzantine Fault Tolerance [53, 54]) of maintaining such a consistent and consensus-based database as long as the majority of the authorities are honest (i.e., in our case $2(\theta - 1) < \phi$).

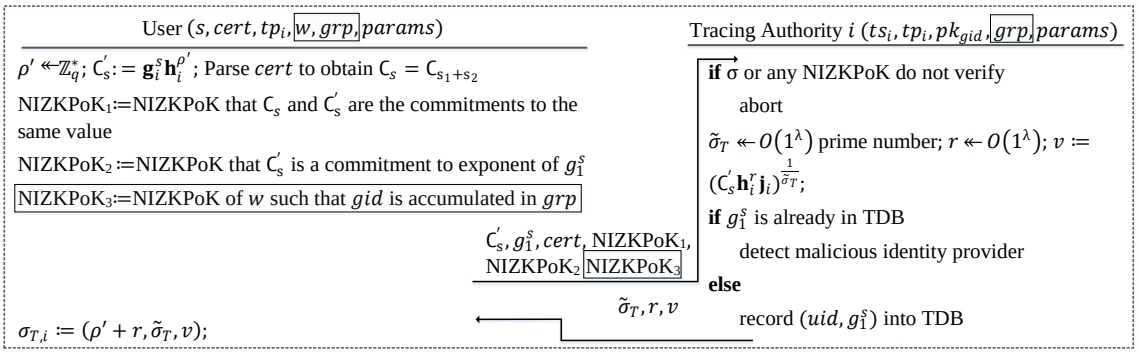

Figure (3) The TraceIssue protocol between a user and a tracing authority.

Sign. After getting her tracing tokens, a signer can use them in her signatures by including the parts inside the rounded cornered rectangles in the Sign algorithm. The tracing tag Φ is the ciphertext obtained by encrypting g_1^s with TPKE [44]. SoK$_{4+k}$ is generated by showing ρ_3 values are the same in all of $g_1^{\rho_3}$, $g_2^{\rho_3}$, $g_1^s h_1^{\rho_3}$ as a SoK on m and $c^{\rho_3} d^{\rho_3 \kappa}$. SoK$_{5+k}$ is composed of the conventional proof of equality of committed values on m. Overall, these proofs show that the user provided her own tracing tag as part of the VABS. SoK$_{6+k}, \ldots,$ SoK$_{5+k+\theta}$ are again generated by the same technique as described in Subsection 4.1 for the knowledge of CL signatures for identity and attribute tokens, showing that she obtained at least θ CL signatures from tracing authorities on her same identity. Note that instead of encrypting s, signer encrypts g_1^s within the tracing tag Φ, which has two advantages: secrecy of s and ease of SoK$_{5+k}$. As expected, in Verify algorithm, the verifier also checks SoK$_{4+k}, \ldots,$ SoK$_{5+k+\theta}$.

Trace. In the Trace protocol, given a valid VABS, tracing authorities run the TPKE decryption [44] for its tracing tag Φ as an authenticated Byzantine Fault Tolerance [53, 54] protocol, so that at the end the ones that follow the protocol would come to a consensus in the decryption[7] of Φ as g_1^s, and they find the associated uid in TDB via searching g_1^s. In our case, we assume at most $\theta - 1$ tracing authorities are malicious, resulting in at least θ honest tracing authorities always coming to the consensus on the correct signer uid. We note that "tracing soundness" of [7] is also ensured, since Φ can only be decrypted to a single value, assuming the θ tracing authorities joining the operation are honest.

4.3 Adding User/Attribute Revocation

To add user/attribute revocation to the system, the following steps should additionally be taken (sharp cornered rectangle parts in the pseudocodes).

Authority Join. In Algorithm 1, the proofs of knowledge for NIZKPoK$_3$ and NIZKPoK$_4$ can be instantiated as in [38] and [13], respectively, to show that revocation public key is set up correctly. Upon executing the algorithm, the identity provider/attribute authority i also publishes its initial revocation public $irp_{i,0}/arp_{i,0}$ together with the proofs NIZKPoK$_3$ and NIZKPoK$_4$, while keeping its secret key irs_i/ars_i.

User Revoke. For UserRevoke, inline with [55], an identity provider i updates its current revocation public key as

$$irp_{i,j+1} := irp_{i,j}^{\tilde{\sigma}_{id}^{-1} \mod 4\mathsf{p}_i''\mathsf{q}_i''} \mod \mathsf{M}_i$$

to revoke an issued σ_{id} using the related revocation handle $\tilde{\sigma}_{id}$. Note that according to [55], after each issuing or revocation, the identity provider may publish $\tilde{\sigma}_{id}$, so that the other users can update their witnesses. Instead, it is also possible to periodically issue or revoke users in a batch as

$$\psi := \left(\frac{\prod_{i \in revoked} \tilde{\sigma}_{id,i}}{\prod_{i \in issued} \tilde{\sigma}_{id,i}} \right) \mod 4\mathsf{p}_j''\mathsf{q}_j'',$$

so that for a user k with $gcd(\psi, \tilde{\sigma}_{id,k}) = 1$, she would only need ψ to efficiently update her witness by first computing a and b such that $a.\tilde{\sigma}_{id,k} + b.\psi = 1$ via the extended Euclidean algorithm, and then setting

[7] "Interactive proofs of validity of partial decryptions" method in [44] should be utilized for honest majority to obtain the correct plaintext.

$w_{id,j+1} := w^b_{id,j}.irp^a_{i+1}$. This is an optimization we propose over [55] to obtain a constant size product via modulo operation (see Appendix B for efficiency improvement analysis).

Sign & Verify. In the Sign algorithm, the SoK_3 related to the CL signature for identity is generated as follows. The signer computes the commitments C_s, $C_{\rho+r}$, $C_{g^{\tilde{id}}}$, C_v, C_{ρ_4}, $C_{\rho_4\tilde{\sigma}_{id}}$, and C, as described in Section 4.1. She then computes the commitments and values listed in "Efficient Proof That a Committed Value Was Accumulated" protocol in [55] to prove that the committed value in $C_{\tilde{\sigma}_{id}}$ is accumulated in the part $\mathbf{u}_{i,j+1}$ of $irp_{i,j}$. The signer then generates zero-knowledge proofs of all listed proofs in the mentioned proof schemes of [34] and [55] as SoKs on m. Overall, in SoK_3 she additionally demonstrates that her identity is not revoked by proving a witness, and in $\text{SoK}_{3+1}, \ldots, \text{SoK}_{3+k}$, she shows that the attributes that she is using to sign are not revoked by the related attribute authorities by proving associated witnesses. In the Verify algorithm, although the notation does not change, we also verify the witnesses mentioned above for $\text{SoK}_3, \ldots, \text{SoK}_{3+k}$.

4.4 Hiding Authorities

It is possible to enhance our Sign algorithm to achieve authority hiding in applications where revealing the authorities can disclose the identity of the signers.

To hide the identity provider, the signer includes in the input the issuing public key *set* IIP (instead of a single iip) and the revocation public key set IRP (instead of a single irp) of all identity providers. She then computes SoK_3 as

$\text{SoK}_3 := \text{SoK}[m]\{(s, \rho_2, \sigma_{id}, w_{id}) : \bigvee_{j=1}^{|IIP|}$ (σ_{id} is a CL signature on the committed value in C_s verifiable with $iip_j \in IIP$, and w_{id} is a witness that $\tilde{gid}$ is accumulated in $irp_j \in IRP$)$\}$

To hide the authority of an attribute token σ_{ω_i}, the signer includes in the input the issuing public key *set* AIP_{ω_i} (instead of the aip of the authority that issued σ_{ω_i}) and the revocation public key set ARP_{ω_i} (instead of the arp of the authority that issued σ_{ω_i}) of all authorities that can issue ω_i. She then computes SoK_{3+i} as

$\text{SoK}_{3+i} := \text{SoK}[m]\{(s, \rho_2, \sigma_{\omega_i}, w_i) : \bigvee_{j=1}^{|AIP_{\omega_i}|}$ (σ_{ω_i} is a CL signature on the committed value in $\mathsf{C}_{s+H(\omega_i)} = \mathsf{C}_s g^{H(\omega_i)}$ verifiable with $aip_j \in AIP_{\omega_i}$, and w_i is a witness that $\tilde{\sigma}_{\omega_i}$ is accumulated in $arp_j \in ARP_{\omega_i}$)$\}$

To hide the tracing authorities that she obtained the tracing tokens from, the signer includes in the input the public key set TP_ϕ of all tracing authorities (instead of TP). She then replaces $\text{SoK}_{6+k} \ldots \text{SoK}_{5+k+\theta}$ with a single SoK_{6+k} computed as

$\text{SoK}_{6+k} := \text{SoK}[m]\{(s, \rho_2, \Sigma_T) : \Sigma_T$ is a set of σ_{CL} signatures on s committed in C_s verifiable with θ elements of $TP_\phi\}$

All of these SoKs can be computed by the generic method provided in [37]. More concretely, to hide an authority, the signer first generates all the commitments for the tokens that she has as before, and then simulates the other commitments for the tokens that she does not have. Then, she generates all the SoKs on m as non-interactive OR proofs for her identity/attribute token or knowledge of θ out of ϕ proofs for her tracing tokens. In our modular design, it is possible, for example, to hide only the attribute authorities.

5 Security of Our VABS Scheme

Theorem 1. *If the Strong RSA assumption [16, 56] holds in the group $\mathbb{Z}^*_{N_i}$ and $\mathbb{Z}^*_{M_i}$ of each identity provider/attribute authority i and in the group $\mathbb{Z}^*_{O_i}$ of each tracing authority i, the underlying SoK schemes are secure (i.e., they satisfy completeness, soundness, and zero-knowledge properties of zero-knowledge proofs), $F_{g,s}(x) = g^{1/s+x}$ is a pseudorandom function (PRF) with input $x \in \mathbb{Z}^*_q$, the TPKE scheme of [44] is secure against chosen ciphertext attack (CCA-secure), $H(\cdot)$ is modeled as random oracle, the assumptions on the authorities hold, i.e., the identity providers/attribute authorities issue the tokens only to deserving users and out of $\phi \geq 2\theta - 1$ tracing authorities at most $\theta - 1$ are malicious; then our VABS scheme is secure (i.e., it satisfies anonymity, signature unforgeability, soundness, and reliable traceability).*

Remark 1. *The security of the recommended SoK schemes and the pseudorandomness of the function $F_{g,s}(x) = g^{1/s+x}$ are proven to be reducible to the Strong RSA assumption and the SDDHI assumption [16] holding in $\langle g \rangle$. Further, the CCA-security of the TPKE scheme of [44] is reducible to the DDH assumption holding in $\langle g_1 \rangle$.*

We prove that our VABS scheme does not allow collusion of users to combine their attribute tokens, and show that it satisfies anonymity, signature unforgeability, soundness, and reliable traceability. Due to space limitation, we keep here only the anonymity proof as an example, and put other full proofs in Appendix A.

Lemma 1. *If the underlying SoK schemes are zero-knowledge, and $F_{g,s}(x) = g^{1/s+x}$ is a pseudorandom function (PRF) for $x \in \mathbb{Z}_q^*$, and the TPKE scheme of [44] is secure against chosen ciphertext attack (CCA-secure); then our VABS scheme achieves anonymity.*

Proof. Assuming that the underlying SoK schemes are zero-knowledge and the threshold encryption of [44] is CCA-secure, we now reduce the anonymity of our VABS scheme to the pseudorandomness of $F_{g,s}(x) = g^{1/s+x}$. If a PPT adversary $\hat{\mathcal{A}}$ wins the anonymity game $\mathsf{VABSAnonym}$ with non-negligible advantage, then we can utilize $\hat{\mathcal{A}}$ to construct a PPT algorithm $\hat{\mathcal{B}}$ that distinguishes $F_{g,s}$ from a random function $f : \mathbb{Z}_q^* \to \langle g \rangle$ with non-negligible advantage. In the $\mathsf{VABSAnonym}$ game, $\hat{\mathcal{A}}$ and $\hat{\mathcal{B}}$ play the roles of $\mathcal{A}$ and $\mathcal{C}$, respectively. In the $\mathsf{DistPRF}$ game, $\hat{\mathcal{B}}$ plays the role of $\mathcal{A}$ against an honest challenger $\hat{\mathcal{C}}$. In $\mathsf{VABSAnonym}$, since we assume the security of the SoKs and the CCA-security of the TPKE scheme, we give the control of the simulators for non-interactive proofs and the TPKE scheme to $\hat{\mathcal{B}}$, e.g., $\hat{\mathcal{B}}$ simulates SoKs for $\hat{\mathcal{A}}$. Note that in the TPKE scheme of [44], $\hat{\mathcal{B}}$ can create a ciphertext by picking a random group element for each of its four parts, and during the decryption, if $\hat{\mathcal{B}}$ knows a plaintext m, using at least one authority under its control, it can simulate the partial decryption of that authority so that overall the decryption outputs m. The reduction follows:

1. In $\mathsf{DistPRF}$, $\hat{\mathcal{B}}$ obtains 1^λ, g, and q. $\hat{\mathcal{C}}$ also gives to $\hat{\mathcal{B}}$ the oracle access to $F_s(\cdot)$ or $f(\cdot)$ (chosen fairly at random).
2. In $\mathsf{VABSAnonym}$, $\hat{\mathcal{B}}$ picks arbitrary $h \in \langle g \rangle$ and $n, \delta \in poly(\lambda)$, and gives to $\hat{\mathcal{A}}$ the values 1^λ, n, δ, and $params = (q, \langle g \rangle, g, h)$. $\hat{\mathcal{B}}$ and $\hat{\mathcal{A}}$ then follow $\mathsf{TraceSetup}$ and publish the outputs as described. The time counter t is initialized as 1, and is started (**Step 1** of $\mathsf{VABSAnonym}$).
3. In $\mathsf{VABSAnonym}$, $\hat{\mathcal{B}}$ follows **Step 2** of the game with $\hat{\mathcal{A}}$ in the exact same way as an honest $\mathcal{C}$ would do.
4. $\hat{\mathcal{B}}$ obtains an attribute policy β chosen by $\hat{\mathcal{A}}$ (**Step 3** of $\mathsf{VABSAnonym}$).
5. In $\mathsf{VABSAnonym}$, $\hat{\mathcal{B}}$ generates two user secret keys s_0 and s_1, and computes commitments to these values using the authority public keys. $\hat{\mathcal{B}}$ gives to $\hat{\mathcal{A}}$ all the commitments to s_0 and s_1. $\hat{\mathcal{A}}$ generates attribute tokens on s_0 and s_1 to prove that both users conform to β and gives those signatures to $\hat{\mathcal{B}}$ (**Step 4** of $\mathsf{VABSAnonym}$).
6. In $\mathsf{VABSAnonym}$, a bit b is randomly picked by $\hat{\mathcal{B}}$ (normally an honest challenger would pick this bit in Step 7 of the game).
7. $\hat{\mathcal{A}}$ is given access to the Sign oracles for s_0 and s_1. However, the oracle outputs are arranged by $\hat{\mathcal{B}}$ as follows. If the Sign oracle for s_b is queried with $(m', \tilde{\beta}')$, $\hat{\mathcal{B}}$ queries the oracle in $\mathsf{DistPRF}$ with $t'2^{\ell_{cnt}} + J_b$, obtains the oracle output S', and then generates $\sigma \leftarrow (S', \mathsf{C}_{J_b}, \varsigma', \Phi,$ the simulated SoKs on $m')$, where $\varsigma' \twoheadleftarrow \langle g \rangle$ is the commitment simulation, and all the SoKs and the ciphertext Φ (so that it always traces to the signer) are simulated. If $\mathsf{Sign}(\cdot, s_{\bar{b}}, \tilde{\beta}, \tilde{\Sigma}_\Omega, \tilde{AP}, t, J_{\bar{b}})$ is queried with $(m', \tilde{\beta}')$, $\hat{\mathcal{B}}$ itself picks $S' \twoheadleftarrow \langle g \rangle$ instead of querying the oracle in $\mathsf{DistPRF}$, and generates the other parts of the signature in the same way as in s_b. $\hat{\mathcal{B}}$ gives the oracle outputs to $\hat{\mathcal{A}}$ (**Step 5** of $\mathsf{VABSAnonym}$).
8. In $\mathsf{VABSAnonym}$, $\hat{\mathcal{A}}$ generates two messages m_0 and m_1 (**Step 6** of $\mathsf{VABSAnonym}$).
9. In $\mathsf{DistPRF}$, $\hat{\mathcal{B}}$ queries the oracle with $t2^{\ell_{cnt}} + J_b$, and obtains the oracle output S_b. In $\mathsf{VABSAnonym}$, $\hat{\mathcal{B}}$ sets $\sigma_0 := (S_b, \varsigma_1, \mathsf{C}_{J_b},$ the simulated SoKs on $m_0, A)$ and $\sigma_1 := (S_{\bar{b}}, \varsigma_2, \mathsf{C}_{J_{\bar{b}}},$ the simulated SoKs on $m_1, A)$ where $S_{\bar{b}} \twoheadleftarrow \langle g \rangle$, $\varsigma_1, \varsigma_2 \twoheadleftarrow \langle g \rangle$ are commitment simulations, and all the SoKs are simulated (**Step 7** of $\mathsf{VABSAnonym}$).
10. In $\mathsf{VABSAnonym}$, $\hat{\mathcal{A}}$ eventually outputs a bit b' (**Step 8** of $\mathsf{VABSAnonym}$). If $b = b'$, in $\mathsf{DistPRF}$, $\hat{\mathcal{B}}$ outputs its guess for the oracle as $F_{g,s}$. Otherwise, $\hat{\mathcal{B}}$ outputs its guess as f.

	1 attribute	3 attributes	5 attributes	10 attributes	15 attributes	20 attributes
Exec. time of Sign	119.5 ms	225.5 ms	331.5 ms	596.5 ms	861.5 ms	1126.5 ms
Exec. time of Verify	98.4 ms	186.4 ms	274.4 ms	494.4 ms	714.4 ms	934.4 ms
VABS size	51.2 kB	92.2 kB	133.2 kB	235.7 kB	338.2 kB	440.7 kB

Table (2) The implementation results of Sign and Verify algorithms of our basic ABS scheme for various attribute policy sizes. The RSA and prime-order group modulus lengths are set as 1024 bit, and SHA256 is used as the random oracle.

In both σ_0 and σ_1, the only values where $\hat{\mathcal{A}}$ can make the differentiation are S_b and $S_{\bar{b}}$, since the Pedersen commitment scheme is information theoretically hiding, and SoKs and Φ are simulated. Moreover, if the oracle in DistPRF is the random function f, then S_b and $S_{\bar{b}}$ are also perfectly indistinguishable, which implies that $\hat{\mathcal{A}}$ would not obtain non-negligible advantage from these values. If $\hat{\mathcal{A}}$ obtains non-negligible advantage from these values, then the oracle is the pseudorandom function $F_{g,s}$. Let ϵ_1 and ϵ_2 denote the advantages of the adversaries in VABSAnonym and DistPRF, respectively. Then, we have

$$\mathsf{Pr}[\hat{\mathcal{A}} \text{ wins VABSAnonym (i.e., } b = b')] = \frac{1}{2} + \epsilon_1$$

$$\mathsf{Pr}[\hat{\mathcal{B}} \text{ outputs } F_{g,s} | \text{ the oracle is } F_{g,s}]\mathsf{Pr}[\hat{\mathcal{B}} \text{ outputs } \mathsf{F_{g,s}} | \text{ the oracle is } f] = \epsilon_2$$

Via the total probability law, we can write the first equality as

$$\mathsf{Pr}[b = b' | \text{ the oracle is } F_{g,s}].\mathsf{Pr}[\text{the oracle is } F_{g,s}] + \mathsf{Pr}[b = b' | \text{ the oracle is } f].\mathsf{Pr}[\text{the oracle is } f] = \frac{1}{2} + \epsilon_1$$

Also, since if $b = b'$, $\hat{\mathcal{B}}$ outputs $F_{g,s}$, and otherwise, it outputs f, we can convert the second equality as

$$\mathsf{Pr}[b' = b | \text{ the oracle is } F_{g,s}] - \mathsf{Pr}[b' = b | \text{ the oracle is } f] = \epsilon_2$$

Combining the above two resulting equation and substituting $1/2$ for both $\mathsf{Pr}[\text{the oracle is } F_{g,s}]$ and $\mathsf{Pr}[b = b' | \text{ the oracle is } f]$, we obtain

$$\frac{1}{2} \cdot (\mathsf{Pr}[b' = b | \text{ the oracle is } f] + \epsilon_2) + \frac{1}{2} \cdot \mathsf{Pr}[b' = b | \text{ the oracle is } f] = \frac{1}{2} + \epsilon_1$$

$$\mathsf{Pr}[b' = b | \text{ the oracle is } f] + \frac{1}{2}\epsilon_2 = \frac{1}{2} + \epsilon_1$$

If the oracle is f, $\hat{\mathcal{A}}$ can win VABSAnonym with exactly $1/2$ probability, since its view for both $b = 0$ and $b = 1$ is statistically identical. Therefore, $\frac{1}{2}\epsilon_2 = \epsilon_1$ and if $\hat{\mathcal{A}}$ wins VABSAnonym with non-negligible advantage ϵ_1, then $\hat{\mathcal{B}}$'s advantage ϵ_2 in DistPRF is also non-negligible. Hence, since ϵ_2 is negligible if the PRF is indistinguishable from random, so must ϵ_1 be, showing that our VABS provides anonymity. □

6 Efficiency

Observe that GlobalSetup and TraceSetup protocols take place only once. Further, IdPJoin, AuthJoin, UserJoin, TraceIssue, AttrIssue, AttrRevoke, and Trace operations also take place infrequently, and require a constant number of public key operations. The most frequently used operations would be Sign and Verify, and hence in this part, we concentrate on analyzing their efficiency.

Asymptotically, Sign algorithm's computational cost and the signature size is proportional to the number of attributes, and hence is $O(|IIP| + |\beta|.|AIP| + \phi)$ SoK costs. Similarly, Verify algorithm requires the verifier to execute $O(|IIP| + |\beta|.|AIP| + \phi)$ SoK verifications and a search for the serial number of the signature in the database for the last time period. The main overhead $O(|\beta|.|AIP|)$ suggests that a limitation on the number of authorities that are issuing each attribute in practice may help with the efficiency of the system.

With a prototype code, we analyzed the cost of the Sign and Verify algorithms of our basic construction in Section 4.1 using the given primitives implemented in the Cashlib cryptographic library[8]

[8] `https://github.com/brownie/cashlib`

[57]. Our test environment is a virtual machine (under VirtualBox) using Intel(R) Core(TM) i7-7600U (2.80GHz) CPU with 6.4 GB RAM running Lubuntu 16.04.5. The RSA and prime-order group modulus lengths are set as 1024 bit, and SHA256 is used as the random oracle. We repeated our tests 10 times. Table 2 shows the average results of our implementation for various attribute policy sizes. More precisely, on average, our Sign algorithm takes 66.5 ms plus 53.0 ms per attribute that needs to be proven. Also, on average, the length of the generated ABS is 30.7 kB plus 20.5 kB per attribute. Further, on average, our Verify algorithm requires 54.4 ms plus 44.0 ms per attribute. In the future, we plan to extend our implementation to cover the modular extensions of our VABS, and obtain efficiency numbers for higher security parameters as well.

References

[1] H. K. Maji, M. Prabhakaran, and M. Rosulek, "Attribute-based signatures," in *CT-RSA '11*, 2011.

[2] D. Cao, B. Zhao, X. Wang, and J. Su, "Flexible multi-authority attribute-based signature schemes for expressive policy," *Mob. Inf. Syst.*, vol. 8, July 2012.

[3] T. Okamoto and K. Takashima, "Decentralized attribute-based signatures," in *PKC '13*, 2013.

[4] A. El Kaafarani, E. Ghadafi, and D. Khader, "Decentralized traceable attribute-based signatures," in *CT-RSA '14*, 2014.

[5] A. El Kaafarani, L. Chen, E. Ghadafi, and J. Davenport, "Attribute-based signatures with user-controlled linkability," in *CNS '14*, 2014.

[6] B. Hampiholi, G. Alpár, F. v. d. Broek, and B. Jacobs, "Towards practical attribute-based signatures," in *Security, Privacy, and Applied Cryptography Engineering*, 2015.

[7] E. Ghadafi, "Stronger security notions for decentralized traceable attribute-based signatures and more efficient constructions," in *CT-RSA '15*, Springer International Publishing, 2015.

[8] M. Urquidi, D. Khader, J. Lancrenon, and L. Chen, "Attribute-based signatures with controllable linkability," in *Trusted Systems*, 2016.

[9] R. Bobba, O. Fatemieh, F. Khan, C. A. Gunter, and H. Khurana, "Using attribute-based access control to enable attribute-based messaging," in *ACSAC '06*, 2006.

[10] R. Bobba, O. Fatemieh, F. Khan, A. Khan, C. A. Gunter, H. Khurana, and M. Prabhakaran, "Attribute-based messaging: Access control and confidentiality," *ACM TISSEC.*, vol. 13, Dec. 2010.

[11] K. B. Frikken, J. Li, and M. J. Atallah, "Trust negotiation with hidden credentials, hidden policies, and policy cycles," in *NDSS '06*, 2006.

[12] X. Boyen, "Mesh signatures," in *EUROCRYPT '07*, 2007.

[13] J. Camenisch and A. Lysyanskaya, "An efficient system for non-transferable anonymous credentials with optional anonymity revocation," in *EUROCRYPT '01*, 2001.

[14] Y. Sakai, J. C. N. Schuldt, K. Emura, G. Hanaoka, and K. Ohta, "On the security of dynamic group signatures: Preventing signature hijacking," in *PKC '12*, 2012.

[15] A. El Kaafarani and E. Ghadafi, "Attribute-based signatures with user-controlled linkability without random oracles," in *Cryptography and Coding*, 2017.

[16] J. Camenisch, S. Hohenberger, M. Kohlweiss, A. Lysyanskaya, and M. Meyerovich, "How to win the clonewars: Efficient periodic n-times anonymous authentication," in *ACM CCS '06*, 2006.

[17] J. Camenisch and T. Groß, "Efficient attributes for anonymous credentials," *ACM Trans. Inf. Syst. Secur.*, vol. 15, Mar. 2012.

[18] C. Garman, M. Green, and I. Miers, "Decentralized anonymous credentials," in *NDSS '14*, 01 2014.

[19] W. Lueks, G. Alpár, J.-H. Hoepman, and P. Vullers, "Fast revocation of attribute-based credentials for both users and verifiers," in *ICT Systems Security and Privacy Protection*, 2015.

[20] D. Derler, C. Hanser, and D. Slamanig, "A new approach to efficient revocable attribute-based anonymous credentials," in *IMACC '15*, 2015.

[21] J. Camenisch, M. Drijvers, and J. Hajny, "Scalable revocation scheme for anonymous credentials based on n-times unlinkable proofs," in *ACM WPES '16*, 2016.

[22] P. P. Tsang, M. H. Au, A. Kapadia, and S. W. Smith, "Blacklistable anonymous credentials: Blocking misbehaving users without ttps," in *ACM CCS '07*, 2007.

[23] L. Nguyen, "Accumulators from bilinear pairings and applications," in *CT-RSA '05*, 2005.

[24] D. Deuber, M. Maffei, G. Malavolta, M. Rabkin, D. Schröder, and M. Simkin, "Functional credentials," *PoPETs*, no. 2, 2018.

[25] J. Camenisch and J. Groth, "Group signatures: Better efficiency and new theoretical aspects," in *Security in Communication Networks*, 2005.

[26] M. Bellare, H. Shi, and C. Zhang, "Foundations of group signatures: The case of dynamic groups," in *CT-RSA '05*, 2005.

[27] G. Yang, S. Tang, and L. Yang, "A novel group signature scheme based on mpkc," in *ISPEC '11*, 2011.

[28] D. Slamanig, R. Spreitzer, and T. Unterluggauer, "Adding controllable linkability to pairing-based group signatures for free," in *Information Security*, 2014.

[29] J. Hwang, L. Chen, H. Cho, and D. Nyang, "Short dynamic group signature scheme supporting controllable linkability," in *IEEE TIFS*, vol. 10, 06 2015.

[30] E. Ghadafi, "Efficient distributed tag-based encryption and its application to group signatures with efficient distributed traceability," in *LATINCRYPT '14*, 2015.

[31] M. Nisansala, S. Perera, and T. Koshiba, "Fully secure lattice-based group signatures with verifier-local revocation," in *IEEE, AINA '17*, 5 2017.

[32] K. Emura, T. Hayashi, and A. Ishida, "Group signatures with time-bound keys revisited: A new model and an efficient construction," in *ASIA CCS '17*, 2017.

[33] J. Bootle, A. Cerulli, P. Chaidos, E. Ghadafi, and J. Groth, "Foundations of fully dynamic group signatures," in *ACNS '16*, 2016.

[34] J. Camenisch and A. Lysyanskaya, "A signature scheme with efficient protocols," in *Security in Communication Networks*, Springer Berlin Heidelberg, 2003.

[35] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *CRYPTO '86*, 1987.

[36] J. Camenisch and M. Stadler, "Efficient group signature schemes for large groups," in *CRYPTO '97*, 1997.

[37] R. Cramer, I. Damgård, and B. Schoenmakers, "Proofs of partial knowledge and simplified design of witness hiding protocols," in *CRYPTO '94*, 1994.

[38] J. Camenisch and M. Michels, "Proving in zero-knowledge that a number is the product of two safe primes," in *EUROCRYPT '99*, 1999.

[39] F. Boudot, "Efficient proofs that a committed number lies in an interval," in *EUROCRYPT '00*, 2000.

[40] C. P. Schnorr, "Efficient signature generation by smart cards," *Journal of Cryptology*, vol. 4, Jan 1991.

[41] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *CRYPTO '91*, 1992.

[42] T. P. Pedersen, "A threshold cryptosystem without a trusted party," in *EUROCRYPT '91*, 1991.

[43] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, "Secure distributed key generation for discrete-log based cryptosystems," *J. Cryptol.*, vol. 20, Jan. 2007.

[44] R. Canetti and S. Goldwasser, "An efficient threshold public key cryptosystem secure against adaptive chosen ciphertext attack," in *EUROCRYPT'99*, 1999.

[45] D. Boneh, X. Boyen, and S. Halevi, "Chosen ciphertext secure public key threshold encryption without random oracles," in *CT-RSA '06*, 2006.

[46] S. Arita and K. Tsurudome, "Construction of threshold public-key encryptions through tag-based encryptions," in *ACNS '09*, 2009.

[47] B. Libert and M. Yung, "Adaptively secure non-interactive threshold cryptosystems," in *Automata, Languages and Programming*, 2011.

[48] H. Wee, "Threshold and revocation cryptosystems via extractable hash proofs," in *EUROCRYPT '11*, 2011.

[49] X. J. Lin and L. Sun, "Efficient cca-secure threshold public-key encryption scheme." Cryptology ePrint Archive, Report 2013/749, 2013. `http://eprint.iacr.org/2013/749`.

[50] Y. Gan, L. Wang, L. Wang, P. Pan, and Y. Yang, "Efficient construction of cca-secure threshold pke based on hashed diffie-hellman assumption," *Computer Journal*, vol. 56, pp. 1249–1257, 2013.

[51] C. Delerablée and D. Pointcheval, "Dynamic threshold public-key encryption," in *CRYPTO '08*, 2008.

[52] Y. Sakai, K. Emura, J. Schuldt, G. Hanaoka, and K. Ohta, "Dynamic threshold public-key encryption with decryption consistency from static assumptions," in *ISP '15*, 2015.

[53] G. S. Veronese, M. Correia, A. N. Bessani, L. C. Lung, and P. Verissimo, "Efficient byzantine fault-tolerance," *IEEE Trans. Comput.*, vol. 62, Jan. 2013.

[54] Y. Lindell, *2. The Composition of Authenticated Byzantine Agreement.* 2003.

[55] J. Camenisch and A. Lysyanskaya, "Dynamic accumulators and application to efficient revocation of anonymous credentials," in *CRYPTO '02*, 2002.

[56] N. Barić and B. Pfitzmann, "Collision-free accumulators and fail-stop signature schemes without trees," in *EUROCRYPT '97*, 1997.

[57] S. Meiklejohn, C. C. Erway, A. Küpçü, T. Hinkle, and A. Lysyanskaya, "Zkpdl: A language-based system for efficient zero-knowledge proofs and electronic cash," USENIX Security '10, 2010.

[58] J. Katz and Y. Lindell, *Introduction to Modern Cryptography.* Chapman & Hall/CRC, 2nd ed., 2007.

A Completion of the Proof of Theorem 1

Lemma 2. *If the identity providers are honestly randomize the secret keys, and $H(\cdot)$ is modeled as random oracle; then the probability that different users can combine their attribute tokens is negligible.*

Proof. In our scheme, users i and j with secret keys s_i and s_j can combine their attribute tokens for the attributes ω_k and ω_l, only in three different cases: (1) $s_i = s_j$, (2) the equality $s_i = s_j + H(\omega_l)$ holds for some $s_i \neq s_j$ and the user attribute authority is permitted to issue attribute ω_l, and (3) the equality $s_i + H(\omega_k) = s_j + H(\omega_l)$ holds for some $s_i \neq s_j$ and some $\omega_k \neq \omega_l$. Our scheme does not allow any collusion in any other case, since CL signatures are only generated on these values and do

not permit combining. Therefore, it will be sufficient to show that the probability that there exists two different users with secret keys s_i and s_j, and two different attributes ω_k and ω_l, such that at least one of the cases (1), (2), and (3) holds is negligible. Let Γ, Ψ, η and φ denote the set of the secret keys s in the system, the set of attributes ω in the system, the number of elements of Γ, and the number of elements of Ψ, respectively. Note that the number of users and attributes can be considered as bounded by a polynomial $poly(\lambda)$. If the identity providers are honest, and randomize the user secret keys, each user key s_i, s_j, and their difference $s_i - s_j$ is statistically identical to picking randomly from $\mathbb{Z}_q$. Also, as $H(\cdot)$ is modeled as a random oracle, each randomized attribute $H(\omega_k)$, $H(\omega_l)$, and their difference $H(\omega_k) - H(\omega_l)$ is statistically identical to picking randomly from $\mathbb{Z}_q$. Considering the fact that the number of elements in $\mathbb{Z}_q$ is q, for cases (1), (2), and (3), we calculate

$$\mathsf{Pr}[\exists s_i, s_j \in \Gamma \text{ s.t. } s_i = s_j, i \neq j] = 1 - \prod_{x=2}^{\eta} \left(1 - \frac{x-1}{q}\right) = O(\eta^2/q),$$

$$\mathsf{Pr}[\exists s_i, s_j \in \Gamma, \exists H(\omega) \in \Psi \text{ s.t. } s_i = s_j + H(\omega), i \neq j] \leq 1 - \left(1 - \frac{\varphi}{q}\right)^{2\cdot\binom{\eta}{2}} = O(\eta^2\varphi/q),$$

$$\mathsf{Pr}[\exists s_i, s_j \in \Gamma, \exists H(\omega_k), H(\omega_l) \in \Psi \text{ s.t. } s_i + H(\omega_k) = s_j + H(\omega_l),$$
$$i \neq j, \omega_k \neq \omega_l] \leq 1 - \left(1 - \frac{\binom{\varphi}{2}}{q-1}\right)^{2\cdot\binom{\eta}{2}} = O(\eta^2\varphi^2/q).$$

From the above, we calculate the probability ϵ of at least one of the above three events occuring as at most the addition of them, i.e., $\epsilon = O(\eta^2/q) + O(\eta^2\varphi/q) + O(\eta^2\varphi^2/q) = O(\eta^2\varphi^2/q)$. Since $q \in \Theta(2^\lambda)$ and $\eta, \varphi \in poly(\lambda)$, we deduce that ϵ is a negligible function of λ. □

Lemma 3. *If the Strong RSA assumption [16, 56] holds in the groups $\mathbb{Z}^*_{N_i}$ and $\mathbb{Z}^*_{M_i}$ of each identity provider/attribute authority i, $\log_g h$ is not deducible by any party (due to the DDH assumption in $\langle g \rangle$), the underlying SoK schemes are sound, and the underlying hash function $H : \{0,1\}^* \to \mathbb{Z}_q$ is modeled as a random oracle; then our VABS scheme achieves signature unforgeability.*

Proof. Assuming that $\log_g h$ is not deducible by any party, and that all the underlying SoK schemes satisfy soundness, and that H is modeled as a random oracle, we now reduce signature unforgeability of our VABS scheme to the Strong RSA assumption that should hold in all $\mathbb{Z}^*_{N_i}$ and $\mathbb{Z}^*_{M_i}$ of identity providers and attribute authorities. Note that we also assumue that non-revocation witnesses are unforgeable, since otherwise, it is also shown in [55] that one can break Strong RSA assumption. If a PPT adversary $\hat{\mathcal{A}}$ wins the signature unforgeability game $\mathsf{VABSForge}$ with non-negligible advantage, then we can utilize $\hat{\mathcal{A}}$ to construct a PPT algorithm $\hat{\mathcal{B}}$ that breaks the unforgeability of CL signature of at least one authority non-negligible advantage. Since forging CL signatures imply breaking the Strong RSA assumption as shown in [34], $\hat{\mathcal{B}}$ can trivally be utilized for constructing a PPT algorithm that breaks the Strong RSA assumption in at least one attribute authority group. In the game $\mathsf{VABSForge}$, $\hat{\mathcal{A}}$ and $\hat{\mathcal{B}}$ play the roles of $\mathcal{A}$ and $\mathcal{C}$, respectively. In the $\mathsf{CLSignForge}$ game (the CL signature equivalent of the conventional existential unforgeability under adaptive chosen message attack game for signatures as described as Sig-forge in [58]), $\hat{\mathcal{B}}$ plays the role of $\mathcal{A}$ against an honest challenger $\hat{\mathcal{C}}$. Briefly, in $\mathsf{CLSignForge}$, given a security parameter λ, a CL public key p and access to the related CL signing oracle, the adversary wins by computing a message m and a signature σ_{CL} on it that gets verified by the public key p, subject to the restriction that m cannot be previously queried to the oracle.

We start by replacing H with a random oracle under control of $\hat{\mathcal{B}}$ and accessible by $\hat{\mathcal{A}}$. Also, $\hat{\mathcal{B}}$ extracts SoKs of $\hat{\mathcal{A}}$ such that if $\hat{\mathcal{A}}$ needs to generate $\mathrm{SoK}[m]\{\xi : \xi \text{ satisfies } \zeta\}$, it needs to give the input a message m, the information ξ, and condition ζ to $\hat{\mathcal{B}}$, who only checks whether ξ satisfies ζ.

1. In $\mathsf{CLSignForge}$, $\hat{\mathcal{B}}$ obtains 1^λ.
2. In $\mathsf{VABSForge}$, $\hat{\mathcal{B}}$ runs $\mathsf{GlobalSetup}$ on 1^λ and obtains $params = (q, \mathbb{G}, g, h)$. Also, it picks arbitrary $n, \delta \in poly(\lambda)$, and gives 1^λ, n, δ, and $params$ to $\hat{\mathcal{A}}$. $\hat{\mathcal{B}}$ and $\hat{\mathcal{A}}$ then follow $\mathsf{TraceSetup}$ and publish the outputs as described. The time counter t is initialized as 1, and is started (**Step 1** of $\mathsf{VABSForge}$).

3. In VABSForge, $\hat{\mathcal{B}}$ follows **Step 2** of the game with $\hat{\mathcal{A}}$ in the exact same way as an honest $\mathcal{C}$ would do, except for one of the attribute authorities (picked randomly by $\hat{\mathcal{B}}$) that can issue ω under $\hat{\mathcal{B}}$'s control generated after $\hat{\mathcal{A}}$'s request. Note that according to the game definition $\hat{\mathcal{A}}$ must request at least one attribute authority (see Step 5 of VABSForge) for verification of ω. For that authority only[9], $\hat{\mathcal{B}}$ acts as the Steps 6 and 8 of this proof below.

4. $\hat{\mathcal{A}}$ is allowed to corrupt all ϕ tracing authorities (**Step 3** of VABSForge.

5. In VABSForge, $\hat{\mathcal{A}}$ returns an attribute ω (that is not queried to the authorities under $\hat{\mathcal{B}}$'s control for users under $\hat{\mathcal{A}}$'s control (**Step 4** of VABSForge).

6. In CLSignForge, $\hat{\mathcal{B}}$ obtains the public key p. $\hat{\mathcal{B}}$ gives to $\hat{\mathcal{A}}$ as one of the public keys of authorities under $\hat{\mathcal{B}}$'s control (**Step 5** of VABSForge).

7. In VABSForge, for each s_i queried for the attribute ω by $\hat{\mathcal{A}}$ that verifies in the checks of the previous step, $\hat{\mathcal{B}}$ itself picks a random value ϱ, sets it as the query output to the random oracle for ω, and sets $m_i = s_i + \varrho$.

8. In VABSForge, if $\hat{\mathcal{A}}$ queries the authority oracle for p, then $\hat{\mathcal{B}}$ queries m_i to the signing oracle in CLSignForge, receives the output and returns it to $\hat{\mathcal{A}}$. Otherwise, $\hat{\mathcal{B}}$ signs the message with the secret key of the queried authority itself.

9. In VABSForge, $\hat{\mathcal{A}}$ eventually outputs $(m, \beta \wedge \omega, \sigma)$ to $\hat{\mathcal{B}}$ (**Step 6** of VABSForge). If $\hat{\mathcal{A}}$ wins, $\hat{\mathcal{B}}$ extracts the SoK_{3+i} query for the winning μ, where i is the index of ω. $\hat{\mathcal{B}}$ then parses the related SoK query from $\hat{\mathcal{A}}$ and using the SoK extractor $\hat{\mathcal{B}}$ obtains the CL signature σ_{CL} that $\hat{\mathcal{A}}$ utilized.

10. In CLSignForge, $\hat{\mathcal{B}}$ outputs σ_{CL}.

As shown in [16], $\hat{\mathcal{A}}$ cannot achieve a forgery in commitments with non-negligible probability, since it cannot deduce $\log_g h$. Also, since SoKs are sound, $\hat{\mathcal{A}}$ cannot prove with non-negligible probability without knowing the CL signatures. Since the non-revocation witnesses are also assumed to be unforgeable, the only way that $\hat{\mathcal{A}}$ wins VABSForge is via forging a CL signature with non-negligible probability. Since the generated signature would correspond to the authority p with non-negligible probability (i.e., at least $1/K$ where $K \in poly(\lambda)$ is the number of the authorities), $\hat{\mathcal{B}}$ also wins CLSignForge with non-negligible probability.

□

Lemma 4. *If Strong RSA assumption holds in the group $\mathbb{Z}^*_{N_i}$ of each identity provider i, and the underlying SoK schemes are sound, then our VABS scheme achieves soundness.*

Proof Intuition. The soundness proof of [16] covers our soundness proof, since we achieve this property by limiting the use of the user secret key s via S, C_s, C_J and three SoKs ($\text{SoK}_1, \ldots, \text{SoK}_3$) related to these serial number and commitments. The serial number S, and commitments C_s and C_J are structured exactly the same as in [16]. The minor difference is that instead of ZKPoK protocols of [16], the user proves the knowledge of the same information via SoKs (which provides the same security guarantees in the random oracle mode) on the signed messages. □

Lemma 5. *If the Strong RSA assumption holds in the group $\mathbb{Z}^*_{O_i}$ of each tracing authority i, the TPKE scheme is CCA-secure (due to DDH assumption holding in $\langle g_1 \rangle$), and the underlying SoK schemes are sound; then our VABS scheme achieves reliable traceability.*

Proof. Assuming that all the underlying SoK schemes satisfies soundness and the DDH assumption holds in $\langle g_1 \rangle$, we now reduce reliable traceability of our VABS scheme to the Strong RSA assumption that should hold in all $\mathbb{Z}^*_{O_i}$ of tracing authorities. If a PPT adversary $\hat{\mathcal{A}}$ wins the reliable traceability game VABSTrace with non-negligible advantage, then we can utilize $\hat{\mathcal{A}}$ to construct a PPT algorithm $\hat{\mathcal{B}}$ that breaks unforgeability of the CL signature of at least one authority with non-negligible advantage. Since [34] already showed that their signature scheme is unforgeable under the Strong RSA assumption, $\hat{\mathcal{B}}$ can trivally be utilized for constructing a PPT algorithm that breaks the Strong RSA assumption in at least one tracing authority group. In the game VABSTrace, $\hat{\mathcal{A}}$ and $\hat{\mathcal{B}}$ play the roles of $\mathcal{A}$ and $\mathcal{C}$, respectively. In the CLSignForge game (described in the proof of Lemma 3), $\hat{\mathcal{B}}$ plays the role of $\mathcal{A}$ against

[9] If there are multiple such authorities, $\hat{\mathcal{B}}$ picks one randomly reducing its chance only inverse polynomially.

an honest challenger $\hat{\mathcal{C}}$. $\hat{\mathcal{B}}$ extracts SoKs of $\hat{\mathcal{A}}$ so that if $\hat{\mathcal{A}}$ needs to generate $\mathrm{SoK}[m]\{\xi : \xi \text{ satisfies } \zeta\}$, $\hat{\mathcal{B}}$ learns the witness ξ in addition to the message m and condition ζ, and only checks whether ξ satisfies ζ.

1. In CLSignForge, $\hat{\mathcal{B}}$ obtains 1^λ.
2. In VABSTrace, $\hat{\mathcal{B}}$ runs GlobalSetup on 1^λ and obtains $params = (q, \mathbb{G}, g, h)$. Also, it picks arbitrary $n, \delta \in poly(\lambda)$, and gives 1^λ, n, δ, $params$ to $\hat{\mathcal{A}}$. $\hat{\mathcal{B}}$ and $\hat{\mathcal{A}}$ then follow TraceSetup and publish the outputs as described. The time counter t is initialized as 1, and is started (**Step 1** of VABSTrace).
3. In VABSTrace, $\hat{\mathcal{B}}$ follows **Step 2** of the game with $\hat{\mathcal{A}}$ in the exact same way as an honest $\mathcal{C}$ would do, except for one of the tracing authorities (picked randomly by $\hat{\mathcal{B}}$) under $\hat{\mathcal{B}}$'s control. For that authority only, $\hat{\mathcal{B}}$ acts as follows. In CLSignForge, $\hat{\mathcal{B}}$ obtains the public key p. $\hat{\mathcal{B}}$ gives p to $\hat{\mathcal{A}}$ as the CL public key part of the public key of that authority (TPKE public key part is prepared as in the honest way). If $\hat{\mathcal{A}}$ asks to involve the tracing authority with public key p in a TraceIssue execution, then $\hat{\mathcal{B}}$ queries to the signing oracle in CLSignForge, receives the output and returns it to $\hat{\mathcal{A}}$. Otherwise, $\hat{\mathcal{B}}$ signs the message with the secret key of the queried authority itself.
4. In VABSTrace, $\hat{\mathcal{A}}$ eventually gives a VABS to $\hat{\mathcal{B}}$ (**Step 3** of VABSTrace). If $\hat{\mathcal{A}}$ wins, $\hat{\mathcal{B}}$ extracts the SoK_{6+k} query for that VABS to obtain the CL signature σ_{CL} that $\hat{\mathcal{A}}$ utilized.
5. In CLSignForge, $\hat{\mathcal{B}}$ outputs σ_{CL}.

Clearly, the only way that $\hat{\mathcal{A}}$ wins VABSTrace is via forging a CL signature with non-negligible probability. Since one of the generated signatures would correspond to the authority p with non-negligible probability (i.e., at least $1/K$ where $K \in poly(\lambda)$ is the number of the tracing authorities), $\hat{\mathcal{B}}$ also wins CLSignForge with non-negligible probability. □

B Efficiency Improvement Due to Bulk Update of Accumulators

Our optimization is due to the observation that in "adding or deleting several values at once" recommended in [55], the costly operation needs to be done only if $gcd(\psi, \tilde{\sigma}_{id,k}) \neq 1$, which may occur in case ψ is a multiple of $\tilde{\sigma}_{id,k}$ since $\tilde{\sigma}_{id,k}$ is prime. Let us approximate to the average frequency $\digamma$ of the costly operation to see the efficiency improvement. Let M_j be $\Upsilon(\lambda)$ bits where $\Upsilon(\cdot)$ is a polynomial, (for ease of calculation) ψ be randomly picked from $[0, 2^{\Upsilon(\lambda)})$, and each $\tilde{\sigma}_{id,k}$ be a prime in the range $[2, 2^{\Upsilon(\lambda)})$. In this range there exists roughly $\vartheta = 2^{\Upsilon(\lambda)} / \ln 2^{\Upsilon(\lambda)}$ prime numbers, and the i-th prime number can be approximated as $i \ln i$. Hence,

$$\digamma \approx \frac{1}{\vartheta}\left(\frac{2^{\Upsilon(\lambda)}/2}{2^{\Upsilon(\lambda)}} + \sum_{i=2}^{\vartheta} \frac{2^{\Upsilon(\lambda)}/(i \ln i)}{2^{\Upsilon(\lambda)}}\right) < \frac{1}{\vartheta}\sum_{i=1}^{\vartheta}\frac{1}{i} < \frac{\ln \vartheta + 1}{\vartheta} < \frac{(\Upsilon(\lambda)\ln 2)^2}{2^{\Upsilon(\lambda)}} = \mathsf{n}(\lambda)$$

where we have applied the Maclaurin-Cauchy test on the harmonic series. Thus, the check for $gcd(\psi, \tilde{gid}_k) = 1$ can be omitted by the users. For AttrRevoke, the same arp update operations is utilized by the attribute authority i via replacing σ_{id}, $\tilde{\sigma}_{id}$, and irp with σ_ω, $\tilde{\sigma}_\omega$, and arp, respectively.