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

Osman Biçer
Koç University

Alptekin Küpçü
Koç University

May 16, 2020

Abstract

Attribute based systems enable anonymity while providing access control using attributes (e.g., student). Attribute based signatures (ABS) enable signing a message with an attribute rather than an identity. 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 ensure that *even the authorities* are prevented from exposing the identity of an ABS signer, and that only joint tracing by a threshold of multiple tracing authorities is allowed. Moreover, in our solution, the authorities cannot sign on behalf of the 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. Second, we provide the first VABS scheme in a modular design such that one may utilize a subset of the features endowed by our VABS, while omitting the unnecessary organizations for better efficiency. Third, we formally 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 running times are 1.5 and 0.7 sec, respectively, and the generated signature size is below 130 KB).

Keywords— attribute based signature, anonymous credentials, access control systems, threshold cryptography.

1 Introduction

Anonymous credential schemes have been proposed for proving the ownership of an attribute, while preserving the anonymity of the user [1, 2, 3, 4, 5, 6]. Although this type of schemes has been studied in cryptography literature extensively, it is not designed for proving ownership of multiple attributes (or proving suitability for an attribute policy), and they do not prevent two individuals from colluding by combining their individual attributes. Group signature schemes are similarly designed for anonymously proving that the signer of a message belongs to some group (e.g., sharing a particular attribute) [7, 8, 9, 10, 11, 12, 13, 14], but they allow group managers to identify the signers. Ring signature schemes [15, 16, 17] also provide non-interactive anonymous self-proving, however they require the signer of the message to know all the group members' public keys and use them in the signing algorithm. This results in computation costs on the order of the number of members.

ABS. To overcome these problems *attribute based signatures* (ABS) [18, 19, 20, 21, 22, 23, 24, 25] are developed for non-interactively proving satisfiability of a Boolean attribute policy. This occurs while signing a message using the attribute tokens that the user has obtained from some attribute authority. The goal is to disclose only suitability to the given attribute 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 by combining their attributes, and defend against attempts to forge signatures without having the required attribute tokens in the policy. ABS has many application areas including attribute based messaging [26, 27], trust negotiation [28], and cloud access control [29]. Also, being non-interactive schemes, ABSs may be suitable for blockchain applications, such as Monero[1] that currently uses ring signatures [15, 16, 17]

[1] https://www.getmonero.org/

	Decentralization	Usage Limitation	Traceability	Threshold Traceability	Reliable Traceability	Attribute Revocation	User Revocation	Authority Hiding
[19]								
[20]	✓							
[21]	✓		✓					
[24]	✓		✓		✓			
VABS	✓*	✓	✓	✓	✓	✓	✓	✓

Table (1) Comparison of our VABS with the existing MA-ABS schemes. ✓ denotes that the corresponding scheme has this feature. 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).

for anonymity. To compare an ABS with similar constructions (e.g., mesh signatures [30] and anonymous credentials [31]), we refer to [18].

Multiple Authorities. Due to the lack of support for multiple attribute authorities in the initially developed ABS schemes, a multi-authority ABS (MA-ABS) scheme was proposed [19]. However, this scheme required a central master authority, so the decentralized ABS scheme of [20] was developed. Later, [21] proposed the first decentralized traceable ABS scheme that allows a tracing authority to detect the identity of the signer of an ABS. They also claim that their scheme allows "proving" that identity to a judge. Unfortunately, [21] allows each attribute authority that a signer has interacted with to obtain the user's secret token, and thus forge a signature traceable to her. [24] 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 [32]), as well as a generic construction satisfying these notions. However, these proposals still allow any malicious tracing authority to disclose the identity of the signer of any message (including honest signers).

In this paper, we confront some of the shortcomings of existing multi-authority and decentralized ABS schemes (altogether we refer by MA-ABS schemes), and achieve the simultaneous existence of 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 the controllable linkability of [22, 25, 33], as the former 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 the signatures generated by the same user. For the attribute based messaging scenario, where the ABS may be utilized for the authentication of the sender [18], periodic usage limitation can provide spam filtering by limiting the number of messages a user can send per time period. Another use is the k-times anonymous authentication [34] scenario.
- *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. This property is useful for official case resolution in practice.
- *Reliable traceability.* This includes the inability to forge a person's signature (even by the authorities) and the 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 could forge a signer's signature, it would have consumed her rights to honestly generate verifying signatures. Note that this notion also covers "traceability" of [21] and "non-frameability" of [24].
- *Dynamic revocation of attributes.* A useful property for dynamically revoking the attributes of signers that no longer deserve them by the attribute authorities (e.g., if a student graduates, her student attribute can be revoked).
- *Dynamic revocation of users.* The ability 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 or the user is deceased.
- *Authority hiding.* This feature is important for further anonymity of a signer by hiding the authorities that she has interacted with, in particular, 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). Note that this is a non-trivial task in public-key infrastructures.

We named our scheme successfully combines 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 [1, 2, 3, 4, 5, 6] are utilized for proving attributes while keeping the anonymity. [1, 6] provide n-times unlinkability, which is useful for a VABS

construction. Note that although by definition anonymous credential schemes are not expected to be non-interactive, both [1] and [6] can be converted to a non-interactive version via Fiat-Shamir transformation [35]. Yet, we stress that anonymous credential schemes, including [1, 6], are neither made for preventing *collusion* of attributes among users nor show a clear way of how to prevent it. In our solution, we build on top of [1] and improve it on many aspects (in particular, by adding collusion resistance and traceability) to get a VABS that satisfies our requirements. However, we could not use [6] for this purpose, due to the fact that it allows the authorities to obtain the secret keys of the users (and hence sign on their behalf). The k-times anonymous authentication scheme of [36] also provides a similar usage limiting functionality, yet it lacks an additional revocation feature as in [1].

Attribute based signature (ABS) schemes. ABS schemes [18, 19, 20, 21, 22, 24, 23, 25, 33, 37] are non-interactive signature solutions for anonymous attribute policy proving. The existing multi-authority and decentralized ABS schemes of [19, 20, 21, 24] lack our mentioned goals for VABS as shown in Table 1.

Functional credential schemes. The functional credential scheme of [38] 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 to be no effective way of multi-authority attribute issuing.

Group signatures. Group signature schemes [7, 8, 9, 10, 11, 12, 13, 14] are non-interactive constructions for proving that the signer of a message belongs to some group (sharing a particular attribute). In particular, revocable [13, 14], traceable [39], or distributed traceable [12], or fully dynamic model of [40] 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.

Ring signatures. Ring signature schemes [15, 16, 17] also provide non-interactive anonymous self-proving. However, they require the signer of the message to know all the group members' public keys and use them in the signing algorithm, resulting in computation on the order of the number of members.

1.2 Our Contributions

1. In Section 3, we provide the first VABS construction with modular design. The desired features can be turned on/off to efficiently achieve the requirements of the planned application. Our security definitions are novel by including the different authority architecture and allowing the proposed VABS operations; yet they are inline with previous definitions on similar paradigms.
2. In Section 4, we implement our VABS scheme with and without traceability, and show the efficiency (i.e., for a VABS with up to 20 attributes, Sign and Verify operations take below 1.5 and 0.7 sec, respectively, and the signature size is below 130 KB).
3. In the full version [41], we provide our formal definitions of the components (GlobalSetup, TraceSetup, IdPJoin, AuthJoin, UserJoin, TraceIssue, AttrIssue, Sign, Verify, UserRevoke, AttrRevoke, Trace) of a VABS scheme. In [41], we also provide our game based security definitions for anonymity, tracing reliability, signature unforgeability, and soundness.
4. In the full version [41], we prove the security of our VABS based on Strong RSA (via reduction to the security of the CL signatures [42]), DDH, and SDDHI [1] (via reduction to the pseudo random function construction of [43]) assumptions in the random oracle model.

1.3 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.

We assume that each state (or region) has an *identity provider* (e.g., civil registry authority providing national identity numbers), who is responsible for issuing/revoking global ids and ensuring that a user picks her secret key randomly (to prevent attribute collusion among users).

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 $\phi \geq 2\theta - 1$, so the majority of the tracing authorities are assumed to be honest. As long as this assumption holds, the majority can always trace and output the correct signer of a VABS in lawful cases.

There are an arbitrary number of *attribute authorities*, each responsible for issuing various attributes to the deserving parties. We only trust these authorities for proper attribute issuance and for non-disclosure of the identities of the users that have interacted with them. Yet, in our solution, 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 all such existing schemes, and are not related to our construction).

Threat model. Regarding anonymity, we allow an adversary to fully corrupt all identity provides, all attribute authorities, and $\theta - 1$ tracing authorities, to control all users except for two, and to obtain arbitrary number of VABSs on messages from any user that it chooses (even from the ones that are not under the adversary's control). We require the VABS scheme still to disallow an adversary from distinguishing the signer of a VABS subject to the following restrictions. Both users should be out of the adversary's control, should satisfy the related attribute policy of the VABS, and should not have not signed more VABSs than the limitation.

Regarding traceability, we allow the adversary to fully corrupt all identity providers, all attribute authorities, $\theta - 1$ tracing authorities except for those that can issue a particular attribute, and all users. We require the VABS scheme still to disallow an adversary from generating a VABS that cannot be traced to its original signer s by the tracing operation executed by θ tracing authorities.

Regarding unforgeability, we allow the adversary to fully corrupt all tracing authorities, all attribute authorities except for those that can issue a particular attribute, and all users except for those that have the attribute issued (still the adversary can ask for VABSs from these users). We require the VABS scheme still to disallow an adversary from generating a VABS with an attribute policy chosen by the adversary that requires that particular attribute for satisfiability and gets verified.

Regarding soundness (i.e., n-times usability), we allow the adversary to fully corrupt all identity providers, all tracing authorities, all attribute authorities, and all users. We require the VABS scheme still to disallow an adversary from generating more than the limitation number of VABSs for the same user s.

The full formal game-based definitions and formal reduction proofs exist in the full version [41].

1.4 Overview of Our Techniques

There exists some standard credential revocation techniques [44, 45] that can possibly be applied to the existing ABS schemes [18, 19, 20, 21, 22, 24, 23, 25, 33] to obtain a VABS scheme. However, achieving all VABS requirements with these schemes (in particular, usage limitation) is non-trivial. Besides, we mentioned the short-comings of functional credential and group signature schemes in Section 1.1. Therefore, instead of these constructions, we start with an n-times unlinkable anonymous credential scheme (i.e., [1]), and add the VABS requirements. We highlight that [1] does not propose a clear way of preventing *collusion* of attributes among users. By preventing collusion, we mean that a user with a *student* attribute and another user with a *disability* attribute should not be able to collude and conform to a *student with disability* policy. In our solution, we prevent such collusion while enabling multiple authorities. Moreover, although their scheme provides n-times unlinkability and credential revocation, it does not provide traceability and user revocation, which we provide. Therefore, we needed to improve [1] by many modifications, additions, and optimizations.

Techniques. When joining the system, the user interacts with an identity provider, and the user's secret key s is generated as the output. The user obtains a Camenisch-Lysyanskaya (CL) signature [42] on s and a certificate including a commitment to s and the real user identity, from the *id* provider. This is the main difference of our authority architecture from the existing multi-authority and decentralized ABS schemes [19, 20, 21]. Afterward, the user interacts with at least θ tracing authorities to obtain her tracing tokens as CL signatures on s.

After the interaction with the tracing authorities, the user can then obtain a token for an attribute ω from any qualified authority (e.g., "student" attribute from her school or ministry of education). The attribute token is generated as a CL signature on $s + H(\omega)$, where $H(\cdot)$ is a hash function modeled as random oracle. This prevents collusion among users for combining their attributes, and is one of our novelties over [1], in addition to traceability and other properties.

The user can sign a message for a policy with Boolean AND/OR[2] combinations of attributes by generating a serial number S for the signature (obtained from the user's secret s and the number J of times she has generated signature in the current period), a commitment C_s to s, a commitment C_J to J, and a tracing tag Φ (i.e., threshold encryption of g_1^s where g_1 is a group parameter), along with non-interactive zero-knowledge proofs on the message for showing that S, C_s, C_J, and Φ are constructed correctly, $J < n$ where n is the number uses allowed in a time period, and the CL signatures for her attributes that she obtained from the authorities are valid. Verification is done by checking all proofs and whether the serial number S is in the database or not.

For tracing a user, given a valid VABS, θ honest tracing authorities can reveal g_1^s, and check the database for the user's real identity. Note that due to the assumption that (at most) $\theta - 1$ tracing authorities can be adversarial, to reach a consensus on the user's identity, at least $2\theta - 1$ authorities should participate in the tracing protocol. Our technique utilizes the threshold public key encryption (TPKE) scheme of [46]. 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., [47, 48, 49, 50, 51, 52]), as long as they allow efficient zero-knowledge

[2]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 [19, 20, 21] also do not have direct NOT operation support.

proof of equality of an 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 [53, 54] are not suitable 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.

2 Preliminaries

Notation. Throughout this paper,

- $a \twoheadleftarrow B$: the value of a is picked from the set B uniformly at random,
- $a \leftarrow \mathsf{B}$: a is set as the output of a probabilistic polynomial time (PPT) B,
- $a := b$: the value of a is set as the value of b,
- $\mathsf{A}(a) \rightarrow b$: a PPT A takes as input a and its output is called b.
- $\mathsf{A}\{B_1(b_1), B_2(b_2)\} \rightarrow (c_1), (c_2)$: 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.
- $\boxed{a}$ (i.e., a inside a sharp-cornered rectangle) : a is an optional step only for achieving the user/attribute revocation feature.
- $\textcircled{a}$ (i.e., a inside a round-cornered rectangle) : a is an optional step only for achieving the reliable traceability.
- 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 [55]: 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 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 [56]. 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 [57] (for proving factorization of a strong RSA modulus), [31] (for proving that some numbers are quadratic residues of a strong RSA modulus), [58] (for proving that a committed value is in a given range), [42] (for proof of a CL signature), [59] (for proof of a discrete logarithm and opening values of Pedersen commitments [60]).

Periodic n-Times Unlinkable Anonymous Credential Scheme of [1]. We now briefly describe the n-times unlinkable anonymous credential scheme of [1] 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 $2\ell_{cnt} - 1 > n$. 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 $\mathsf{p}'\mathsf{q}'$ 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 [42] 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 [57] and that $\langle \mathbf{g} \rangle = \langle \mathbf{h} \rangle$ are quadratic residues modulo N [31]. To obtain a credential, a user first interacts with the issuer, and they run the protocol below 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 $s_1 + s_2'$. [60]. She then sends $\mathsf{C}_{s_1 + s_2'}$ to the issuer and proves that it is correctly formed via [59].
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 [42] 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.

Pseudo Random Functions (PRFs). In the PRF security game $\mathsf{DistPRF}$, the challenger $\mathcal{C}$ gives to the adversary $\mathcal{A}$ 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 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

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

Also, we call $F_s(\cdot)$ a PRF. Note that [1] 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 $s \leftarrow\!\!\!\!\leftarrow \mathbb{Z}_q^*$), if SDDHI assumption [1] holds in $\mathbb{G}$. We refer the reader to [1] for the further details.

Digital Signatures. A digital signature scheme consists of three algorithms: (1) $\mathsf{DSKeyGen}$ for public-private key pair generation, (2) DSSign for generating a signature on a transcript with the private key, (3) $\mathsf{DSVerify}$ for verifying a signature on a given transcript. The digital signatures satisfy the conventional existential unforgeability under adaptive chosen message attack (EU-ACMA) game for signatures described as $\mathsf{Sig\text{-}forge}$ in [61].

3 Our Modular VABS

In this section, we present our VABS solution in a modular manner. In the protocol and algorithm descriptions, parts that are inside $\boxed{\textit{sharp-cornered}\text{ rectangles}}$ are utilized for user/attribute revocation, and parts that are within $\boxed{\textit{round-cornered}\text{ rectangles}}$ are employed for reliable threshold traceability. If these properties are not required for the application, they can be omitted for further improving efficiency.

Global Setup. In $\mathsf{GlobalSetup}$, a cyclic group $\langle g \rangle = \mathbb{G}$ of prime order q such that $2^{\ell_q - 1} < q < 2^{\ell_q}$ is generated, where $\ell_q \in \Theta(\lambda)$. Another generator h of $\mathbb{G}$ is also generated in a distributed computation [62, 63] 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.[3] 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 $(q, \mathbb{G}, g, h)$.

Tracing Setup. In $\mathsf{TraceSetup}$, first two generators g_1 and g_2 of the group $\mathbb{G}$ are 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 [46]. Each tracing authority i runs Algorithm 1 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, NIZKPoK$_{t,1}$, and NIZKPoK$_{t,2}$, showing correct generation of the values. Each honest authority checks the transcripts of every other tracing authority, and signals an error in case of detection of any malicious behaviour.

Algorithm 1 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 (NIZKPoK$_{t,1}$, NIZKPoK$_{t,2}$).

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

Authority Join. To join the system, each identity provider or attribute authority runs Algorithm 2. The proofs of knowledge for NIZKPoK$_1$ and NIZKPoK$_3$ can be instantiated as in [57] for knowledge of strong primes, while the ones for NIZKPoK$_2$ and NIZKPoK$_4$ can be instantiated as in [31] for showing that the values are quadratic residues, respectively. Upon executing the algorithm, the authority publishes its public keys iip_i/aip_i and $irp_{i,0}/arp_{i,0}$ together with the proofs NIZKPoK$_1$, NIZKPoK$_2$, NIZKPoK$_3$, and NIZKPoK$_4$, while keeping its secret keys. For efficiency, 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 $\mathsf{DSKeyGen}$ algorithm of a conventional digital signature scheme ($\mathsf{DSKeyGen}, \mathsf{DSSign}, \mathsf{DSVerify}$) to obtain a public-private key pair (pk_{id}, sk_{id}) and publishes the public key.

User Join. To join the system, a user interacts with the identity provider in her state through an authenticated channel, where they run the $\mathsf{UserJoin}$ protocol in Figure 1, i.e., an extended version of the "Signature on a Committed Value" protocol of [42]. Note that we removed the Pedersen commitment input from the original protocol of [42], as it runs on Fujisaki-Okomoto commitment [64] after proving the equality of the committed values. 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

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

Algorithm 2 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$_4$).

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

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 [1], in our solution, a user has only one secret key s due to the removal of the double spending tag.

Tracing Issue. Upon running the protocol in Figure 1 with an identity provider, the user interacts with at least θ tracing authorities, with each of which she runs the TraceIssue protocol given in Figure 2 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 [65]) of maintaining such a consistent and consensus-based database as long as the majority of the authorities are honest (i.e., in our case $2\theta - 1 \leq \phi$). We highlight that if an identity provider does not randomize a user secret key s via picking s_2 randomly and generating the CL signature on $s_1 + s2$ as in honest execution of UserJoin, and let two different users to have the same secret key s, then the tracing authorities can detect it in the last step of their TraceIssue protocol execution.

Attribute Issue. To obtain the token for an attribute ω from an authority i, the user and the authority run the protocol in Figure 3 through a secure and authenticated channel (i.e., an extended version of the "Signature on a Committed Value" protocol of [42]), where $H : \{0,1\}^* \rightarrow \mathbb{Z}_q$ 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 3).[4]

Sign. We present our Sign algorithm with AND policy $\omega_1 \wedge \ldots \wedge \omega_k$ in Algorithm 3 without authority hiding for simplicity, where ℓ_{cnt} is a sytem parameter such that $\ell_q - 2 \geq \ell_{cnt} > n + 1/2$. We provide the modifications for authority hiding at the end of this section. The algorithm outputs a serial number S for the VABS, commitments C_J and C_s to the number J of generated signatures in the current time period and the user's secret key s, 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 schemes for SoK$_1$ and SoK$_2$ can be realized using [59] and [58] by 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 used only once).

For SoK$_3$, the signer computes the commitments $C_s := g^s h^{\rho_1}$, $C_{\rho+r} := g^{\rho+r} h^{\rho_2}$, $C_{\tilde{\sigma}_{id}} := g^{\tilde{\sigma}_{id}} h^{\rho_3}$, $C_v := v g^{\rho_4}$, $C_{\rho_4} := g^{\rho_4} h^{\rho_5}$, $C_{\rho_4 \tilde{\sigma}_{id}} := g^{\rho_4 \tilde{\sigma}_{id}} h^{\rho_6}$, 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 [42].[5] She then generates zero-knowledge *OR proofs* of all listed proofs in the mentioned proof scheme of [42] as SoKs on m. Moreover, she computes the commitments and values listed in "Efficient Proof That a Committed Value Was Accumulated" protocol in [66] to prove that the committed value in $C_{\tilde{\sigma}_{id}}$ is accumulated in the part $\mathbf{u}_{i,j+1}$ of $irp_{i,j}$ (i.e., the current revocation public key of the authority i). For SoK$_{3+1}$, ...,SoK$_{3+k}$, the signer also follows the same method as SoK$_3$ by replacing s with $s + H(\omega)$ and id with ω. Thus, these SoKs enable the signer to prove that she has a valid id from an id provider, and the required attributes from the attribute authorities, without disclosing her identity.

The tracing tag Φ is the ciphertext obtained by encrypting g_1^s with TPKE [46]. SoK$_{4+k}$ is generated 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 in the previous ones 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, the signer encrypts g_1^s within the tracing tag Φ, which has two advantages: secrecy of s and ease of SoK$_{5+k}$.

For the Sign algorithm with OR policy, the subpolicies can be combined in a SoK using [56]. For example,

[4] Unlike the ABS schemes of [19, 20, 21], we employ a hash function for randomization, instead of trusting authorities for that purpose. This is more realistic in scenarios where there exist multiple authorities and multiple attributes, and each attribute is not known from the start but rather is dynamically established.

[5] The commitments provided here and listed in [42] are in the same order.

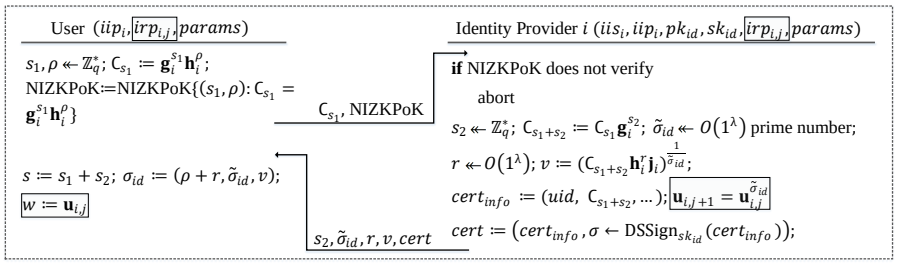


Figure (1) The UserJoin protocol between a new user and an identity provider.

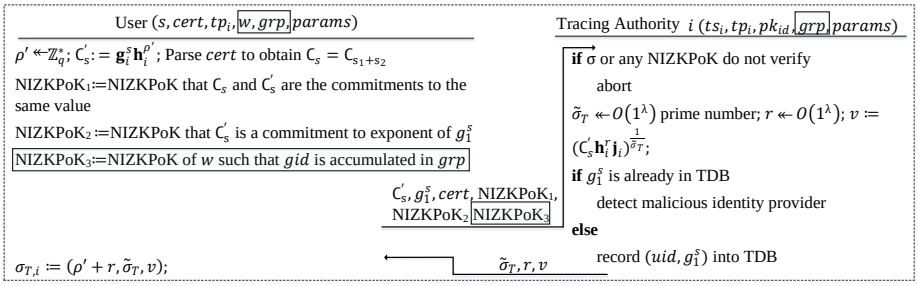


Figure (2) The TraceIssue protocol between a user and a tracing authority.

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.

Note that our signing algorithm has some improvements over what would have been obtained if one had just converted [1] to a non-interactive signature scheme. First, we provide a method for attribute authorities to generate a signature on the same secret key of the user to eliminate collusion among the users. That is, an authority issues an attribute ω to a party with secret key s by generating a CL signature on $s + H(\omega)$, which makes it intractable for another party to use that CL signature with its secret key s', since it requires that $s + H(\omega) = s' + H(\omega')$ where $s \neq s'$ and H is a random oracle. Second, regarding efficiency, 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 efficiently utilizing C_s. Third, our algorithm effectively combines the attribute proving with the threshold encryption scheme for traceability. Fourth, due to the fact that we only require the more than n signing attempts to be detectable but not de-anonymized (which is done differently during tracing), we removed the double spending tags of [1] for efficiency.

Algorithm 3 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}, $\boxed{\text{a non-revocation witness witness } w_{id} \text{ for } \sigma_{id}}$, a set $\Sigma_\beta = (\sigma_{\omega_1}, \ldots, \sigma_{\omega_k})$ of the attribute tokens for β, $\boxed{\text{a set } W_\beta = (w_1, \ldots, w_k) \text{ of non-revocation witness of those attributes}}$, $\boxed{\text{a set } \Sigma_T = (\sigma_{T,1}, \ldots, \sigma_{T,\theta}) \text{ of the tracing tokens}}$, the issuing public key iip $\boxed{\text{and the revocation public key } irp}$ of the provider of the σ_{id}, the issuing public key set AIP $\boxed{\text{and the revocation public key set } ARP}$ of the authorities that have issued Σ_β, $\boxed{\text{the tracing public key set } TP \text{ of the tracing authorities that have issued } \Sigma_T}$, 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 $\sigma = (S, \mathsf{C}_J, \mathsf{C}_s, \boxed{\Phi}, \mathrm{SoK}_1, \ldots, \mathrm{SoK}_{3+k}, \boxed{\mathrm{SoK}_{4+k}, \ldots, \mathrm{SoK}_{5+k+\theta}})$.

$\rho_1, \rho_2, \rho_3 \leftarrow \mathbb{Z}_q$; $S := g^{1/(s+t2^{\ell_{cnt}}+J)}$; $\mathsf{C}_J := g^J h^{\rho_1}$; $\mathsf{C}_s := g^s h^{\rho_2}$
$\boxed{\Phi := (g_1^{\rho_3}, g_2^{\rho_3}, g_1^s h_1^{\rho_3}, c^{\rho_3} d^{\rho_3 \kappa}) \text{ where } \kappa := H(g_1^{\rho_3}, g_2^{\rho_3}, g_1^s h_1^{\rho_3})}$
SoK_1 :=SoK$[m]\{(J, \rho_1) : J \in \{0, \ldots, n-1\} \wedge \mathsf{C}_J = g^J h^{\rho_1}\}$
SoK_2 :=SoK$[m]\{(\alpha, \gamma) : S = g^\alpha \wedge g = (\mathsf{C}_s g^{t2^{\ell_{cnt}}} \mathsf{C}_J)^\alpha h^\gamma\}$
SoK_3 :=SoK$[m]\{(s, \rho_2, \sigma_{id}, \boxed{w_{id}}) : \sigma_{id}$ is a σ_{CL} on s committed in C_s verifiable with iip, $\boxed{\text{and } w_{id} \text{ is a witness that } \tilde{\sigma}_{id} \text{ is accumulated in } irp}\}$
for $i = 1, \ldots, k$ **do**
SoK_{3+i} :=SoK$[m]\{(s, \rho_2, \sigma_{\omega_i}, \boxed{w_i}) : \sigma_{\omega_i}$ 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 $aip_i \in AIP$, $\boxed{\text{and } w_i \text{ is a witness that } \tilde{\sigma}_{\omega_i} \text{ is accumulated in } arp_i \in ARP}\}$

$\boxed{\mathrm{SoK}_{4+k}$:=SoK$[m]$ that Φ is constructed correctly
SoK_{5+k} :=SoK$[m]$ that C_s and $g_1^s h_1^{\rho_3}$ are commitments to the same value
for $i = 1, \ldots, \theta$ **do**
SoK_{5+k+i} :=SoK$[m]\{(s, \rho_2, \sigma_{T,i}) : \sigma_{T,i}$ is a σ_{CL} on s committed in C_s verifiable with $tp_i \in TP\}$

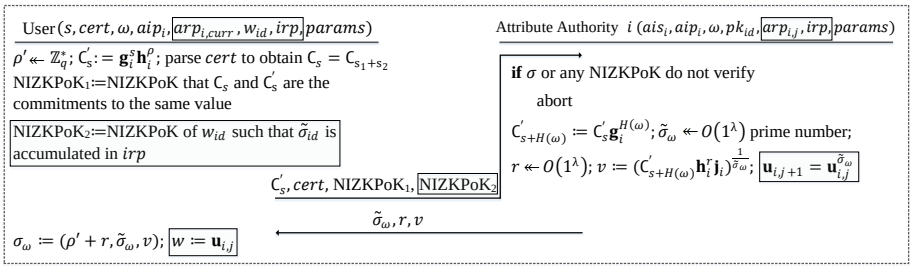


Figure (3) The AttrIssue protocol between a user and an attribute authority.

Verify. Algorithm 4 shows our Verify algorithm for an AND policy. It essentially returns 1, if all the SoKs in the signature verifies, and the serial number S is not utilized before (to ensure n-times limited use). Otherwise, it returns 0.

Algorithm 4 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}.

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

User Revoke. For UserRevoke, inline with [66], an identity provider i updates its current revocation public key $irp_{i,j} = (\mathbf{u}_{i,j}, \mathbf{g}_i', \mathbf{h}_i', M_i)$ as

$$irp_{i,j+1} = (\mathbf{u}_{i,j+1}, \mathbf{g}_i', \mathbf{h}_i', M_i) \text{ where } u_{i,j+1} := u_{i,j}^{\tilde{\sigma}_{id}^{-1} \mod 4\mathsf{p}_i''\mathsf{q}_i''} \mod \mathsf{M}_i$$

to revoke an issued σ_{id} using the related revocation handle $\tilde{\sigma}_{id}$. Note that according to [66], 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 efficiently updates 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 $w_{id,j+1} := w_{id,j}^b . irp_{i+1}^a$. This is an optimization we propose over [66], of which details are given in Section 4. We note that this optimization does not violate anonymity and signature unforgeability, since even if an attacker obtains $\tilde{\sigma}_{id}$ and w_{id} of a user the knowledge of these values are proven in zero-knowledge in Sign algorithm, and one cannot generate the SoK$_3$ without knowing (s, ρ_2, σ_{id}).

Attribute Revoke. AttrRevoke algorithm is the same as UserRevoke, where the attribute revocation handle $\tilde{\sigma}_\omega$ is used instead of $\tilde{\sigma}_{id}$, and the attribute revocation public key arp is used instead of irp.

Trace. In the Trace protocol, given a valid VABS with the tracing tag Φ, tracing authorities run the TPKE decryption [46] as an authenticated Byzantine Fault Tolerance [67, 68] protocol, so that at the end the ones that follow the protocol would come to a consensus[6] in the decryption 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 [24] is also ensured, since Φ can only be decrypted to a single value, assuming the θ tracing authorities (out of at least $2\theta - 1$) joining the operation are honest.

[6]"Interactive proofs of validity of partial decryptions" method in [46] should be utilized for honest majority to obtain the correct plaintext.

Hiding Authorities. It is possible to enhance our Sign and Verify algorithms to achieve authority hiding in applications where revealing the authorities can harm anonymity of the signers (e.g., by revealing the school authority that issued the student attribute to the user).

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 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 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 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 [56]. 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. The corresponding Verify algorithm then checks the related SoKs by taking as input all the authority public keys that are involved in signing. We note that although authority hiding increases the running times of Sign and Verify and size of the generated VABS, it preserves the efficiency of the rest of the operations in our scheme.

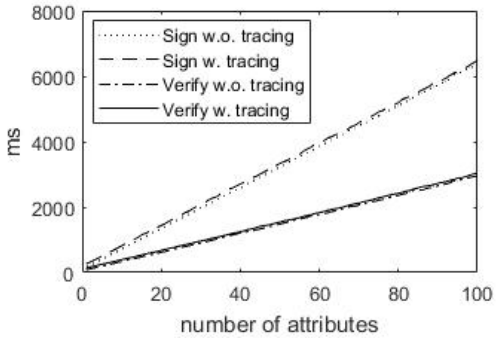
We provide the security proof of our VABS scheme in the full version [41].

4 Efficiency

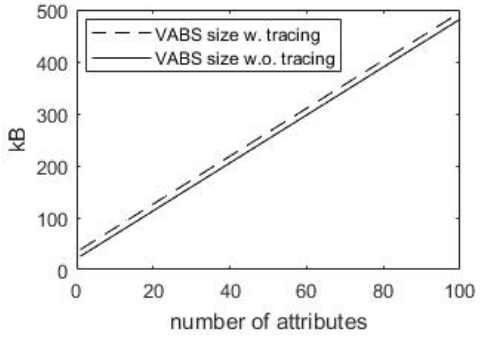
Asymptotical Analysis. Observe that GlobalSetup and TraceSetup protocols take place only once. Further, IdPJoin, AuthJoin, UserJoin, TraceIssue, AttrIssue, AttrRevoke, and Trace operations also take place infrequently, and require a constant number of public key operations. Therefore, the main concern for efficiency is on the more frequently used operations, Sign and Verify. 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 search for the serial number in the database for a time period.

Implementation Results. We implemented our scheme using the primitives implemented in the C++ Cashlib cryptographic library[7] [69]. Using our implementation, we conducted efficiency tests on a computer with Intel Xeon CPU@2.00GHz-quadcore and 8GB RAM. The RSA and prime-order group modulus lengths are set as 2048 bits, and SHA256 and DSA are used for instantiating the random oracle and the digital signature algorithm, respectively. Also, we use the hash-and-sign paradigm. Therefore, we conducted our tests on constant message size of 256 bits. We note that GlobalSetup and TraceSetup executed locally, since in practice they will take place only once. Our tests were repeated 10 times and we report average numbers. Figure 4 shows the results of our implementation with respect to various attribute policy sizes (the number of attributes in the boolean AND policy) with and without VABS traceability, respectively. We do not provide efficiency comparisons with the previous MA-ABS schemes as their respective papers do not provide implementation resuts. Note that IdPJoin and AuthJoin are only run once per authority. On average, our Sign algorithm implementation takes 54.9 ms per attribute that needs to be proven on top of a 202.1 ms fixed cost, with traceability. Also, on average, the length of the generated ABS is 4.6 kB per attribute on top of the 34.0 kB fixed size, with traceability. Further, on average, our Verify algorithm requires 29.1 ms per attribute in addition to the 116.9 ms fixed cost, with traceability. The additional costs of traceability are 39.1 ms

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



(a) **(a)** Computation times of Sign and Verify



(b) **(b)** VABS size

	GlobalSetup	TraceSetup	IdPJoin	AuthJoin	UserJoin	TraceIssue	AttrIssue
Computation Time	2455 ms	20267 ms	5838 ms	4202 ms	for user 24 ms, for authority 85 ms	for user 68 ms, for authority 196 ms	for user 48 ms, for authority 247 ms
Bandwidth	N/A	N/A	N/A	N/A	for user 0.6 kB for authority 3.5 kB	for user 6.3 kB, for authority 0.5 kB	for user 5.7 kB, for authority 0.5 kB

(c) **(c)** Computation times and bandwidth uses of GlobalSetup, TraceSetup, IdPJoin, AuthJoin, UserJoin, TraceIssue, and AttrIssue algorithms/protocols

Figure (4) The implementation results of GlobalSetup, TraceSetup, IdPJoin, AuthJoin, UserJoin, TraceIssue, AttrIssue, Sign and Verify algorithms/protocols of our VABS scheme for various attribute policy sizes. The RSA and prime-order group modulus lengths are set as 2048 bits, and SHA256 and DSA are used as the random oracle and the digital signature algorithm, respectively. In (a) and (b), w. and w.o. denote "with" and "without", respectively. The GlobalSetup and TraceSetup run locally.

on Sign, 8.3 kB on VABS size, and 32.9 ms on Verify, plus the costs for proving CL signature from each tracing authority (the same as the additional costs of each attribute).

Bulk Update of Revocation Accumulators. Our optimization is due to the observation that in "adding or deleting several values at once" recommended in [66], 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 F 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,

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

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.

References

[1] 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*, 2006.

[2] J. Camenisch and T. Groß, "Efficient attributes for anonymous credentials," *ACM TISSEC*, vol. 15, Mar. 2012.

[3] C. Garman, M. Green, and I. Miers, "Decentralized anonymous credentials," in *NDSS*, 2014.

[4] W. Lueks, G. Alpár, J.-H. Hoepman, and P. Vullers, "Fast revocation of attribute-based credentials for both users and verifiers," in *IFIP SEC*, 2015.

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

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

[7] J. Camenisch and J. Groth, "Group signatures: Better efficiency and new theoretical aspects," in *SCN*, 2005.

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

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

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

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

[12] E. Ghadafi, "Efficient distributed tag-based encryption and its application to group signatures with efficient distributed traceability," in *LATINCRYPT*, 2014.

[13] M. Nisansala, S. Perera, and T. Koshiba, "Fully secure lattice-based group signatures with verifier-local revocation," in *IEEE AINA*, 2017.

[14] K. Emura, T. Hayashi, and A. Ishida, "Group signatures with time-bound keys revisited: A new model and an efficient construction," in *ACM ASIACCS*, 2017.

[15] R. L. Rivest, A. Shamir, and Y. Tauman, "How to leak a secret: Theory and applications of ring signatures," in *TCS*, pp. 164–186, Springer, 2006.

[16] S. Wang, R. Ma, Y. Zhang, and X. Wang, "Ring signature scheme based on multivariate public key cryptosystems," *COMPUT MATH APPL*, vol. 62, no. 10, 2011.

[17] J. K. Liu and D. S. Wong, "Linkable ring signatures: Security models and new schemes," in *ICCSA*, 2005.

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

[19] 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.

[20] T. Okamoto and K. Takashima, "Decentralized attribute-based signatures," in *PKC*, 2013.

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

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

[23] B. Hampiholi, G. Alpár, F. v. d. Broek, and B. Jacobs, "Towards practical attribute-based signatures," in *SPACE*, 2015.

[24] E. Ghadafi, "Stronger security notions for decentralized traceable attribute-based signatures and more efficient constructions," in *CT-RSA*, 2015.

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

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

[27] 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.

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

[29] J. Li, M. H. Au, W. Susilo, D. Xie, and K. Ren, "Attribute-based signature and its applications," in *ACM ASIACCS*, 2010.

[30] X. Boyen, "Mesh signatures," in *EUROCRYPT*, 2007.

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

[32] 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*, 2012.

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

[34] I. Teranishi, J. Furukawa, and K. Sako, "k-times anonymous authentication (extended abstract)," in *ASIACRYPT 2004* (P. J. Lee, ed.), 2004.

[35] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *CRYPTO*, 1986.

[36] M. H. Au, W. Susilo, Y. Mu, and S. S. M. Chow, "Constant-size dynamic k-times anonymous authentication," *IEEE Systems Journal*, vol. 7, no. 2, pp. 249–261, 2013.

[37] C.-C. Drăgan, D. Gardham, and M. Manulis, "Hierarchical attribute-based signatures," in *CNS*, 2018.

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

[39] S. S. M. Chow, "Real traceable signatures," in *SAC*, 2009.

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

[41] Blinded for submission. Already exists as a technical report.

[42] J. Camenisch and A. Lysyanskaya, "A signature scheme with efficient protocols," in *SCN*, 2002.

[43] Y. Dodis and A. Yampolskiy, "A verifiable random function with short proofs and keys," in *PKC*, 2005.

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

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

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

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

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

[49] B. Libert and M. Yung, "Adaptively secure non-interactive threshold cryptosystems," in *ICALP*, 2011.

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

[51] X. J. Lin and L. Sun, "Efficient cca-secure threshold public-key encryption scheme." Cryptology ePrint Archive, Report 2013/749, 2013.

[52] 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.

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

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

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

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

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

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

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

[60] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *CRYPTO*, 1991.

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

[62] T. P. Pedersen, "A threshold cryptosystem without a trusted party," in *EUROCRYPT*, 1991.

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

[64] E. Fujisaki and T. Okamoto, "Statistical zero knowledge protocols to prove modular polynomial relations," in *CRYPTO*, 1997.

[65] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," *ACM TOPLAS*, vol. 4, July 1982.

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

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

[68] Y. Lindell, *Composition of Secure Multi-Party Protocols: A Comprehensive Study*, vol. 2815. Springer, 2003.

[69] S. Meiklejohn, C. C. Erway, A. Küpçü, T. Hinkle, and A. Lysyanskaya, "Zkpdl: A language-based system for efficient zero-knowledge proofs and electronic cash," in *USENIX Security*, 2010.

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

A VABS Operations

$\mathsf{GlobalSetup}(1^\lambda) \rightarrow params$: This is an operation that takes place once in the setup phase of the scheme. It takes as input a unary security parameter 1^λ. It outputs global setup parameters $params$ (which is assumed to include 1^λ for simplicity of the presentation). 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 the 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 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 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 its initial revocation public key arp_0.

$\mathsf{UserJoin}\{\text{User}(iip, irp_j, params), \text{Identity provider}(iis, iip, irp_i, sk_{id}, pk_{id}, params)\} \rightarrow (s, \sigma_{id}, cert, w_{id}), (irp_{i+1}, \tilde{\sigma}_{id})$: 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 irp_i, and its digital signature key pair (sk_{id}, pk_{id}), while the user knows the public values $iip, irp_i, params$. 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_{j+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 the 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}, and public values $irp, params$. 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 jointly by an attribute authority and 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 digital signature public key pk_{id} and its current revocation public key irp (i.e., the provider with whom the user ran the $\mathsf{UserJoin}$ protocol), while the user knows a user secret key s, her user certificate $cert$, ω, the non-revocation witness w_{id} of her global id σ_{id}, and the public values $aip, arp_j, irp, params$. 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 witnesses of those attributes, a set Σ_T of the θ 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 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 validity 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, an attribute 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, the 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 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.

UserRevoke$(irs, irp_j, \tilde{\sigma}_{id}, params) \rightarrow irp_{j+1}$: 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}.

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. The inputs are 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}.

Trace$\{$**for** $i = 1, \ldots, 2\theta - 1$, 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.

B VABS Security Definitions

A VABS scheme Π = (GlobalSetup, TraceSetup, IdPJoin, AuthJoin, UserJoin, TraceIssue, AttrIssue, Sign, Verify, UserRevoke, AttrRevoke, Trace) must satisfy anonymity, tracing reliability, signature unforgeability, and soundness as security requirements. In all of our game-based definitions, 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 games are as follows:

1. $\mathcal{C}$ runs 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 $\theta - 1$ 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 $\phi - \theta + 1 \geq \theta$ of them honestly follow the protocol. All tracing authorities together run 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. At any step in the games,
   (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 UserJoin with any identity provider or AttrIssue with any attribute authority for any attribute ω. For all of the users generated, $\mathcal{A}$ can request 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.

Below we provide the individual security games. Our signature unforgeability and anonymity definitions are as strong as the ones of [21] (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 via the game VABSAnonym$_{\mathcal{A},\Pi}(\lambda)$:

3. $\mathcal{C}$ generates two secret keys s_0 and s_1. $\mathcal{C}$ runs UserJoin, TraceIssue, AttrIssue, and AttrRevoke for s_0 and s_1 with authorities of $\mathcal{A}$'s choice.

4. $\mathcal{A}$ 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 $\tilde{\beta}$, $\tilde{\Sigma}_{\tilde{\beta},0}$ and $\tilde{W}_{\tilde{\beta},0}$ as the attribute tokens and non-revocation witnesses that are issued to s_0 related to $\tilde{\beta}$, $\tilde{AIP}$ 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 queries during a time period when its counter J_0/J_1 reach $n-1$.
5. $\mathcal{A}$ gives $\mathcal{C}$ an attribute policy β, and two messages m_0 and m_1 to $\mathcal{C}$, when both $J_0 < n-1$ and $J_1 < n-1$. If any of the counters is equal to $n-1$, $\mathcal{A}$ loses. If there exists any missing attribute tokens on s_0 and s_1 for conforming to β, $\mathcal{C}$ obtains them.
6. $\mathcal{C}$ 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 $\mathcal{A}$.
7. $\mathcal{A}$ 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* $\forall n, \delta \in poly(\lambda)$*, PPT adversary* $\mathcal{A}$*, there exists a negligible function* $\mathsf{n}(\cdot)$ *such that*

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

Our **reliable traceability** definition ensures that tracing is done correctly as long as the adversary controls at most $\theta - 1$ tracing authorities. Our definition covers "traceability" of [21] and "non-frameability" of [24], as long as θ tracing authorities are honest. Our notion implies "tracing soundness" of [24] by requiring a single original signer per VABS that is deterministically traced. We note that this is the first work that provides a compact tracing 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 by [12].[8] Consider the following game $\mathsf{VABSTrace}_{\mathcal{A},\Pi}(\lambda)$:

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 2 (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*

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

The **signature unforgeability** definition that we provide is similar to the unforgeability definition of [18], and is even stronger than that by allowing the adversary to issue any attribute to a user while the latter does not allow this. For our **signature unforgeability** definition, consider the following signature unforgeability game $\mathsf{VABSForge}_{\mathcal{A},\Pi}(\lambda)$:

[8] Distributed tracing of [12] 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

3. $\mathcal{A}$ is allowed to fully corrupt all ϕ tracing authorities.
4. $\mathcal{A}$ returns 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 a way that if $\mathcal{A}$ queries to an authority oracle as $\mathsf{AttrIssue}$ with ω for users under $\mathcal{A}$'s control, ω is issued but is *immediately revoked*[9].
6. $\mathcal{A}$ eventually generates a message m (not queried for signing to the users under $\mathcal{C}$'s control for the attribute policy $\beta \wedge \omega$ where β is any policy) and a VABS μ. 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 3 (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 [1], but differs slightly due to usage, i.e., ours limits the number of signatures by a party that is 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 4 (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*

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

C Security Proof of Our VABS

Theorem 1. *If the Strong RSA assumption [1, 70] holds in the groups* $\mathbb{Z}^*_{N_i}$ *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 of knowledge),* $F_{g,s}(x) = g^{1/s+x}$ *is a PRF with input* $x \in \mathbb{Z}^*_q$*, the digital signature scheme* $(\mathsf{DSKeyGen}, \mathsf{DSSign}, \mathsf{DSVerify})$ *satisfies EU-ACMA [61], the TPKE scheme of [46] is secure against chosen ciphertext attacks (CCA-secure),* $H(\cdot)$ *is modeled as random oracle, the trust 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, reliable traceability, signature unforgeability, and soundness).*

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 [1] holding in $\langle g \rangle$. Further, the CCA-security of the TPKE scheme of [46] is reducible to the DDH assumption holding in $\langle g_1 \rangle$. In what follows, we provide our full proof of Theorem 1.

Lemma 1. *If the digital signature scheme* $(\mathsf{DSKeyGen}, \mathsf{DSSign}, \mathsf{DSVerify})$ *satisfies EU-ACMA, 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 [46] is CCA-secure; then our VABS scheme achieves anonymity.*

Proof. Assuming that the digital signature scheme $(\mathsf{DSKeyGen}, \mathsf{DSSign}, \mathsf{DSVerify})$ satisfies EU-ACMA, and the underlying SoK schemes are zero-knowledge, and the threshold encryption of [46] is CCA-secure; we now reduce the anonymity of our VABS scheme to the pseudorandomness of $F_{g,s}(x) =$

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

$g^{1/s+x}$. If a PPT adversary $\hat{\mathcal{A}}$ wins the anonymity game 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 $f : \mathbb{Z}_q^* \rightarrow \langle g \rangle$ with non-negligible advantage. In the VABSAnonym game, $\hat{\mathcal{A}}$ and $\hat{\mathcal{B}}$ play the roles of $\mathcal{A}$ and $\mathcal{C}$, respectively. In the DistPRF game, $\hat{\mathcal{B}}$ plays the role of $\mathcal{A}$ against an honest challenger $\hat{\mathcal{C}}$. In 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 $\hat{\mathcal{B}}$; i.e., $\hat{\mathcal{B}}$ simulates SoKs for $\hat{\mathcal{A}}$. Note that in the TPKE scheme of [46], $\hat{\mathcal{B}}$ 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.

1. In 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 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 TraceSetup and publish the outputs as described. The time counter t is initialized as 1, and is started (Step 1 of VABSAnonym).
3. In 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. In 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. (Step 3 of VABSAnonym).
5. In VABSAnonym, a bit b is randomly picked by $\hat{\mathcal{B}}$ (normally an honest challenger picks this bit in Step 6 of the game).
6. $\hat{\mathcal{A}}$ 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 DistPRF with $t'2^{\ell_{cnt}} + J_b$, obtains the oracle output S', and then generates $\sigma \leftarrow (S', \mathsf{C}_{J_b}, \varsigma', \text{the simulation of } \Phi, \text{the simulated SoKs on } m')$, 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 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 4 of VABSAnonym).
7. In VABSAnonym, $\hat{\mathcal{A}}$ generates β, m_0 and m_1, when both $J_0 < n-1$ and $J_1 < n-1$. $\hat{\mathcal{A}}$ 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 5 of VABSAnonym).
8. In DistPRF, $\hat{\mathcal{B}}$ queries the oracle with $t2^{\ell_{cnt}} + J_b$, and obtains the oracle output S_b. In VABSAnonym, $\hat{\mathcal{B}}$ sets $\sigma_0 := (S_b, \mathsf{C}_{J_b}, \varsigma_1, \text{the simulation of } \Phi, \text{the simulated SoKs on } m_0, A)$ and $\sigma_1 := (S_{\bar{b}}, \mathsf{C}_{J_{\bar{b}}}\varsigma_2, \text{the simulation of } \Phi, \text{the simulated SoKs on } m_1, A)$ 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 6 of VABSAnonym).
9. In VABSAnonym, $\hat{\mathcal{A}}$ eventually outputs a bit b' (Step 7 of VABSAnonym). If $b = b'$, in DistPRF, $\hat{\mathcal{B}}$ outputs $F_{g,s}$. Otherwise, $\hat{\mathcal{B}}$ outputs f.

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 equations, we obtain

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

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

Substituting 1/2 for both $\mathsf{Pr}[\text{the oracle is } F_{g,s}]$ and $\mathsf{Pr}[\text{ the oracle is } f]$,

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

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

If the oracle is f, $\hat{\mathcal{A}}$ can win VABSAnonym with exactly $1/2$ probability, since its views for both $b = 0$ and $b = 1$ are statistically identical. Thus, $\frac{1}{2}\epsilon_2 = \epsilon_1$ 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. Since ϵ_2 is negligible if the PRF is indistinguishable from random, so must ϵ_1 be, showing our VABS provides anonymity. □ □

Lemma 2. *If the digital signature scheme* (DSKeyGen, DSSign, DSVerify) *satisfies EU-ACMA, 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 the digital signature scheme (DSKeyGen, DSSign, DSVerify) satisfies EU-ACMA and 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 [42] already showed that their signature scheme is unforgeable under the Strong RSA assumption, $\hat{\mathcal{B}}$ can be utilized to break 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 4), $\hat{\mathcal{B}}$ plays the role of $\mathcal{A}$ against an honest challenger $\hat{\mathcal{C}}$. $\hat{\mathcal{B}}$ extracts SoKs of $\hat{\mathcal{A}}$ so that if $\hat{\mathcal{A}}$ needs to generate $\text{SoK}[m]\{\xi : \xi \text{ satisfies } \zeta\}$, $\hat{\mathcal{B}}$ 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. □ □

Lemma 3. *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 $\mathbb{Z}_q$. Since 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 4. *If the Strong RSA assumption [1, 70] 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. We also assumue that non-revocation witnesses are unforgeable due to the Strong RSA assumption [66]. 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 [42], $\hat{\mathcal{B}}$ 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 EU-ACMA game for signatures as described as $\mathsf{Sig\text{-}forge}$ in [61]), $\hat{\mathcal{B}}$ 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 $\mathsf{VABSForge}$, $\hat{\mathcal{B}}$ 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 $\mathsf{VABSForge}$) for verification of ω. For that authority only, $\hat{\mathcal{B}}$ acts as the Steps 6 and 8 of this proof below. If there are multiple such authorities, $\hat{\mathcal{B}}$ picks one randomly reducing its chance only inverse polynomially.
4. $\hat{\mathcal{A}}$ is allowed to corrupt all ϕ tracing authorities (Step 3 of $\mathsf{VABSForge}$.
5. In $\mathsf{VABSForge}$, $\hat{\mathcal{A}}$ 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 $\mathsf{VABSForge}$).
6. In $\mathsf{CLSignForge}$, $\hat{\mathcal{B}}$ 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 $\mathsf{VABSForge}$).
7. In $\mathsf{VABSForge}$, 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 $\mathsf{VABSForge}$, if $\hat{\mathcal{A}}$ queries the authority oracle for p, then $\hat{\mathcal{B}}$ queries m_i to the signing oracle in $\mathsf{CLSignForge}$, receives and gives the output to $\hat{\mathcal{A}}$. Otherwise, $\hat{\mathcal{B}}$ signs the message with the secret key of queried authority itself.
9. In $\mathsf{VABSForge}$, $\hat{\mathcal{A}}$ eventually outputs $(m, \beta \wedge \omega, \sigma)$ to $\hat{\mathcal{B}}$ (Step 6 of $\mathsf{VABSForge}$). 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 $\mathsf{CLSignForge}$, $\hat{\mathcal{B}}$ outputs σ_{CL}.

As shown in [1], $\hat{\mathcal{A}}$ 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 $\mathsf{VABSForge}$ is via forging a CL signature. 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 $\mathsf{CLSignForge}$ with non-negligible probability. □

□

Lemma 5. *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 [1] 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 ($\mathrm{SoK}_1, \ldots, \mathrm{SoK}_3$) 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 [1]. The minor difference is that instead of ZKPoK protocols of [1], 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. □