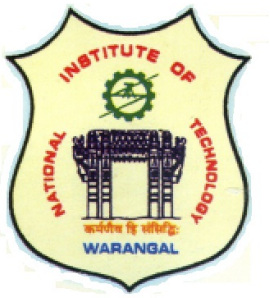
Attribute-Based Group Signature Schemes with Attribute Anonymity and Constant Size Signature



Submitted in partial fulfilment of the requirements

for the award of the degree of

DOCTOR OF PHILOSOPHY

by

Syed Taqi Ali
Roll No. 700969

Under the guidance of

Prof. B. B. Amberker

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY
WARANGAL-506004, TELANGANA, INDIA

NOVEMBER-2014

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY WARANGAL
TELANGANA-506004, INDIA

November 25, 2014

THESIS APPROVAL FOR PhD

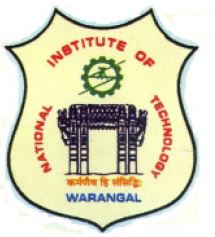
This is to certify that the thesis entitled, **Attribute Based Group Signature Schemes with Attribute Anonymity and Constant Size Signature**, submitted by **Mr. SYED TAQI ALI [Roll No. 700969]** is approved for the degree of **DOCTOR OF PHILOSOPHY** at National Institute of Technology Warangal.

Examiner

Research Supervisor
Prof. B. B. Amberker
Department of
Computer Science & Engg.
NIT Warangal

Chairman
Dr. K. Ramesh
Head, Department of
Computer Science & Engg.
NIT Warangal

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY WARANGAL
TELANGANA-506004, INDIA



CERTIFICATE

This is to certify that the thesis entitled, **Attribute-Based Group Signature Schemes with Attribute Anonymity and Constant Size Signature**, submitted in partial fulfilment of requirement for the award of degree of **DOCTOR OF PHILOSOPHY** to National Institute of Technology Warangal-506004, is a bonafide research work done by **Mr. SYED TAQI ALI [Roll No. 700969]** under my supervision. The contents of the thesis have not been submitted elsewhere for the award of any degree.

Date:

Place: Warangal

Prof. B. B. Amberker
Research Supervisor
Dept.of Computer Science & Engg.
National Institute of Technology
Warangal-506004

Dedicated

to

My Parents

Abstract

Attribute-Based Group Signature (ABGS) scheme is a kind of group signature scheme where the group members possessing certain privileges (characterized by attributes) only are eligible for signing the document. In a dynamic ABGS scheme the number of members and attributes are not fixed or known in the setup phase. In this thesis, we address various issues that arise in the design of efficient and secure dynamic ABGS schemes.

There are ABGS schemes proposed in the literature which do not provide *attribute anonymity* feature - the verifier should be able to verify whether the signer possess a valid set of attributes without learning which set of attributes he used in signing the document. We propose ABGS schemes with attribute anonymity in the random oracle model as well as in the standard model. These schemes also achieve constant size signature. We then propose an enhanced ABGS scheme by adding a new feature called *attribute tracing* - which reveals with what privilege the signer has produced the signature.

A prominent issue is membership revocation which is needed when there is a key loss, a member leaves or a member is expelled from the group. After revocation no more group signatures be allowed to produce with that revoked member's private key. This makes the ABGS scheme practical as the users can join or leave the group at any time. In *Verifier-Local Revocation* (VLR) schemes, only verifiers are involved in the revocation of a member, while the signers are not. *Backward unlinkability* ensures that even after a member is revoked, the signatures produced by the member before the revocation remain anonymous. There is an ABGS scheme with VLR feature in the literature but it neither

supports backward unlinkability nor attribute anonymity and moreover its signature size is not constant. We propose ABGS schemes with VLR and backward unlinkability feature in the random oracle as well as in the standard models. Further, we add one more security feature namely, *attribute unforgeability*. The attribute unforgeability is a special case of collusion resistance security feature which means that it should be impossible for any individual member to satisfy the predicate with invalid set of attributes. Moreover, these schemes have signatures of constant size. We prove security of all the proposed schemes under well known complexity assumptions.

Contents

List of Figures

List of Tables

Abbreviations and Symbols

Symbol	Meaning
ABGS	Attribute Based Group Signature
ABS	Attribute Based Signature
VLR	Verifier Local Revocation
NIZK	Non Interactive Zero Knowledge
NIWI	Non Interactive Witness Indistinguishability
k	security parameter
PPT	Probabilistic Polynomial Time algorithm
params	system parameter
GM	group manager
U_i	user i
Att	universal set of attributes
ζ	attribute set ($\subset Att$)
Γ	access structure
Υ	predicate
T_Υ	access tree representing the predicate Υ
$\mathcal{T}_\Upsilon$	public values associated with T_Υ
gpk	group public key used to verify the validity of the group signature
ik	issuing key used for issuing private keys to the users
ok_{User}	opening key used for opening the signer's identity from the given group signature
tk_{Att}	attribute tracing key used to trace the attributes of the group signature
(upk_i, usk_i)	verification/signing key of a signature scheme $DSig$ for user U_i
$\mathcal{A}_i \subseteq Att$	set of attributes assigned to the user U_i
A_i	The membership certificate for U_i
$T_{i,j}$	The attribute certificate of $att_j \in Att$ to U_i
sk_i	group private key for the member U_i
$\vec{reg}$	registration table with the group manager where the current group members information are stored
$\mathbb{T}$	The number of time intervals
RL_t	A set of revocation tokens of the revoked users at the interval t
$RL = \{RL_t\}_{t=1}^{\mathbb{T}}$	Revocation List
$grt[i][t]$	The token of member U_i at interval t
ϕ	null set
$\Upsilon(\zeta) = 1$	The attribute set ζ satisfies the predicate Υ
$x \in_R X$	Randomly picking an element from X
$\mathbb{Z}$	A set of integers
$\mathbb{Z}_p$	A set of integers $\{0, 1, ..., p-1\}$
$\bigwedge$	Logical AND
$\perp$	An empty element

Chapter 1

Introduction

Integrity of the data communicated over the Internet is ensured with the help of digital signature. A digital signature is a block of data sent along with the message that attests the origin of the message and the integrity of the message. User executes the signing algorithm with his private key to produce a digital signature. The verifier verifies the signature by executing the verification algorithm with the signer's public key. The first digital signature scheme was proposed by Rivest, Shamir and Adleman [95] in 1978. Group Signature allows a member of a group to anonymously make a digital signature on a message on behalf of the whole group. However, later in case of any dispute the designated *opener* or the group manager can reveal the identity of the signer. This we call as *signer tracing*. Chaum and van Heyst have first introduced the group signature [43] in 1991. The group signature has applications like keycard access to restricted areas, where it is necessary to secure the areas to only the employees of the group without tracking individual employee's movements. Other applications are company authenticating pricelist, press releases, digital contracts [40], anonymous credit cards, access control [75], e-cash [71], e-voting, e-auction [100]. The basic security requirements of group signature scheme according to [43] are *anonymity* - the group signature should not reveal the identity of the signer, and *traceability* - with the help of group opening key it should be possible to extract the identity of the signer of the group signature. Since the publication of [43], many schemes were proposed [6; 7; 16; 28; 40; 44] and more security requirements were added viz. unlinkability, unforgeability, collision resistance [9], exculpability [9] and framing resistance [44].

Bellare et al. [16] have given the formal definitions of the security properties of the group signature scheme by combining all the above security requirements into two, namely *full-anonymity* and *full traceability*. Later, Bellare et al. [20] have extended the formal definitions to dynamic group setting, where the number of group members are not fixed or known in the setup phase. The basic security requirements of a group signature scheme in the dynamic group setting are *anonymity*, *traceability* and *non-frameability*. *Non-frameability* means that even if two or more members collude including group manager, they should not be able to generate a signature which trace back to a non-colluded member.

Group signature schemes with membership revocation feature are proposed to make the scheme practical [8; 31; 90]. The membership revocation feature is needed when there is a key loss, a member leaves or a member is expelled from the group. After revocation no more group signatures are allowed to produce with that revoked member's secret key. Boneh and Shacham [31] have suggested that the best method of revocation is where a revocation messages are only sent to signature verifiers, so that there is no need to communicate to signers. This is known as *Verifier-Local Revocation* (VLR).

A well-known feature which combined with VLR is backward unlinkability. The *backward unlinkability* means that even after a member is revoked the signatures produced by the member before the revocation remain anonymous. This property was first introduced by Song in [100]. It also allows a user to come back into the group after having been revoked and use the same keys as before while remaining anonymous, i.e. there is a provision to suspend a group member for a certain period. The VLR feature is useful where signers are often offline or are computationally weak devices (mobile devices, smart cards, etc.). Some applications of VLR schemes are Direct Anonymous Attestation (DAA) in the context of Trusted Computing [35; 36], Vehicular Ad-hoc NETworks (VANETs) [101] and anonymous authentication [37]. Many group signature schemes are proposed with VLR feature either in random oracle model [31; 38; 90] or in the standard model [82].

Maji et al. [85] in 2011 have introduced an *Attribute-Based Signatures* (ABS), where a signer can sign a message with any predicate that is satisfied by his attributes.

Here, the signature reveals no information about the signer's identity or the attributes he holds but guarantees that the signer possesses the required attributes. There are many applications of ABS such as attribute-based messaging, trust-negotiation, attribute-based authentication and leaking secrets [84; 85]. Many ABS schemes in the standard model have been proposed [53; 70; 85; 92] among which the scheme presented by Herranz et al. [70] has constant length signature but for the threshold predicates. For monotone predicate Escala et al. have given the ABS scheme whose signature size is linear in terms of the size of the predicate and also it has an additional property of *revocability* which revoke the anonymity of the signer. [53]. For non-monotone predicate Okamoto et al. have proposed an ABS scheme but its signature length is not constant [92].

Attribute-Based Group Signature (ABGS) scheme is a kind of group signature scheme where the group members possessing certain privileges, characterized by attributes, only are eligible for signing the document. In ABGS scheme, each member is assigned a subset of attributes. Verifier accepts the signed document only if the associated signature proves that it is signed by the member who possess sufficient attributes that satisfy the given predicate. The security requirements of group signature scheme is also applicable to ABGS scheme.

The *predicate*, in terms of attribute relationships (the access structures), is represented by an *access tree*. For example, consider the predicate Υ for the document M: is (Institute = Univ. A) AND (TH2((Department = Biology), (Gender = Female), (Age = 50's)) OR (Position = Professor)), TH2 means the threshold gate with threshold value 2 and OR gate implies the TH1 - a threshold gate with threshold value 1. Attribute $\mathcal{A}_1$ of Alice is ((Institute := Univ. A), (Department := Biology), (Position := Postdoc), (Age := 30), (Gender := Female)), and attribute $\mathcal{A}_2$ of Bob is ((Institute := Univ. A), (Department := Mathematics), (Position := Professor), (Age := 45), (Gender := Male)). Although their attributes, $\mathcal{A}_1$ and $\mathcal{A}_2$, are quite different, it is clear that $\Upsilon(\mathcal{A}_1)$ and $\Upsilon(\mathcal{A}_2)$ hold, and that there are many other attributes that satisfy Υ. Hence Alice and Bob can generate a signature on this predicate, and according to anonymity requirement of ABGS, a signature should not reveal any information except that the attributes of the signer satisfy the given predicate Υ.

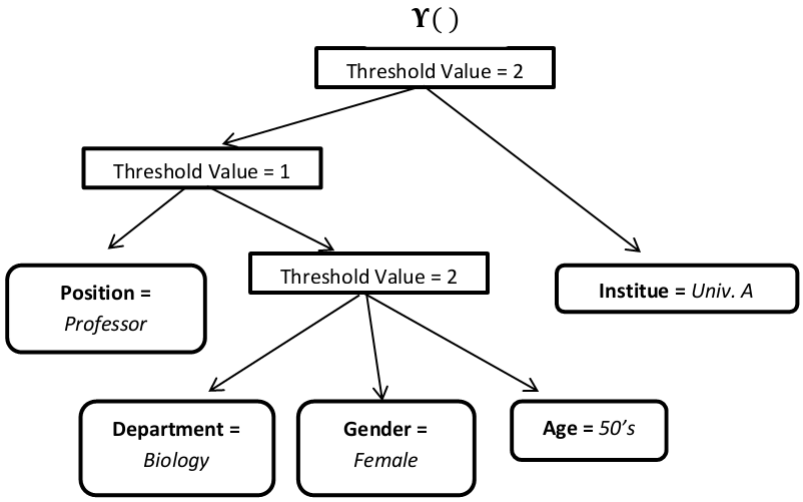


Figure 1.1: Predicate example in Tree Structure.

Thus the ABGS scheme is a kind of group signature scheme where a user with a set of attributes can prove anonymously whether he possess these attributes or not [51]. The first ABGS scheme was proposed by Dalia Khader [73] in 2007. Khader has listed *attribute anonymity* - the verifier should be able to verify whether the signer has required attributes without learning which set of attributes he used for signing, as a desirable feature to achieve. Later Khader proposed an ABGS scheme [72] with membership revocation feature without addressing attribute anonymity. Thereafter, Emura et al. in [52] have proposed a dynamic ABGS scheme, which is efficient when there is a frequent change in attribute relationships, achieved non-frameability over the Khader's schemes but it does not address the attribute anonymity issue.

1.1 Problem Description

Attribute anonymity is as necessary as *anonymity* property and in certain cases it is mandatory. Consider the case where there is a unique attribute which belongs to only one group member along with other attributes and whenever the verifier finds that attribute in the signature then he can conclude that the signature is signed by that

particular group member who alone owns that attribute. Thus anonymity itself is not preserved which is the basic security requirement in any group signature scheme.

Suppose Alice wants a document to be signed by an employee in Bob's company. Alice requires that the employee should have certain properties such as *being part of the IT staff and at least a junior manager in the cryptography team or a senior manager in the biometrics team* [73]. Now if group member with attributes *IT staff, biometric team, senior manager* sign the document and if there is only one senior manager in biometric team then the signature implies his identity. Many similar cases exist. Thus attribute anonymity is as important as anonymity property and we name *anonymity* property as a *user anonymity* property.

Attribute tracing feature allows a user to know with what privilege (an attribute set) the signer has signed the document regardless of who did it. A user has choice to query either to reveal the signer identity or the attributes of the signature. Consider the scenario where a user has multiple ways to satisfy the predicate from his attribute set. Then in some cases it is necessary to know with what attributes (privileges) the user has produced the signature and this cannot be known if the scheme possess attribute anonymity feature. We emphasize that with attribute anonymity feature the attribute tracing feature is also needed to avoid any denial by the signer. This scenario is illustrated with the following example.

Suppose there is a group member Carol with the attribute $\mathcal{A}_3$ as ((Institute := Univ. A), (Department := Biology), (Position := Professor), (Age := 55), (Gender := Male)). Obviously the $\Upsilon(\mathcal{A}_3)$ holds and moreover Carol can satisfy the predicate Υ either with the attribute ((Institute := Univ. A), (Position := Professor)) or with attribute ((Institute := Univ. A), (Department := Biology), (Age := 55)). In ABGS scheme with attribute anonymity, it is difficult to know with what attribute set the Carol has produced the signature and therefore there is a need for attribute tracing feature as well. This requirement is needed for the applications like keycard access to the restricted areas, where it may be necessary to know with what authority the employee has visited the area. Similarly in an educational institute the Director can also be a Professor in some department. He may sign a document as Director or as a Professor. Thus the attribute tracing feature should also be included along with

attribute anonymity feature in the ABGS scheme. Surely this will be a good feature for many applications. There is *no* ABGS or ABS scheme with this feature in the literature.

All the security requirements of group signature scheme are also applicable to ABGS scheme. Apart from these, collusion resistance of attributes [52] is needed. *Collusion resistance of attributes* means that a group of users with each possessing a invalid set of attributes should not be able to collude with each other to pool a valid attribute set for producing a valid group signature. This is similar to the non-frameability feature with respect to attributes. We add two more necessary security features namely, attribute anonymity and attribute unforgeability. *Attribute unforgeability* is the special case of collusion resistance of attributes security feature of [52], here attribute unforgeability means that it should be impossible for any individual member to satisfy the predicate with invalid set of attributes, which is similar to traceability feature in terms of attributes. Thus the attribute unforgeability in ABGS scheme is as important as traceability in group signature scheme. With these an ABGS scheme with attribute anonymity can be apply to applications of ABS scheme with an addon feature of revealing the signer's identity [53].

1.2 Related Work

Khader has proposed two ABGS schemes in random oracle model, one is without revocation [73] and another is with VLR feature [72]. In both the schemes the author makes use of Goyal's Attribute-Based Encryption (ABE) [66] and Boneh's group signature [28]. Emura et al. [52] have proposed the dynamic ABGS scheme with CCA-user anonymity in which they introduce the bottom-up approach for the construction of access tree in contrast to top-down approach of Goyal et al. [66]. All these schemes does not provide attribute anonymity and therefore no attribute tracing feature were introduced. Moreover, their signature length is not constant, it is linear in terms of the number of attributes, and are proven secure in the random oracle model. We note that one can design an ABGS scheme with attribute anonymity by combining an ABS scheme [70] with any group signature scheme [34; 79; 82], but it incurs cost of both the schemes. For example, to get an VLR-ABGS scheme in

standard model one can combine Herranz et al. ABS scheme [70] with Libert et al. VLR-GS scheme [82].

Following are some research questions that arise based on the above related work:

- Can we design an ABGS scheme with attribute anonymity and attribute tracing feature?
- Can we devise an ABGS scheme with VLR feature and backward unlinkability to handle the problem of key loss or a member leaves?
- Can we design a constant signature length ABGS scheme?
- Can we design an ABGS scheme which is provable secure in the standard model with the above features?
- Is it possible to add attribute revocation feature to an ABGS scheme?

1.3 Contributions

In this thesis, we address various issues in attribute-based group signatures in dynamic setting where the number of members and attributes are not fixed or known in the setup phase. The highlights of the contributions are given below:

- Introduced formal definitions of the security features viz. attribute anonymity, attribute unforgeability and collision-resistance of attributes.
- Proposed an ABGS scheme with attribute anonymity and attribute tracing in the random oracle model.
- Proposed a VLR-ABGS scheme with attribute anonymity and backward unlinkability in the random oracle model.
- Proposed an ABGS scheme with attribute anonymity in the standard model without non-frameability.

- Proposed an ABGS scheme with attribute anonymity and attribute tracing in the standard model.
- Proposed a VLR-ABGS scheme with attribute anonymity and backward unlinkability in the standard model.

The details of each scheme is given below:

1.3.1 An ABGS Scheme with Attribute Anonymity and Attribute Tracing in the Random Oracle Model

There is no ABGS scheme in which signature hides the attribute details. We propose a constant size ABGS scheme with attribute anonymity [2] by using the Emura et al.'s [51] ABGS scheme as a base scheme. We prove that the proposed ABGS scheme is secure under random oracle model with DL, q-SDH, DLDH and XDH assumptions. We use the membership certificate format of [48] that makes the scheme *non-frameable* (unforgeable even when the group manager colludes with the members) and thus the proof of traceability and non-frameability are similar to one presented in [48]. For generating the public values of the access tree we use the bottom-up approach technique introduced by Emura et al. [51]. We make use of Lagranges interpolation property to achieve the attribute anonymity. We consider the knowledge of corresponding attribute certificates combinely instead of separately as in [51]. The proposed ABGS scheme's signature length is independent of the number of attributes. We have also provided independent opening of the signer's identity known as *signer tracing* feature and opening of the attribute set identity from the signature known as *attribute tracing* feature. Thus these tasks can be assigned to two independent authorities and it is also useful when anyone wants to know the privileges of the signer rather than its identity. We adapt membership revocation mechanism from [48] and it allows group manager to revoke multiple members from the group at anytime. We have also given a short ABGS scheme whose signature length is extremely short irrespective of the number of attributes. Further, the verification cost of the proposed scheme construction is constant when compared to other ABGS schemes [51; 72] in the

random oracle model. Also the user's secret key length is shorter, achieves attribute anonymity, has attribute tracing feature and the signature length is constant.

1.3.2 A VLR-ABGS Scheme with Backward Unlinkability and Attribute Anonymity in the Random Oracle Model

To the best of our knowledge there is only one ABGS scheme with VLR feature proposed by Khader [72] but this is not dynamic and does not address backward unlinkability feature nor attribute anonymity. Moreover, in this scheme the signature length is linear in terms of the number of attributes and it is proven secure in the random oracle model. We propose a VLR ABGS scheme with backward unlinkability and attribute anonymity. We prove that the proposed scheme is secure under random oracle model with DL, q-SDH, DLIN and KEA1 assumptions. To build our scheme we use VLR-GS scheme of Nakanishi et al. [91] as the base scheme. We use the membership certificate format of [48] to make the scheme non-frameable i.e. even the group manager cannot forge signature of a trusted member. We use the bottom-up approach technique, introduced by Emura et al. in [52], for generating the public values of the access tree representing a predicate. We device an `BuildTree-Validity` algorithm which enables to verify publicly the correctness of the generated public values of the predicate and with this we reduce the trust on the group manager in producing public keys of the predicates. We device an idea to achieve attribute anonymity and we formally define the security definitions of attribute anonymity and attribute unforgeability. The proposed VLR-ABGS scheme achieves the better efficiency than the other schemes [52; 72] in terms of signing cost, verification cost, secret key length and signature length. Under special case with a single attribute the proposed scheme can be used as VLR-GS scheme. Further, when compare to VLR-GS scheme in standard model by Libert et al. [82] we observe that our scheme achieves additional features viz. non-frameability and has shorter signature length.

1.3.3 ABGS Schemes with Attribute Anonymity without Non-frameability in the Standard Model

An ABGS scheme in the standard model can be constructed with the combination of ABS [70] and group signature scheme [34; 79] in the standard model but it incurs cost of both the schemes. We propose a constant size ABGS scheme with attribute anonymity and proven that it is secure in the standard model [4]. Our construction is based on the two-level signature scheme from [34] and the technique to build the access trees from [51]. We use non-interactive proof system technique of Groth and Sahai [68] to generate the NIWI proofs for the relations in the group signature which helps to preserve the user anonymity. We prove that our scheme preserves attribute anonymity unconditionally, user anonymity in CPA attacks under subgroup decision assumption, traceability under ℓ-HSDH assumption and attribute unforgeability under DL and KEA1 assumptions, in the standard model. In contrast to other existing ABGS schemes [51; 72; 73] our scheme is built in the standard model with attribute anonymity and achieves a constant size signature independent of the number of attributes.

We also propose a constant size ABGS scheme with attribute anonymity in the standard model as above but its signature length is shorter [3]. This construction is based on the two-level signature scheme from [79] and the technique to build the access trees from [51]. We use non-interactive proof system technique [68] to hide the values in the group signature. We prove that our scheme preserves attribute anonymity unconditionally, user anonymity in CPA attacks under subgroup decision assumption, traceability under ℓ-MOMSDH and attribute unforgeability under DL and KEA1 assumptions, in the standard model.

We compare both the proposed schemes [3; 4] with the other ABGS schemes in the standard model (the combination of ABS scheme with group signature scheme in the standard model). We observe that the signing cost and verification cost is lesser. Also the size of the signature length and user's secret key length is reduced. Further, note that the efficient ABS scheme in the literature supports threshold predicates whereas our proposed schemes supports monotone predicates.

1.3.4 An ABGS Scheme with Attribute Anonymity and Attribute Tracing in the Standard Model

In the literature, there is no ABGS scheme in the standard model. However, one can get it with the combination of ABS [70] and group signature scheme [34] in the standard model. We propose an ABGS scheme with attribute anonymity and attribute tracing feature with constant size signature [61]. For our construction we use the membership certificate format of [24; 48] to achieve non-frameability and the technique to build the access trees from [51]. We use Groth-Sahai non-interactive proof system [68] to generate the NIWI proofs for the relations in the group signature. We use existing constructions [24; 51] as a base to build our scheme which addresses the said issues and proven that the construction is secure in the standard model. We observe that in contrast to other existing ABGS schemes [51; 72; 73], our scheme [61] is built in the standard model with attribute anonymity, achieves a constant size signature i.e., independent of the number of attributes, and has constant verification cost. One can also use our scheme as a group signature scheme in the standard model by distributing single attribute to all the members. When compared to other group signature schemes [80; 81], our scheme is efficient in terms of signature length and user's secret key length, but does not have revocation feature. One can also use our ABGS scheme as a ABS scheme in the standard model with the addon feature called signer tracing and attribute tracing. When compared to other ABS schemes [70; 85], our scheme achieves constant signature and it is for monotone predicates.

1.3.5 A VLR-ABGS Scheme with Backward Unlinkability and Attribute Anonymity in the Standard Model

There is no VLR-ABGS scheme in standard model in the literature. However, one can construct such a scheme with the combination of ABS [70] and VLR-GS scheme [82] in the standard model but it will not preserves non-frameability and incurs cost of both the schemes. We propose a VLR-ABGS scheme with attribute anonymity and backward unlinkability [5] which achieves constant signature size, and prove that

it is secure in the standard model. We give formal definition of VLR-ABGS scheme with attribute anonymity and extends the security definitions to include attribute anonymity and attribute unforgeability. For our construction we use the membership certificate format of [24; 48] to achieve non-frameability and the technique to build the access trees from [51]. We use Groth-Sahai non-interactive proof system [68] to generate the NIWI and NIZK proofs under SXDH settings for the relations in the group signature. We make use of VLR-GS scheme of [82] as a base construction. Similar to [51] we generate the public values of the access tree representing a predicate. We device an `BuildTree-Validity` algorithm which enables to verify publicly the correctness of the generated public values of the predicate and with this we reduce the trust on the group manager in producing public keys of the predicates. We observe that when compared to other existing ABGS schemes [51; 72; 73], our scheme [5] is built in the standard model with attribute anonymity, achieves a constant size signature i.e., independent of the number of attributes and has constant verification cost. This scheme can also be used as VLR-GS scheme since it preserve non-frameability when compared to Libert et al.'s VLR-GS scheme [82].

1.4 Organisation of Thesis

Chapter 2 gives the preliminary details regarding complexity assumptions and cryptographic primitives. In Chapter 3, we describe an ABGS scheme with attribute anonymity and attribute tracing in the random oracle model. Chapter 4 contains a VLR-ABGS scheme with backward unlinkability and attribute anonymity in the random oracle model. In Chapter 5, first we describe an ABGS scheme with attribute anonymity. Next we describe another ABGS scheme with attribute anonymity but having shorter signature length. Both the schemes are in standard model. Chapter 6 describes an ABGS scheme with attribute anonymity and attribute tracing in the standard model. Chapter 7 contains the ABGS scheme with VLR feature in the standard model. We conclude the thesis in Chapter 8.

Chapter 2

Preliminaries

In this chapter, we explain the complexity assumptions and cryptographic primitives used in the construction of the proposed schemes.

2.1 Groups

A *group* is set $\mathbb{G}$ of elements with operation * defined on it. A group has the following properties:

- If $a, b \in \mathbb{G}$ then $a * b \in \mathbb{G}$.
- The operation is associative: $(a * b) * c = a * (b * c)$.
- There is an *identity element* I such that $I * a = a * I = a$ for all $a \in \mathbb{G}$.
- For every $a \in \mathbb{G}$ there exists an inverse, denoted a^{-1}, such that $a * a^{-1} = a^{-1} * a = I$.

If $*$ is commutative i.e., $a*b = b*a$, then the group is called abelian or commutative. The number of elements of a group, $|\mathbb{G}|$, is called its order. A group is cyclic group if there exists an element $a \in \mathbb{G}$ such that every element $b \in \mathbb{G}$ can be written in the form a^x for some $x \in \mathbb{Z}$. Then the element a is called the generator of $\mathbb{G}$.

2.2 Bilinear Pairings

Let k be the security parameter. Let $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ be cyclic groups of prime order p, where $|p| = k$. Let g_1 and g_2 be generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively.

Definition 2.2.1 (Bilinear Map) *A bilinear map or pairing e is an efficiently computable function, $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ with the following properties.*

- *Bilinearity :* $\forall u, u' \in \mathbb{G}_1$ *and* $\forall v, v' \in \mathbb{G}_2$, $e(uu', v) = e(u, v)e(u', v)$ *and* $e(u, vv') = e(u, v)e(u, v')$.
- *Non-degeneracy:* $e(g_1, g_2) \neq 1_{\mathbb{G}_T}$ ($1_{\mathbb{G}_T}$ *is the* $\mathbb{G}_T's$ *identity element*).

We assume the existence of an efficient randomized procedure $\mathcal{G}$ that, on input the security parameter k, outputs a bilinear instance $\mathbf{G}= \langle p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e\rangle \stackrel{\$}{\leftarrow} \mathcal{G}(k)$, such that $|p| = k$. In practice, bilinear instances may be realized on certain algebraic varieties or curves over finite fields [14; 21; 58], by computing Weil, Tate, or a related pairing using Miller's efficient algorithm [87] or variants [13; 83]. Additional information can be found in [22; 45].

There are 3 types of bilinear maps depending on whether the group isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ and its inverse $\psi^{-1} : \mathbb{G}_1 \to \mathbb{G}_2$ are efficiently computable. Using the terminology from [59], we say that $\langle \mathbb{G}_1, \mathbb{G}_2\rangle$ is of:
"type 1" - if both ψ and ψ^{-1} are efficiently computable (this includes the case where $\mathbb{G}_1 = \mathbb{G}_2$);
"type 2" - if ψ is efficiently computable, but not ψ^{-1};
"type 3" - if neither ψ nor ψ^{-1} is efficiently computable.

2.3 Complexity Assumptions

Let k be the security parameter. Unless otherwise indicated let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ be a bilinear instance of "type 2" as defined above. Let $x \in_R X$ denote randomly picking an element x from X.

Definition 2.3.1 (Discrete Logarithm (DL) Assumption) *For all Probabilistic Polynomial Time (PPT) algorithm $\mathcal{A}$, the probability*

$$Pr[\mathcal{A}(g, g^{\xi}) = \xi]$$

is negligible function in k, where $g \in_R \mathbb{G}_1$ and $\xi \in_R \mathbb{Z}_p^*$.

Definition 2.3.2 (Decision Diffie-Hellman (DDH) Assumption) *For all PPT algorithm $\mathcal{A}$, the probability*

$$|Pr[\mathcal{A}(u, h, u^{a'}, h^{a'}) = 0] - Pr[\mathcal{A}(u, h, u^{a'}, h^{b'}) = 0]|$$

is negligible function in k, where $u, h \in_R \mathbb{G}_1$ and $a', b' \in_R \mathbb{Z}_p^*$.

Definition 2.3.3 (eXternal Diffie-Hellman (XDH) Assumption [28]) *The XDH assumption states that the DDH assumption holds in $\mathbb{G}_1$ even if DDH assumption does not hold in $\mathbb{G}_2$.*

Definition 2.3.4 (Symmetric eXternal Diffie-Hellman (SXDH) Assumption [28]) *The SXDH assumption states that the DDH assumption holds in both $\mathbb{G}_1$ and $\mathbb{G}_2$. Here the bilinear instance is of "type 3".*

Definition 2.3.5 (q-Strong Diffie-Hellman (q-SDH) Assumption [27]) *For all PPT algorithm $\mathcal{A}$, the probability*

$$Pr[\mathcal{A}(g, g', g'_1 = (g')^{\xi}, ..., g'_q = (g')^{\xi^q}) = (g^{1/(\xi+x)}, x) \wedge x \in_R \mathbb{Z}_p]$$

is negligible, where $g' \in_R \mathbb{G}_2, g = \psi(g')$ and $\xi \in_R \mathbb{Z}_p^*$.

Definition 2.3.6 (Subgroup Decision Assumption [29]) *Let $\mathcal{G}'$ be a randomized procedure which on input a security parameter k outputs a tuple $\langle p, q, \mathbb{G}, \mathbb{G}_T, e\rangle \xleftarrow{\$}$*

$\mathcal{G}'(k)$, such that $|p| = |q| = k$, $\mathbb{G}, \mathbb{G}_T$ are groups of order $n = pq$ and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a bilinear map. Then Subgroup Decision Assumption says that for all PPT algorithm $\mathcal{A}$, the probability

$$\left| Pr\left[\mathcal{A}(n, \mathbb{G}, \mathbb{G}_T, e, w) = 1 : \begin{array}{c}(p, q, \mathbb{G}, \mathbb{G}_T, e) \leftarrow \mathcal{G}'(k) \\ n = pq, w \leftarrow \mathbb{G}\end{array}\right] - Pr\left[\mathcal{A}(n, \mathbb{G}, \mathbb{G}_T, e, w^p) = 1 : \begin{array}{c}(p, q, \mathbb{G}, \mathbb{G}_T, e) \leftarrow \mathcal{G}'(k) \\ n = pq, w \leftarrow \mathbb{G}\end{array}\right] \right|$$

is negligible function in k.

Definition 2.3.7 (ℓ-Hidden Strong Diffie Hellman Assumption (ℓ-HSDH) [34]) *For all PPT algorithm $\mathcal{A}$, the probability*

$$Pr[\mathcal{A}(\tilde{g}, u, \tilde{g}^{\omega}, \tilde{g}^{\frac{1}{\omega+c_1}}, \tilde{g}^{c_1}, u^{c_1}, ..., \tilde{g}^{\frac{1}{\omega+c_{\ell-1}}}, \tilde{g}^{c_{\ell-1}}, u^{c_{\ell-1}}) = (\tilde{g}^{\frac{1}{\omega+c}}, \tilde{g}^{c}, u^{c}) \wedge c \neq c_i, \text{ for } i = 1, ..., \ell - 1]$$

is negligible function in k, where $\tilde{g}, u$ are the generators of prime order group $\mathbb{G}_p$, $\omega \in_R \mathbb{Z}_p^*$ and $c, c_i \in_R \mathbb{Z}_p$ for $i = 1, ..., \ell - 1$.

Definition 2.3.8 (ℓ-Modified One More Strong Diffie-Hellman Assumption (ℓ-MOMSDH) [79]) *Let $\mathbb{G}_p$ be the prime order group. Let p and q be primes, and $n = pq$. Then the ℓ-MOMSDH assumption says that for all PPT algorithm $\mathcal{A}$, the probability*

$$Pr[\mathcal{A}(v, v^x, u, c_1, v^{\frac{1}{x+c_1}}, c_2, v^{\frac{1}{x+c_2}}, ..., c_{\ell'}, v^{\frac{1}{x+c_\ell}}) = (v^c, v^{\frac{1}{x+c}}, u^{\frac{1}{c+m}}, m) \wedge c \neq c_i, \text{ for } i = 1, ..., \ell]$$

is negligible function in k, where u, v are the generators of $\mathbb{G}_p$, $x \in \mathbb{Z}_p^*, c_i \in \mathbb{Z}_n$ and $m \in \mathbb{Z}_n$.

Definition 2.3.9 (ℓ-Decisional Diffie-Hellman Inverse (ℓ-DDHI) in $\mathbb{G}_1$) *For all*

PPT algorithm $\mathcal{A}$, the advantage of adversary $\mathcal{A}$

$$|Pr[\mathcal{A}(g, g^y, ..., g^{y^\ell}, g^{1/y}) = 0] - Pr[\mathcal{A}(g, g^y, ..., g^{y^\ell}, D) = 0]|$$

is negligible function in k, where g is the generator of $\mathbb{G}_1$, $D \in_R \mathbb{G}_1$ and $y \in_R \mathbb{Z}_p^*$.

Definition 2.3.10 (Decisional Tripartite Diffie-Hellman (DTDH) Assumption [78]) *For all PPT algorithm $\mathcal{A}$, the advantage of adversary $\mathcal{A}$*

$$|Pr[\mathcal{A}(Z_1 = g_1^{z_1}, Z_1' = g_2^{z_1}, Z_2 = g_1^{z_2}, Z_2' = g_2^{z_2}, Z_3 = g_1^{z_3}, \eta = g_2^{z_1 z_2 z_3}) = 0] - Pr[\mathcal{A}(Z_1, Z_1', Z_2, Z_2', Z_3, g_2^u) = 0]|$$

is negligible function in k, where $z_1, z_2, z_3, u \in_R \mathbb{Z}_p^*$.

This assumption perfectly holds similar to [78] under generic group model, as it is not having efficiently computable isomorphism, $\psi : \mathbb{G}_2 \to \mathbb{G}_1$.

Definition 2.3.11 (Decisional Bilinear Diffie-Hellman (D-BDH) Assumption [32]) *For all PPT algorithm $\mathcal{A}$, the advantage of adversary $\mathcal{A}$*

$$|Pr[\mathcal{A}(g_1, g_1^\gamma, g_2, g_2^\alpha, g_2^\beta, e(g_1, g_2)^{\alpha\beta\gamma}) = 0] - Pr[\mathcal{A}(g_1, g_1^\gamma, g_2, g_2^\alpha, g_2^\beta, v) = 0]|$$

is negligible function in k, where $v \in \mathbb{G}_T$ and $\alpha, \beta, \gamma \in_R \mathbb{Z}_p^*$.

Definition 2.3.12 (q-Hybrid Hidden Strong Diffie-Hellman (q-HHSDH) Assumption in $\mathbb{G}_1, \mathbb{G}_2$ [24]) *For all PPT algorithm $\mathcal{A}$, the probability*

$$Pr[\mathcal{A}(g_1, h, g_2, g_2^\gamma, (g_1^{x_i}, g_2^{x_i}, y_i, (hg_1^{y_i})^{1/(\gamma+x_i)})_{i\in[1,q]}) = (g_1^x, g_2^x, g_1^y, g_2^y, (hg_1^y)^{1/(\gamma+x)}) \wedge (x, y) \neq (x_i, y_i)_{i\in[1,q]}]$$

is negligible, where h is the generator of $\mathbb{G}_1$ and $\gamma, x, y, x_i, y_i \in \mathbb{Z}_p^*$ for $i = 1, ..., q$.

Definition 2.3.13 (Advanced Computational Diffie-Hellman (CDH^+) Assumption [23]) *For all PPT algorithm $\mathcal{A}$, the probability*

$$Pr[\mathcal{A}(g_1, g_2, g_1^a, g_2^a, g_1^b) = g_1^{ab}]$$

is negligible function in k, where $a, b \in_R \mathbb{Z}_p$.

Definition 2.3.14 (Decision Linear (DLIN) Assumption [28]) *For all PPT algorithm $\mathcal{A}$, the advantage of adversary $\mathcal{A}$*

$$|Pr[\mathcal{A}(u, v, h, u^a, v^b, h^{a+b}) = 0] - Pr[\mathcal{A}(u, v, h, u^a, v^b, h^c) = 0]|$$

is negligible function in k, where $u, v, h \in_R \mathbb{G}_2$ and $a, b, c \in_R \mathbb{Z}_p^*$.

Definition 2.3.15 (Knowledge of Exponent Assumption - 1 (KEA1) [17; 69]) *For any adversary $\mathcal{A}$ that takes an input p, g, g^a where g is a generator of a cyclic group $\mathbb{G}_1$ and returns a pair of elements g', g'^a from $\mathbb{G}_1$, there exists an extractor $\bar{\mathcal{A}}$, which given the same inputs as $\mathcal{A}$ returns ξ such that $g^\xi = g'$.*

2.4 Cryptographic Primitives

In this section, we describe the cryptographic primitives which are used in the construction of the proposed schemes.

2.4.1 Digital Signature

A digital signature is a block of data sent along with the message that attests to the origin of the message and to the integrity of the message. User produces the signature on the message with his secret key and anyone can verify the signature using the corresponding public key. It is required that it should be infeasible to

produce a signature for any message without the knowledge of signer's secret key. The first digital signature scheme was proposed by Rivest, Shamir and Adleman [95] in 1978.

Adversary *attacks* the digital signature scheme by producing a signature of a message without knowing the secret key. Following are some attacks on the digital signatures:[64]:

- *Key-Only Attack*: Adversary knows only the public key of the signer and therefore she can only check the validity of signatures of messages given to him.
- *Known Signature Attack*: Adversary knows public key and has seen message-signature pairs chosen and produced by the legal signer.
- *Chosen Message Attack*: The adversary is allowed to ask the signer to sign messages of her choice.

After providing one of the above mentioned facility the adversary has several levels of success in forging a signature.

- *Existential Forgery*: The adversary succeeds in forging the signature of a message, not necessarily of her choice.
- *Selective Forgery*: The adversary succeeds in forging the signature of some message of her choice.
- *Universal Forgery*: The adversary is able to forge the signature of any message, even though she is unable find the secret key.
- *Total Break*: The adversary can compute the signer's secret key.

Clearly, the best digital signature scheme is that which is secure against *existential forgery under chosen message attack*.

2.4.2 Public Key Encryption

In public key encryption scheme, a secret message is sent to the receiver by encrypting it with the public key of the receiver and the encrypted message can only be decrypt by the corresponding secret key which will be with receiver. A public key encryption scheme is secure if it satisfies polynomial indistinguishability - given two plain text messages and one cipher text which is the random encryption of one of the plain text it should be impossible to relate the cipher text with its plain text without having the knowledge of secret key, known as polynomial indistinguishability [64]. There are different types of attacks on public key encryption scheme viz. Chosen-Plaintext Attack (CPA), Chosen-Ciphertext Attack (CCA) and Adaptive Chosen-Ciphertext Attack (CCA2) [86].

In CPA, adversary knows public key, has access to encryption oracle (through which she can get encryption of messages of her choice) and allowed to choose two challenge messages, after which she is given a challenge ciphertext (which is the encryption of one of the challenge messages). We say public key encryption scheme is secure under CPA if it is hard for an adversary to relate the challenge ciphertext to its plaintext.

In CCA, in addition to the above facilities an adversary is given access to decryption oracle (through which she can get the decryption of ciphertext of her choice) before the challenge ciphertext is produced. As in case of CPA, we say public key encryption scheme is secure under CCA.

In CCA2, adversary has access to decryption oracle even after the challenge ciphertext is given to her, but with the restriction that she cannot query challenge ciphertext to the decryption oracle. As in case of CPA, we say that the public key encryption scheme is CCA2 secure.

2.4.3 Interactive Proof System

Traditionally, a proof of a (target-) statement is a sequence of statements. When interpreted, this sequence eventually leads to the validity of the target-statement.

Anyone who is able to interpret the sequence can verify the proof. Such a proof is a fixed object that, once obtained, can be passed on to other people to convince them of the validity of the statement. For instance, to prove that a graph is Hamiltonian it is suffices to exhibit a Hamiltonian cycle in it. And once that cycle is generated by a *prover* then anyone who gets it can pass onto other *verifier* as a prover to prove the same.

In contrast, an interactive proof of a statement is an interactive protocol between two entities, a prover P and a verifier V. After the execution of the protocol, the verifier is convinced of the validity of the statement. Interactive proof systems were proposed by Babai et al. [11; 12] and by Goldwasser et at. [65] who also introduced the notion of zero-knowledge. Zero-knowledge proofs are those interactive proofs that convey no additional information other than the correctness of the target-statement. Moreover, the proof system is considered to be zero-knowledge if whatever the verifier can compute while interacting with the prover, the same can be computed by itself without any such interaction.

2.4.3.1 Interactive Proofs of Knowledge

Let $R \subseteq \{0,1\}^* \times \{0,1\}^*$ be a binary relation. We say that R is *polynomially bounded* if there exists a polynomial p such that $|y| <= p(|x|)$ for all $(x,y) \in R$. We let $L_R = \{x : \exists y \text{ such that } (x,y) \in R\}$ the language defined by R. The definition of interactive proof of knowledge [15] is as below:

Definition 2.4.1 *Let $R \subseteq \{0,1\}^* \times \{0,1\}^*$ be a polynomially bounded binary relation and let L_R be the language defined by R. An* interactive proof of knowledge *is a protocol (P,V), an interactive turning machine, that has the following two properties:*

- *Completeness: If $(x,w) \in R$ then $[V,P(w)](x) = accept$.*
- *Validity: There exists a probabilistic expected polynomial-time machine K (knowledge extractor) such that for every $\tilde{P}$, for all polynomials $p(.)$ and all sufficiently*

large $x \in L_R$,

$$Prob((x, K^{\tilde{P}(x)}) \in R) \geq Prob([V, \tilde{P}](x) = accept) - \frac{1}{p(|x|)}.$$

The probabilities are taken over all random choices of V, P, $\tilde{P}$ *and K respectively.*

Here $(x, w) \in R$ means w is a witness for x, $\tilde{P}$ denotes any prover who does not know the witness, $K^{\tilde{P}(x)}$ means that the knowledge extractor is given oracle access to $\tilde{P}$ on input x.

2.4.3.2 Indistinguishability of Families of Random Variables

We define the indistinguishability of families of random variables which is needed in subsequent sections.

Definition 2.4.2 *Let* $L \subset \{0,1\}^*$ *be a language and let* $A = \{A(x)\}_{x \in L}$ *and* $B = \{B(x)\}_{x \in L}$ *be two families of random variables. We say that the families A and B are*

- perfectly indistinguishable *if for all* $x \in L$ *the random variables* $A(x)$ *and* $B(x)$ *are identically distributed.*
- statistically indistinguishable *if their statistical difference is negligible or more technically, if for every polynomial* $p(.)$ *and for all sufficiently large* $x \in L$ *it holds that*

$$\sum_{\alpha \in \{0,1\}^*} |Prob(A(x) = \alpha) - Prob(B(x) = \alpha)| < \frac{1}{p(|x|)}.$$

- computationally indistinguishable *if no efficient algorithm exists that can distinguish them, i.e., for every probabilistic polynomial-time algorithm D, for every*

polynomial $p(.)$ and for all sufficiently large $x \in L$ it holds that

$$|Prob(D(x, A(x)) = 1) - Prob(D(x, B(x)) = 1)| < \frac{1}{p(|x|)}.$$

2.4.3.3 Zero-knowledge Protocols

According to the above defined three kinds of indistinguishability, there are three different degrees of zero-knowledgeness of an interactive protocol.

Definition 2.4.3 *[65] An interactive protocol (P, V) is said to be perfect/statistical/computational zero-knowledge, if for every probabilistic polynomial-time verifier $\tilde{V}$ there exists a probabilistic expected polynomial-time simulator $S_{\tilde{V}}$ so that the two families*

$$\{[\tilde{V}, P](x)\}_{x \in L} \text{ and } \{S_{\tilde{V}}(x)\}_{x \in L}$$

are perfectly/statistically/computationally indistinguishable.

When a protocol is simply said to be "zero-knowledge", it means that it is computational zero-knowledge. The parallel composition of zero-knowledge protocols is in general no longer zero-knowledge, sequential compositions can be shown to be zero-knowledge, when the definition is slightly modified [63], i.e., the verifier and the simulator are allowed an extra input z (which stores the history of interaction) the size of which is polynomially bounded in the size of x. This definition is called *auxiliary input zero knowledge.*

To prove that a protocol is zero-knowledge according to Definition 2.4.3, one would have to construct a simulator for every possible verifier. In practice, this is often done by constructing a single simulator that works for all verifiers. To match this situation, a third definition of zero-knowledge was proposed, *black-box zero-knowledge.* Here, it is required that there exists a single simulator that works for all verifiers. This single simulator is allowed to use a verifier as a black-box, i.e., the simulator can choose the input and the coin tosses of the verifier. It has been shown [63] that all protocols that are black-box zero-knowledge are a subset of all protocols that are

auxiliary-input zero-knowledge, which in turn are a proper subset of the protocols that are zero-knowledge according to Definition 2.4.3.

2.4.4 Signature Proof of Knowledge

Signature proof of knowledge (SPK) is basically a signature which proves the knowledge of the secret keys (which also obeys some relation). It is obtained by converting zero knowledge proofs of knowledge (PK) into a signature scheme by replacing the verifier by a hash function. This approach of conversion is introduced in [55] and formalised by Bellare and Rogaway in [18], called as Random Oracle Model where the hash function is replaced by an oracle. Camenisch et al. [40] gave the technique for constructing a signature of the knowledge of a discrete logarithm which can be used to build a signature that involves more complex statements. We denote it as, $SPK\{(x_1, ..., x_n) : R(x_1, ..., x_n)\}(M)$, which means a signature on message M by a signer who knows secret values $x_1, ..., x_n$ satisfying a relation $R(x_1, ..., x_n)$. As an example we define the signature of the knowledge of the discrete logarithm of y to the base g. This is basically the Schnorr signature scheme [97].

Definition 2.4.4 (SPK $\{(x) : y = g^x\}(M)$) *A pair $(c, s) \in \{0,1\}^\ell \times \mathbb{Z}_p$ satisfying $c = \mathcal{H}(S||R||M)$ with $S = g||y$ and $R = g^s y^c$, is an SPK of the discrete logarithm of a group element y to the base g of the message $M \in \{0,1\}^*$.*

Such a signature can be computed if the secret value $x = log_g y$ is known, by choosing a random integer $r \in \mathbb{Z}_p$ and computing $R = g^r$ and then $c = \mathcal{H}(g||y||R||M)$ and $s = r - cx \ mod \ p$. Here c is challenge, s is response and R is called commitment,

2.4.5 Commitment Schemes

Commitment scheme is a basic ingredient in many cryptographic protocols that are used to enable a party to commit itself to a value while keeping it secret. In a latter

stage, the commitment is "opened" and it is guaranteed that the "opening" can yield only a single value determined in the commit phase. Commitment schemes are the digital analogue of non-transparent sealed envelopes. By keeping the note in such an envelope a party commits itself to the contents of the note while keeping it secret.

A commitment scheme is an efficient two-phase two-party protocol. The first phase is called the *commit phase* and the second phase is called the *reveal phase*, by which one party, called the *sender*, can commit itself to a value so that the following requirements hold [62].

(i) *Secrecy* (*or hiding*): At the end of the first phase, the other party, called the *receiver*, does not gain any knowledge of the sender's value. This requirement has to be satisfied even if the receiver tries to cheat.

(ii) *Unambiguity* (*or binding*): Given the transcript of the interaction in the first phase, there exists at most one value that the receiver can later (i.e., in the second phase) accept as a legal "opening" of the commitment. This requirement has to be satisfied even if the sender tries to cheat.

It is required that the commit phase yields no knowledge (at least no knowledge of the sender's value) to the receiver, whereas the commit phase does "bind" the sender to a unique value (which is accepted by the receiver in the reveal phase).

2.4.5.1 Single Bit Commitment - Construction Based on any One-Way Permutation

Let $f : \{0,1\}^* \to \{0,1\}^*$ be a function, and let $b : \{0,1\}^* \to \{0,1\}$ be a predicate.

(i) *Commit Phase*: To commit to value $v \in \{0,1\}$ (let k be a security parameter), the sender uniformly selects $s \in \{0,1\}^k$ and sends the pair $(f(s), b(s) \oplus v)$ to the receiver.

(ii) *Reveal phase*: In the reveal phase, the sender reveals the bit v and the string s used in the commit phase. The receiver accepts the value v if $f(s) = \alpha$ and $b(s) \oplus v = \sigma$, where (α, σ) is the receiver's view of the commit phase.

Commitment schemes are designed using Common Reference String (CRS) model [1]. Informally, the CRS model assumes that the parties executing the protocol have access to a common string that is guaranteed to be taken from some pre-defined distribution. A commitment is *extractable* if there exists an efficient algorithm, called an extractor, capable of generating a new set of common parameters (i.e., a new CRS) whose distribution is equivalent to that of an honestly generated CRS and such that it can extract the committed value x from any commitment C. This is possible for computationally hiding commitments, such as encryption schemes: the decryption is the extraction trapdoor. We called such commitment schemes as extractable commitment schemes (*Ext-Commit*).

A commitment is *equivocable* if there exists an efficient algorithm, called an equivocator, capable of generating a new CRS and a commitment with similar distribution to those of the actual scheme such that the commitment can be opened in different ways. This is possible for computationally binding commitments only, such as the Pedersen commitment: the knowledge of the discrete logarithm is a trapdoor that allows the opening of a commitment in more than one way.

2.4.6 Non-Interactive Proof System

Non-Interactive Zero-Knowledge (NIZK) proofs were introduced by Blum et al. [25]. Their paper and subsequent work [47; 54; 74; 96] shows that NIZK proofs exist for all of NP. But, unfortunately all these NIZK proofs are very inefficient. Later, Groth and Sahai [68] have developed an efficient techniques for proving statements involving bilinear maps. We call it as Groth-Sahai proof system. They first give an efficient non-interactive witness-indistinguishable (NIWI) proofs for the simultaneous satisfiability of a set of equations involving bilinear groups. Then they show how to, under certain conditions, transform these into zero-knowledge proof systems.

2.4.6.1 Witness-indistinguishability and Zero-Knowledge

A statement may have many possible witnesses. A non-interactive proof is witness-indistinguishable if the proof does not reveal which of those witnesses the prover has used. The standard definition of witness-indistinguishability requires that proofs using different witnesses for the same statement are computationally indistinguishable. In contrast, zero-knowledge proof is a proof that shows the statement is true, but does not reveal anything else. Traditionally, zero-knowledge proof is defined by having a simulator (S_1, S_2) that can simulate respectively the Common Reference String (CRS) and a simulation trapdoor τ, and the second part of the simulation uses the simulation trapdoor to simulate proofs for statements without knowing the corresponding witnesses. The standard definition of zero-knowledge then says that real proofs on a real CRS should be computationally indistinguishable from simulated proofs on a simulated CRS [68].

2.4.7 Groth-Sahai Non-interactive Proof Systems under Subgroup Decision Assumption

Groth and Sahai gave an instantiation of their proof system under Subgroup Decision assumption, Symmetric External Diffie-Hellman (SXDH) assumption and Decisional Linear (DLIN) assumption. Here we describe the Groth-Sahai proof system under Subgroup Decision assumption.

In [68], Groth and Sahai have given a non-interactive witness-indistinguishable (NIWI) proof system for the pairing product equation of the form,

$$\prod_{i=1}^{\hat{n}} e(\mathcal{A}_i, \mathcal{X}_i) \prod_{i=1}^{\hat{n}} \prod_{j=1}^{\hat{n}} e(\mathcal{X}_i, \mathcal{X}_j)^{\gamma_{ij}} = t_T \tag{2.1}$$

for the variables $\mathcal{X}_1, ..., \mathcal{X}_{\hat{n}} \in \mathbb{G}$ and constants $t_T \in \mathbb{G}_T, \mathcal{A}_1, ..., \mathcal{A}_{\hat{n}} \in \mathbb{G}, \gamma_{ij} \in \mathbb{Z}_n$, for $i, j \in \{1, ..., \hat{n}\}$.

The prover in a Groth-Sahai proof system knows secret values $\{\mathcal{X}_i\}_{i=1}^{\hat{n}}$ and wants to prove that these values satisfy the above pairing product equation. In the instantiation based on the Subgroup Decision Assumption under the setup $(n, G, G_T, e, g), (p, q)$ where $n = pq, p$ and q are primes, the common reference string (CRS) is h where for soundness setting $h = g^{rp}$ and for witness-indistinguishable setting $h = g^r$, for random $r \in \mathbb{Z}_n^*$. The commitments to the group elements $\mathcal{X}_1, ..., \mathcal{X}_{\hat{n}} \in \mathbb{G}$ are made as follows

$$\mathcal{C}_i = \mathcal{X}_i h^{t_i}$$

for randomly chosen $\vec{t} \in \mathbb{Z}_n^{\hat{n}}$.
The proof for the pairing product equation (2.1) is

$$\pi = \prod_{i=1}^{\hat{n}} \mathcal{A}_i^{t_i} \prod_{i=1}^{\hat{n}} \prod_{j=1}^{\hat{n}} \mathcal{X}_j^{t_i(\gamma_{ij}+\gamma_{ji})} \prod_{i=1}^{\hat{n}} \prod_{j=1}^{\hat{n}} h^{t_i t_j \gamma_{ij}}$$

Upon receiving $\vec{\mathcal{C}} = \{\mathcal{C}_1, ..., \mathcal{C}_{\hat{n}}\}, \{\pi_\ell\}_{\ell=1}^{N_p}$ where N_p is the total number of pairing product equations, for each pairing product equation with proof π_ℓ the verifier checks that

$$\prod_{i=1}^{\hat{n}} e(\mathcal{A}_i, \mathcal{C}_i) \prod_{i=1}^{\hat{n}} \prod_{j=1}^{\hat{n}} e(\mathcal{C}_i, \mathcal{C}_j)^{\gamma_{ij}} \stackrel{?}{=} t_T e(h, \pi_\ell).$$

In [68], Groth and Sahai have shown that the above NIWI proof has perfect completeness, perfect L_{co}-soundness and composable witness-indistinguishability. *composable witness-indistinguishability* is the stronger notion of witness-indistinguishability, which means that the adversary cannot distinguish a real CRS from a simulated CRS. Here, the subgroup decision assumption implies that soundness and witness-indistinguishability CRS's are indistinguishable.

The size of the proof is $\hat{n} + N_p$ group elements in $\mathbb{G}$, where $\hat{n}$ is the number of variables in $\vec{\mathcal{X}} = \{\mathcal{X}_1, ..., \mathcal{X}_{\hat{n}}\}$ and N_p is the total number of pairing product equations.

2.4.7.1 Groth-Sahai Non-interactive Proof Systems under SXDH settings

Here we show under SXDH settings how Groth-Sahai proof system [68] commit the signature elements. The commitment key consists of $\vec{u} = (\vec{u_1}, \vec{u_2} = \vec{u_1}^t) \in \mathbb{G}_1^{2\times 2}$

and $\vec{v} = (\vec{v_1}, \vec{v_2} = \vec{v_1}^{t'}) \in \mathbb{G}_2^{2\times 2}$, where $\vec{u_1} = (g_1, g_1^{\alpha})$ and $\vec{v_1} = (g_2, g_2^{\alpha'})$ for some $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$ and $t, t', \alpha, \alpha' \in \mathbb{Z}_p^*$. There exist two initializations of the parameters either in the perfectly binding setting, or in the perfectly hiding one. And these initializations are indistinguishable under SXDH assumption which will be used in the simulation. We note that for equal dimension vectors or matrices A and B containing group elements, $A \odot B$ denotes their entry-wise product. We note $\mathcal{C}(X) = (1, X) \odot \vec{u_1}^r \odot \vec{u_2}^s \in \mathbb{G}_1^2$ is a commitment of a group element $X \in \mathbb{G}_1$ with a random $r, s \in \mathbb{Z}_p^*$. Similarly a group element of $\mathbb{G}_2$ is committed using $\vec{v_1}$ and $\vec{v_2}$ with $r', s' \in_R \mathbb{Z}_p^*$. An element is always committed in the group ($\mathbb{G}_1$ or $\mathbb{G}_2$) it belongs to. If one knows the commitment key in the perfectly binding setting, one can extract the value of X, else it is perfectly hidden. We note $\mathcal{C}^{(1)}(x) = \vec{\varphi}^x \odot \vec{u_1}^r \in \mathbb{G}_1^2$ is a commitment of a scalar embedded in $\mathbb{G}_1$ as g_1^x, where $\vec{\varphi} = \vec{u_2} \odot (1, g_1)$ and $r \in_R \mathbb{Z}_p^*$. If one knows the commitment key in the perfectly binding setting, one can extract the value of g_1^x else x is perfectly hidden. The same things can be done in $\mathbb{G}_2$, if we want to commit a scalar, embedding it in $\mathbb{G}_2$, we denote it as $\mathcal{C}^{(2)}(x)$ and $\vec{\varphi'} = \vec{v_2} \odot (1, g_2)$.

Using Groth-Sahai technique one can have Non-interactive Witness-Indistinguishable (NIWI) proofs for the committed variables that satisfy the set of equations (either pairing-product equation or multi-exponentiation equation or quadratic equation). The whole proof consists of one commitment per variable and two proof elements $\Theta = (\vec{\pi}, \vec{\theta})$ (each contains a constant number of group elements) per equation.

For the variables $\mathcal{X}_1, ..., \mathcal{X}_m \in \mathbb{G}_1, \mathcal{Y}_1, ..., \mathcal{Y}_n \in \mathbb{G}_2, x_1, ..., x_{m'}, y_1, ..., y_{n'} \in \mathbb{Z}_p$.

- The pairing-product equations are of the form

$$\prod_{i=1}^{n} e(\mathcal{A}_i, \mathcal{Y}_i) \prod_{i=1}^{m} e(\mathcal{X}_i, \mathcal{B}_i) \prod_{i=1}^{n} \prod_{j=1}^{m} e(\mathcal{X}_i, \mathcal{Y}_j)^{\gamma_{ij}} = t_T, \tag{2.2}$$

  for the constants $\mathcal{A}_i \in \mathbb{G}_1, \mathcal{B}_i \in \mathbb{G}_2, t_T \in \mathbb{G}_T$ and $\gamma_{ij} \in \mathbb{Z}_p$. The proof cost 4 elements in each group. Linear pairing-product equation (when $\gamma_{ij} = 0$ for all i, j) proof cost 2 elements of respective group.

- The multi-exponentiation equations in $\mathbb{G}_1$ are of the form

$$\prod_{i=1}^{n'} \mathcal{A}_i^{y_i} . \prod_{i=1}^{m} \mathcal{X}_i^{b_i} . \prod_{i=1}^{n'} \prod_{i=1}^{m} \mathcal{X}_i^{y_i \gamma_{ij}} = T_1, \tag{2.3}$$

  for the constants $\mathcal{A}_i, T_1 \in \mathbb{G}_1$ and $b_i, \gamma_{ij} \mathbb{Z}_p$. The proof cost 2 elements in $\mathbb{G}_1$ and 4 elements in $\mathbb{G}_2$. Linear multi-exponentiation equations of the type $\prod_{i=1}^{n'} \mathcal{A}_i^{y_i} = T_1$ (resp. $\prod_{i=1}^{m} \mathcal{X}_i^{b_i} = T_1$) demand 1 element in $\mathbb{G}_1$ (resp. 2 elements in $\mathbb{Z}_p$) .

- Similarly for multi-exponentiation equations in $\mathbb{G}_2$.

- And the quadratic equations in $\mathbb{Z}_p$ are of the form

$$\sum_{i=1}^{n'} a_i y_i + \sum_{i=1}^{m'} x_i b_i + \sum_{i=1}^{m'} \sum_{j=1}^{n'} \gamma_{ij} x_i y_i = t \bmod p, \tag{2.4}$$

  for the constants $a_i, b_i, \gamma_{ij}, t \in \mathbb{Z}_p$. The proof requires 2 elements in each group. Linear equation proof requires 1 element in $\mathbb{Z}_p$.

The details on construction of proofs is given in [68]. Multi-exponentiation equation allows zero-knowledge proofs with no additional cost. A trapdoor in the simulated Common Reference String (as in WI setting) makes it possible to simulate the proofs without knowing witness and the distribution of simulated proofs are indistinguishable to real proofs.

But for the pairing-product equations it is not known to always have zero-knowledge proofs. To prove the equations of the form (2.2) in Non-interactive Zero-knowledge (NIZK) needs auxiliary variables and the proof size is not independent of the number of variables. If $t_T = 1_{\mathbb{G}_T}$ in equation (2.2), the NIZK simulator can always use $\mathcal{X}_i = 1_{\mathbb{G}_1}, \mathcal{Y}_i = 1_{\mathbb{G}_2}$ as witnesses. If $t_T = \prod_{j=1}^{n'} e(g_j, h_j)$ for known group elements $g_1, ..., g_{n'} \in \mathbb{G}_1, h_1, ..., h_{n'} \in \mathbb{G}_2$, the simulator can prove that

$$\prod_{i=1}^{n} e(\mathcal{A}_i, \mathcal{Y}_i) \prod_{i=1}^{m} e(\mathcal{X}_i, \mathcal{B}_i) \prod_{i=1}^{n} \prod_{j=1}^{m} e(\mathcal{X}_i, \mathcal{Y}_j)^{\gamma_{ij}} = \prod_{j=1}^{n'} e(g_j, \mathcal{Z}_j), \tag{2.5}$$

and that introduced variables $\mathcal{Z}_j$ satisfy the linear equations $\mathcal{Z}_j = h_j$ for $j \in \{1, ..., n'\}$. Since NIZK proofs exist for linear equations and the proof of equation

(2.5) can be simulated using witnesses $\mathcal{X}_i = 1_{\mathbb{G}_1}, \mathcal{Y}_i = \mathcal{Z}_i = 1_{\mathbb{G}_2}$. When t_T is an arbitrary element of $\mathbb{G}_T$, the NIZK proofs for pairing-product equations are currently not known to exist.

2.4.8 Group Signatures

A Group Signature scheme allows a member of a group to sign anonymously on behalf of the group. However, later in case of any dispute the designated *opener* or the group manager reveals the identity of the signer. This is called *signer tracing* feature. Chaum and van Heyst first introduced the group signature [43] in 1991. Bellare et al. [16] in 2003 have formalized the definitions of the group signature scheme and categorized the security requirements into two core requirements, namely *Full-Anonymity* and *Full Traceability*.

Full-Anonymity: Adversary should have negligible probability in determining under which of the two identities (which she knows) the target signature was produced. Additionally the adversary may corrupt all the members of group, including the suspected signers. Also, the adversary has privilege to query to get the signer's identity of the signatures (an analogy to the definition of the encryption scheme against CCA) of her choice (except the challenge signature).

Full-Traceability: A group of colluding members should not be able to produce a valid group signature such that the opening algorithm fails to trace a member belonging to a colluding group, and this should hold even if the colluding group knows the opening key, which is used to open the signatures.

Bellare et al. [20] have extended the definitions to dynamic group settings, where the number of group members are not fixed or known in the setup phase, i.e. user can join the group at any time. The basic security requirements of group signature scheme in dynamic group settings are *anonymity*, *traceability* and *non-frameability* [20].

Anonymity means that the signature should not reveal the identity of the signer. *Traceability* means that the valid signature should always trace back to the valid

Table 2.1: Notations for group signature scheme

Symbol	Meaning
k	security parameter
params	system parameter
GM	group manager
U_i	user i
gpk	group public key used to verify the validity of the group signature
ik	issuing key used for issuing private keys to the users
ok_{user}	opening key used for opening the signer's identity from the group signature
(upk_i, usk_i)	verification/signing key of a signature scheme $DSig$ for user U_i $(i = 1, 2, ..., n)$, $n = O(k)$
sk_i	group private key for the member U_i
$\vec{reg}$	registration table maintained by the group manager where the current group members information are stored
ϕ	null set

identity with the help of the group opening key. *Non-frameability* means that even if two or more members *including* group manager collude, they should not be able to generate a signature which trace back to a non-colluded member.

The formal definition of group signature scheme under dynamic group setting is as follows. We assume that each user U_i owns a pair (usk_i, upk_i) of secret and public keys certified by the PKI.

Definition 2.4.5 (Group Signature) *An group signature scheme consists of following algorithms. Unless otherwise mentioned, algorithms are randomized.*

- $params \leftarrow \mathtt{Setup}(1^k)$: *This algorithm takes the security parameter k as an input and returns the system parameter params.*
- $(gpk, ik, ok_{user}) \leftarrow \mathtt{KeyGen}(params)$: *This algorithm takes the system parameter params, and returns a group public key gpk, an issuing key ik and a user opening key ok_{user}.*
- $sk_i \leftarrow \mathtt{Join}(\langle params, gpk, ik, upk_i, \mathcal{A}_i\rangle, \langle params, gpk, upk_i, usk_i\rangle)$: *This is an interactive group joining protocol between a user U_i (using his secret key usk_i) and the GM (using the issuing key ik). In the protocol U_i ends with a member private key sk_i and GM ends with an updated registration table $\vec{reg}$.*

- $\sigma \leftarrow \texttt{Sign}(params, gpk, sk_i, M)$: *This algorithm takes* $params, gpk, sk_i$ *and a message* M *as an input and returns a group signature* σ *on* M.

- $0/1 \leftarrow \texttt{Verify}(params, gpk, M, \sigma)$: *This is a deterministic algorithm verifies the validity of the group signature* σ *against* gpk *and returns* 1 (*a valid signature*) *otherwise* 0(*invalid signature*).

- $i/\perp \leftarrow \texttt{Open}(params, gpk, ok_{user}, \sigma, M, r\vec{e}g)$*: This is a deterministic algorithm which takes as input* $params, gpk, ok_{user}, \sigma, M$ *and* $r\vec{e}g$*, and returns either* $i \geq 1$ *or* $\perp$*. If* i*, specifies that the group member with identity* i *has produced* σ*, and if* $\perp$*, then no group member produced* σ *on* M.

Entities: There are several entities in group signature scheme:

- The group manager `GM`, also known as *Issuer*, has issuing key ik using which he enrolls a user U_i into the group by issuing a user's private key sk_i, after running interactive `Join` algorithm with the user.

- The *Opener* has user opening key ok_{user} by which he is able to open the signature and reveal the user identity through `OpenUser` algorithm.

- Group members or signers who are having their private keys sk_i. They run `Sign` algorithm to produce a group signature on a document M.

- Outsider or verifier who can only verify the group signature using the group public key gpk.

Remark `Setup` and `KeyGen` algorithms are run by some trusted party and he will distribute the appropriate keys to the concerned entities.

Definition 2.4.6 (`Correctness`) *Correctness requires that for all* $params \leftarrow \texttt{Setup}(1^k)$, *all* $(gpk, ik, ok_{user}) \leftarrow \texttt{KeyGen}(params), sk_i \leftarrow \texttt{Join}(\langle params, gpk, ik, upk_i \rangle, \langle params,$

$gpk, upk_i, usk_i\rangle)$ *and all* $M \in \{0,1\}^*$,
if $\mathrm{U}_i \in r\vec{e}g$ *and* $\sigma = \texttt{Sign}(params, gpk, sk_i, M)$ *then*

$$1 \leftarrow \texttt{Verify}(params, gpk, M) \bigwedge i \leftarrow \texttt{OpenUser}(params, gpk, ok_{user}, \sigma, M, r\vec{e}g)$$

holds.

The adversary can run the following versions of Join protocol (similar to [24]):

- Either through the joinP-oracle (passive join), which means that it creates an honest user for whom it does not know the private keys: the index i is added to the HU (Honest Users) list;
- or through the joinA-oracle (active join), which means that it interacts with the group manager to create a user under its control: the index i is added to the CU (Corrupted Users) list.

Note that when the adversary is given the issuing key (the group manager is corrupted) then the adversary does not need access to the joinA oracle since it can simulate it by itself, to create corrupted users (that are not necessarily in CU). After a user is created, the adversary plays the role of corrupted users, and can interact with honest users, granted some oracles:

- corrupt(i), if $i \in$ HU, provides the specific private key of this user. The adversary can now control it during the whole simulation. Therefore i is moved from HU to CU;
- sign(i, M), if $i \in$ HU, plays as the honest user i to generate a signature on message M;
- openusr(M, σ), if (M, σ) is valid, returns the identity i of the signer.

Definition 2.4.7 (`User anonymity`) *We say that the group signature scheme preserves user anonymity if for all PPT* $\mathcal{A}$*, the probability that* $\mathcal{A}$ *wins the following game is negligible.*

- ***Setup:*** *The challenger* $\mathcal{C}$ *runs* $(gpk, ik, ok_{user}) \leftarrow \texttt{KeyGen}(params)$. $\mathcal{C}$ *gives* gpk, ik *to* $\mathcal{A}$.
- `Phase1` : $\mathcal{A}$ *is given access to the oracles:* joinP, corrupt, sign *and* openusr.
- `Challenge` : $\mathcal{A}$ *outputs* M^* *and an uncorrupted users* $\mathrm{U}_{i_0}, \mathrm{U}_{i_1}$ *(i.e.* $i_0, i_1 \notin \mathsf{CU}$*).* $\mathcal{C}$ *randomly selects* $\kappa \in_R \{0,1\}$ *and responds with a group signature* $\sigma^* \leftarrow \texttt{Sign}(params, gpk, sk_{i_\kappa}, M^*)$.
- `Phase 2` : $\mathcal{A}$ *can make queries similar to Phase* 1. *However* $\mathcal{A}$ *cannot make query to* corrupt *on* i_0 *and* i_1 *at any time.*

Output: *Finally,* $\mathcal{A}$ *outputs a bit* κ'*, and wins if* $\kappa' = \kappa$.

The advantage of $\mathcal{A}$ *is defined as* $Adv^{user-anon}(\mathcal{A}) = |Pr(\kappa = \kappa') - \frac{1}{2}|$.

Thus there should not exists any PPT adversary to link a group signature to a signer with non-negligible probability.

Definition 2.4.8 (`Traceability`) *We say that the group signature scheme preserves traceability if for all PPT* $\mathcal{A}$*, the probability that* $\mathcal{A}$ *wins the following game is negligible.*

- ***Setup:*** *The challenger* $\mathcal{C}$ *runs* $(gpk, ik, ok_{user}) \leftarrow \texttt{KeyGen}(params)$. $\mathcal{C}$ *gives* gpk and ok_{user} *to* $\mathcal{A}$.
- ***Queries:*** $\mathcal{A}$ *is given access to the oracles:* joinP, joinA, corrupt *and* sign.
- ***Output:*** $\mathcal{A}$ *outputs a message* M^* *and a group signature* σ^*.

$\mathcal{A}$ *wins if*
(1) $\texttt{Verify}(params, gpk, M^*, \sigma^*) = 1$ *and*
(2) $\texttt{OpenUser}(params, gpk, ok_{user}, \sigma^*, M^*, \vec{reg}) = \perp$.
The advantage of $\mathcal{A}$ *is defined as the probability that* $\mathcal{A}$ *wins.*

Thus it should be impossible to produce an untraceable valid group signature by any PPT adversary.

Definition 2.4.9 (`Non-frameability`) *We say that the group signature scheme preserves non-frameability if for all PPT $\mathcal{A}$, the probability that $\mathcal{A}$ wins the following game is negligible.*

- ***Setup:*** *The challenger $\mathcal{C}$ runs $(gpk, ik, ok_{user}, tk_{att}) \leftarrow \texttt{KeyGen}(params)$. $\mathcal{C}$ gives gpk, ik* and *ok_{user} to $\mathcal{A}$.*
- ***Queries:*** *$\mathcal{A}$ is given access to the oracles:* $\mathsf{joinP}, \mathsf{corrupt}$ *and* sign.
- ***Output:*** *Finally, $\mathcal{A}$ outputs a message M^* and a group signature σ^*.*

$\mathcal{A}$ wins if
(1) $\texttt{Verify}(params, gpk, M^*, \sigma^*) = 1$,
(2) $\texttt{OpenUser}(params, gpk, ok_{user}, \sigma^*, M^*, \vec{reg}) = i^*$,
(3) $i \in \mathsf{HU}$.
The advantage of $\mathcal{A}$ is defined as the probability that $\mathcal{A}$ wins.

Thus even the group manager should not be able to forge a group signature which trace back to a honest member.

2.4.8.1 Revocation

To make the scheme practical a membership revocation [8; 31; 90] feature need to be incorporated, which is needed when there is a key loss, a member leaves or a member is expelled from the group. After revocation no more group signatures are allowed to produce with that revoked member's secret key. The best known method of revocation, among the methods listed by Boneh et al. [31] is Verifier-Local Revocation (VLR) which require to sends a revocation message only to the signature verifiers, so that there is no need to communicate with other members.

Another well-known feature which is combined with VLR is backward unlinkability. The *backward unlinkability* means that even after a member is revoked the signatures produced by the member before the revocation remain anonymous. This property was first introduced by Song in [100]. It also allows a user to come back into the group after having been revoked and use the same keys as before while remaining anonymous, i.e. there is provision to suspend a group member for a certain period. There are group signature schemes proposed with VLR feature either in random oracle model [31; 38; 90] or in the standard model [82].

2.4.9 Access Structure

Let $Att = \{att_1, att_2, ..., att_m\}$ be a set of attributes. For $\Gamma \subseteq 2^{Att} \backslash \{\emptyset\}$, Γ satisfies the *monotone* property if $\forall B, C \subseteq Att, B \in \Gamma$ *and* $B \subseteq C$, then $C \in \Gamma$ holds. An *access structure* (respectively, monotone access structure) is a collection (respectively, monotone collection) Γ of non-empty subsets of Att, i.e., $\Gamma \subseteq 2^{Att} \backslash \{\emptyset\}$ [66; 72].

A *predicate* Υ is a boolean function with literals as attributes. The notation $\Upsilon(\zeta) = 1, \zeta \subset Att$ expresses the fact that a set of attributes ζ satisfies the predicate Υ. The *access structure* Γ *of a predicate* Υ is a collection of non-empty subset of attributes $\zeta \subset Att$ such that $\Upsilon(\zeta) = 1$.

In *threshold predicate*, atleast a threshold number of attributes are needed to satisfy the predicate. It is expressed using a Threshold gate. A *monotone predicate* is a predicate which is expressed using AND, OR and Threshold gates, an example is given in Chapter 1. It covers the threshold predicate as a special case. A *non-monotone predicate* is expressed using NOT, AND, OR and Threshold gates. It covers monotone predicate as a special case. We restrict our attention to monotone predicates.

2.4.10 Access Tree

An access tree is used for expressing an access structure of a predicate Υ by using a tree structure. An access tree T_Υ consists of threshold gates as non-leaf nodes and

attributes as leaves. Let l_x be the number of children of node x, and k_x $(0 < k_x \leq l_x)$ be the threshold value on the threshold gate of node x. A threshold gate x is satisfied if the number k_x of l_x children branching from the node x are satisfied. Note that if the number of children of a node is equal to the threshold value then it is an AND gate and if the threshold value is one then it is an OR gate. Satisfying a leaf means owning an attribute. The notation $Leaves \models T_\Upsilon$ expresses the fact that a set of attributes *Leaves* satisfies the predicate Υ.

2.4.11 Assignment of Secret Values to Access Tree Using Bottom-Up Approach

To build an access tree for our ABGS schemes we use bottom-up approach introduced by Emura et al. [51]. Here we give the description of the approach. Let *index* be a function which returns the unique identity number of a node. Let p be a prime number. Let x represent any node in the access tree, then l_x represents the number of children of x and k_x denotes the threshold value of the node x. Let T be an access tree of a predicate.

`AddDummyNode`(T): This algorithm takes as input an access tree T, and returns the *extended access tree* T^{ext} with dummy nodes on T.

(i) For an interior node x of T, the number of dummy nodes (leaves) $l_x - k_x$ is added to x's children.

(ii) Change the threshold value of x from k_x to l_x.

(iii) All nodes are assigned unique index numbers from $\mathbb{Z}_p^*$.

(iv) The resulting tree, called T^{ext}, is output.

Let D_T be a set of dummy nodes determined by `AddDummyNode`. Let $s_j \in \mathbb{Z}_p^*$ be a secret value for an attribute $\mathrm{att}_j \in Att$. Let $S = \{s_j\}_{\mathrm{att}_j \in Att}$.

`AssignedValue`(p, S, T^{ext}): This algorithm takes as input p, S and T^{ext} and returns a secret value $s_x \in \mathbb{Z}_p^*$ for each dummy node $x \in D_T$ and a secret value of the root node of T^{ext}. Let $\{child\}_x$ be the set of node x's children except the dummy nodes, and $\{d\}_x$ be the set of node x's dummy nodes.

(i) For the interior node x of T^{ext}, a polynomial q_x of degree $l_x - 1$ is assigned as follows:

  (a) q_x is the polynomial of degree $l_x - 1$ passing through $(index(x'), s_{x'})$, where $x' \in \{child\}_x$. Note that $|\{child\}_x| = l_x$, so we can easily construct the unique polynomial.

  (b) For the dummy node $d_j \in \{d\}_x$, the secret value $s_{d_j} = q_x(index(d_j))$ is assigned.

  (c) For x, $s_x = q_x(0)$ is assigned.

(ii) Repeat the above procedure up to the root node, $s_T = q_{root}(0)$ is the secret value of T.

(iii) Output $\{s_{d_j}\}_{d_j \in D_T}$ and s_T.

`MakeSimplifiedTree`$(Leaves, T^{ext})$: This algorithm takes as input the set $Leaves \subseteq Att$ satisfying $Leaves \models T$, where T is the original access tress of T^{ext} and returns the product of Lagrange coefficients Δ_{leaf}.

(i) The set of attributes $\{\mathrm{att}_j\}_{\mathrm{att}_j \in Att \setminus Leaves}$ are deleted from T^{ext}.

(ii) Interior nodes x having children less than the threshold value (namely, l_x) are deleted from T^{ext} along with x's descendants.

(iii) Let D_T^{Leaves} be the set of dummy nodes which have remained after the steps 1 and 2, and T^{Leaves} be the access tree after 1 and 2.

(iv) We assume that the leaves are at depth 0. For each node x of T^{Leaves} except root, define L_x as follows:

(a) Define the depth 1 subtree of T^{Leaves} with x as leaf node. Let c_x be the set of indices of leaves of the subtree.

(b) Compute $L_x = \prod_{k \in c_x \setminus \{index(x)\}} \frac{-k}{index(x)-k}$.

(v) Let $leaf \in Leaves \cup D_T^{Leaves}$ be a leaf node of T^{Leaves}.
For $leaf$, we define Δ_{leaf} as follows:

(a) Let $Path_{leaf} = \{leaf, parent_1, ..., parent_{n_{leaf}} = root\}$ be the set of nodes that appears in the path from $leaf$ to root node.

(b) Compute $\Delta_{leaf} = \prod_{node \in Path_{leaf} \setminus \{root\}} L_{node}$.

Output $\Delta_{leaf} (\forall leaf \in Leaves \cup D_T^{Leaves})$.

Clearly,

$$\sum_{\text{att}_j \in Leaves} \Delta_{\text{att}_j} s_j + \sum_{d_j \in D_T^{Leaves}} \Delta_{d_j} s_{d_j} = s_T \tag{2.6}$$

holds. An example is given in [51].

2.4.12 Linear Encryption

Linear Encryption (LE) scheme is the natural extension of ElGamal encryption scheme whose security depends on the DLIN assumption. In this scheme, a user's public key is a triple of generators $u, v, h \in \mathbb{G}_1$; private key is the exponents $x, y \in \mathbb{Z}_p$ such that $u^x = v^y = h$.

To encrypt a message $M \in \mathbb{G}_1$, choose a random values $a, b \in \mathbb{Z}_p$, and output the triple $(u^a, v^b, M.h^{a+b})$.

To recover the message from an encryption (T_1, T_2, T_3), the user computes $T_3/(T_1^x.T_2^y)$.

By a natural extension of the proof of security of ElGamal, LE is semantically secure against a CPA, assuming DLIN holds [28].

2.4.13 Boyen et al.'s Two-level Signature Scheme

We use the two level hierarchical signature scheme proposed by Boyen and Waters in [34] which is existential unforgeable against chosen message attacks. In the first level the certificate is signed by the group manager and in the second level a short signature on message M is produced. This signature is based on the short signature proposed by Boneh et al. in [26].

2.4.13.1 Scheme

Let k be the security parameter. The user identities $\mathrm{id} \in \{0,1\}^k$ and the message M are taken as binary strings of fixed length $m' = O(k)$.

`Setup`(1^k): Let $\mathbb{G}$ and $\mathbb{G}_T$ be the groups of order $n = pq$ for which there exists a bilinear map e from $\mathbb{G} \times \mathbb{G}$ to $\mathbb{G}_T$, where p and q are primes of size k. Choose a generator $g \in \mathbb{G}_p$, where $\mathbb{G}_p$ is a subgroup of $\mathbb{G}$ with order p. Thus all group elements in the two-level signature scheme will have prime order p in $\mathbb{G}$ and $\mathbb{G}_T$. Select secret integers $\alpha, z \in \mathbb{Z}_p^*$ at random. Compute $Z = g^z$ and $A = e(g,g)^\alpha$. Next, select two integers $y, z' \in \mathbb{Z}_p$ and a vector $\vec{z} = (z_1, ..., z_{m'}) \in \mathbb{Z}_p^{m'}$ at random. Output the public parameters PP and the master key MK as,

$$PP = \{g, Z = g^z, u = g^y, v' = g^{z'}, v_1 = g^{z_1}, ..., v_{m'} = g^{z_{m'}}, A = e(g,g)^\alpha\} \in \mathbb{G}^{m'+4} \times \mathbb{G}_T$$

$$MK = (z, g^\alpha)$$

The public parameters, PP, also implicitly include k, m', and a description of $(p, \mathbb{G}, \mathbb{G}_T, e)$.
`Extract`(PP, MK, id): It outputs the private key for a user with identity id. Choose a secret value $s_{\mathrm{id}} \in \mathbb{Z}_p$. Output user private key,

$$k_{\mathrm{id}} = (k_{\mathrm{id},1}, k_{\mathrm{id},2}, k_{\mathrm{id},3}) = ((g^\alpha)^{\frac{1}{z+s_{\mathrm{id}}}}, g^{s_{\mathrm{id}}}, u^{s_{\mathrm{id}}}) \in \mathbb{G}^3$$

`Sign`(PP, k_{id}, M): To sign a message represented as a bit string $M = (\mu_1, ..., \mu_{m'}) \in \{0,1\}^{m'}$, using a private key k_{id}, select a random $s \in \mathbb{Z}_p$ and compute $\mathcal{F}(M) =$

$v'\Pi_{j=1}^{m'} v_j^{\mu_j}$, and output

$$\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4) = (k_{\mathrm{id},1}, k_{\mathrm{id},2}, k_{\mathrm{id},3}.\mathcal{F}(M)^s, g^{-s}) \in \mathbb{G}^4$$

$\texttt{Verify}(PP, M, \sigma)$: The signature σ is valid for a message M if the following equations holds,

$$e(\sigma_1, \sigma_2 Z) \stackrel{?}{=} A$$

$$e(\sigma_2, u) \stackrel{?}{=} e(\sigma_3, g).e(\sigma_4, \mathcal{F}(M))$$

Theorem 2.4.10 *Consider an adversary $\mathcal{A}$ that existentially forges the hybrid two-level signature scheme in an adaptive chosen message attack. Assume that $\mathcal{A}$ makes no more than $\ell - 1 \ll p$ signature queries and produces a successful forgery with probability ϵ in time t. Then there exists an algorithm $\mathcal{B}$ that solves the ℓ-HSDH problem with probability $\tilde{\epsilon} \approx \epsilon/(4m'\ell^2)$ in time $\tilde{t} \approx t$ where m' is the length of the message.*

Proof The proof is given in the paper [34].

2.4.14 Liang et al.'s Two-level Signature Scheme

We use the two level signature scheme proposed by Liang et al. in [79] which is existential unforgeable against chosen message attacks. In the first level, the certificate is signed by the group manager and in the second level a short signature on message M is produced. This signature is based on the short signature proposed by Boneh et al. in [26] and improvement over Boyen et al.'s two-level signature scheme [34].

2.4.14.1 Scheme

Let k be the security parameter. User's identity id and the message M are chosen from $\{0,1\}^k$.

`Setup`(1^k): Select the primes p and q of size k. Define the groups $\mathbb{G}$ and $\mathbb{G}_T$ of order $n = pq$, for which there exists a bilinear map e from $\mathbb{G} \times \mathbb{G}$ to $\mathbb{G}_T$. Choose generators $g, u \in \mathbb{G}_p$, where $\mathbb{G}_p$ is a subgroup of $\mathbb{G}$ of order p. Generate master key $MK = z \in \mathbb{Z}_p^*$. Compute $Z = g^z$. Define collision-resistant hash function $\mathcal{H} : \{0,1\}^k \to \mathbb{Z}_p$. Output the system parameters $params$, the public parameter PP and the master key MK as,

$$params = \{p, q, n, \mathbb{G}, \mathbb{G}_T, \mathbb{G}_p, e, \mathcal{H}\}, PP = \{g, u, Z\}, MK = z$$

`Extract`(PP, MK, id): It outputs the private key for a user with identity id. Choose a secret value $s_{\mathrm{id}} \in_R \mathbb{Z}_p$. Output user private key,

$$k_{\mathrm{id}} = (k_{\mathrm{id},1}, k_{\mathrm{id},2}) = (s_{\mathrm{id}}, g^{\frac{1}{z+s_{\mathrm{id}}}}) \in \mathbb{Z}_p \times \mathbb{G}_p$$

`Sign`(PP, k_{id}, M): The signature of message M with private key k_{id} is,

$$\sigma = (\sigma_1, \sigma_2, \sigma_3) = (g^{s_{\mathrm{id}}}, g^{\frac{1}{z+s_{\mathrm{id}}}}, u^{\frac{1}{s_{\mathrm{id}}+\mathcal{H}(M)}})$$

`Verify`(PP, M, σ): The signature σ is valid if the following equations hold, else it is invalid.

$$e(Z\sigma_1, \sigma_2) \stackrel{?}{=} e(g, g)$$

$$e(g^{\mathcal{H}(M)}\sigma_1, \sigma_3) \stackrel{?}{=} e(g, u)$$

Theorem 2.4.11 *The Two-level signature scheme is $(t, q_e, q_s, \epsilon)-$ secure against existential forgery under a chosen message attack provided that $(t', q, \epsilon_{qSDH})-SDH$ assumption and $(t'', \ell, \epsilon_{MOMSDH})-MOMSDH$ assumption hold in $\mathbb{G}_p$, where $\epsilon \leq 2q_s\epsilon_{qSDH} + 2\epsilon_{MOMSDH}$ and $t \approx max(t', t''), q \geq q_s + 1$ and $\ell \geq q_e + q_s$.*
Here, t is the time taken to forge the two-level signature, q_s is the total number of signature queries, q_e is the total number of extraction queries, t' is the time take to solve q-SDH problem, ϵ_{qSDH} is the advantage of solving q-SDH problem, t'' is the

time required to solve ℓ-MOMSDH problem and ϵ_{MOMSDH} is the advantage of solving ℓ-MOMSDH problem.

Proof The proof is given in the paper [79].

2.4.15 Waters Signature

We use a slight variant of Waters signature [23; 102], in the SXDH setting: Given three generators $(g_1, h, g_2) \in \mathbb{G}_1^2 \times \mathbb{G}_2$, a public key $pk = (g_1^t, g_2^t)$, the secret key t to sign a message M, a user simply needs to pick a random scalar s and compute $\sigma = (h^t.\mathcal{F}(M)^s, g_2^s)$. Here, $\mathcal{F}$ is the waters function defined as $\mathcal{F}(M) = v'\Pi v_j^{m_i}$, where (v', v_j) are independent generators of $\mathbb{G}_1$ and $M = (m_i)$. The verification simply consists in checking if $e(\sigma_1, g_2) = e(h, pk_2).e(\mathcal{F}(M), \sigma_2)$. This scheme is proven to be existentially unforgeable under CDH$^+$ assumption.

2.4.16 Attribute-Based Signature

Maji et al. [85] introduced an Attribute-Based Signatures (ABS), where a signer can sign a message with a predicate that is satisfied by his attributes. Here, the signature reveals no information about the signer's identity or the attributes he holds and guarantees that the signer possesses the required attributes. Many ABS schemes in standard model have been proposed [53; 70; 85; 92], among which the scheme presented by Herranz et al. [70] has constant length signature but for the threshold predicates. For monotone predicates Escala et al. have given the ABS scheme whose signature size is linear in terms of the size of the predicate and with an additional property of *revocability*, which revokes the anonymity of the signer [53]. For non-monotone predicates Okamoto et al. have proposed an ABS scheme but its signature length is not constant [92]. The revocability feature of ABS is same as signer tracing feature of group signature scheme which reveals the signer's identity from the signature.

We give the formal definition of ABS scheme [85].

Definition 2.4.12 (ABS) *An ABS scheme consists of the following algorithms. Unless otherwise indicated, algorithms are randomized.*

- $params \leftarrow \texttt{Setup}(1^k)$: *This algorithm is run by signature trustee which takes the security parameter k as an input and returns the public reference information $params$.*
- $(APK, ASK) \leftarrow \texttt{KeyGen}(params)$: *This algorithm is run by an attribute issuing authority which takes the public reference information $params$, and returns a attribute public key APK and an attribute secret key ASK.*
- $SK_{\mathcal{A}} \leftarrow \texttt{AttrGen}(ASK, \mathcal{A} \subseteq Att)$: *This is takes the ASK and attribute set $\mathcal{A}$ as an input and returns the attribute keys $SK_{\mathcal{A}}$.*
- $\sigma \leftarrow \texttt{Sign}(params, APK, SK_{\mathcal{A}}, M, \Upsilon)$: *This algorithm takes $params, APK, SK_{\mathcal{A}}$, message M and the predicate Υ as an input, where $\Upsilon(\mathcal{A}) = 1$, and returns a signature σ on M.*
- $0/1 \leftarrow \texttt{Verify}(params, APK, M, \Upsilon, \sigma)$: *This is a deterministic algorithm verifies the validity of the signature σ against APK and returns $1/0$. If 1 then the algorithm claims that the σ is a valid signature, otherwise, σ is invalid.*

Entities: Following are the entities in ABS scheme:

- The *signature trustee* who setups the ABS scheme by generating the public reference information *params*.
- The *attribute issuing authority* with attribute secret keys ASK issue an attribute keys for the attribute set $\mathcal{A}$ to the user by running an algorithm `AtrrGen`.
- A signers who are having their attribute private keys $SK_{\mathcal{A}}$. They run `Sign` algorithm to produce a signature on a document M with predicate Υ if they possess valid attribute set $\mathcal{A}$ which satisfies the predicate.

- Outsider or verifier can verify the signature using the public key, *params* and APK.

Definition 2.4.13 (`Correctness`) Correctness *requires that for all* $params \leftarrow \texttt{Setup}(1^k)$, *all* $(APK, ASK) \leftarrow \texttt{KeyGen}(params)$, *all* $\mathcal{A} \subseteq Att$, $SK_{\mathcal{A}} \leftarrow \texttt{AttrGen}(ASK, \mathcal{A} \subseteq Att)$, *all* Υ *and all* $M \in \{0,1\}^*$, *all claim-predicate* Υ *such that* $\Upsilon(\zeta) = 1$ *and all signatures* $\sigma = \texttt{Sign}(params, APK, SK_{\mathcal{A}}, M, \Upsilon)$, *we have* $\texttt{Verify}(params, APK, M, \Upsilon, \sigma) = 1$.

Perfect Privacy: The signer privacy is only relies on the signature trustee and not the attribute-issuing authority. Even a malicious and computationally unbounded attribute-issuing authority cannot link a signature to a set of attributes or the signing key used to generate it.

Unforgeability: Any signature which could not have been legitimately made by a single one of the adversary's signing keys is considered a forgery [85].

2.4.17 Attribute-Based Group Signature

ABGS scheme is a group signature scheme where the group members possessing certain privileges, characterized by attributes, are only eligible for signing the document [51; 72; 73]. In ABGS, each member is assigned some subset of attributes. The predicates in terms of attribute relationships (the access structures) are represented by an *access tree* and it is associated to the document. The group members whose attributes satisfy the access tree can sign the associated document.

Attribute anonymity means the verifier should be able to verify whether the signer has required attributes without learning which set of attributes he used for signing. Attribute tracing feature allows a user to know with what privilege (an attribute set) the signer has signed the document regardless of who did it. A user has choice to query either to reveal the signer identity or the attributes of the signature. A VLR ABGS scheme is an ABGS scheme where the revocation process does not influence

Table 2.2: Notations for ABGS scheme

Symbol	Meaning
k	security parameter
params	system parameter
GM	group manager
U_i	user i
Att	universal set of attributes
Υ	predicate
T_Υ	access tree representing the predicate Υ
$\mathcal{T}_\Upsilon$	public values associated with T_Υ
gpk	group public key used to verify the validity of the group signature
ik	issuing key used for issuing private keys to the users
ok_{user}	opening key used for opening the signer's identity from the given group signature
tk_{att}	attribute tracing key used to trace the attributes of the group signature
(upk_i, usk_i)	verification/signing key of a signature scheme $DSig$ for user U_i
$\mathcal{A}_i \subseteq Att$	set of attributes assigned to the user U_i
sk_i	group private key for the member U_i
$r\vec{e}g$	registration table with the group manager where the current group members information are stored
ϕ	null set
$\Upsilon(\zeta) = 1$	denotes that the attribute set ζ satisfies the predicate Υ

the activity of the signers. In VLR scheme, members are revoked by publishing the relative part of the secret value of the members, namely *revocation token.*

We give the formal definition of ABGS scheme.

Definition 2.4.14 (ABGS) *An ABGS scheme consists of the following algorithms. Unless otherwise indicated, algorithms are randomized.*

- $params \leftarrow \texttt{Setup}(1^k)$: *This algorithm takes the security parameter k as an input and returns the system parameter params.*
- $(gpk, ik, ok_{user}, tk_{att}) \leftarrow \texttt{KeyGen}(params)$: *This algorithm takes the system parameter params, and returns a group public key gpk, an issuing key ik, a user opening key* ok_{user} *and an attribute tracing key* tk_{att}*.*
- $sk_i \leftarrow \texttt{Join}(\langle params, gpk, ik, upk_i, \mathcal{A}_i\rangle, \langle params, gpk, upk_i, usk_i\rangle)$: *This is an interactive group joining protocol between a user* U_i *(using his secret key* usk_i*)*

and the **GM** *(using the issuing key ik and the attributes* $\mathcal{A}_i \subseteq Att$ *for* U_i*). In the protocol* U_i *ends with a member private key* sk_i *and* **GM** *ends with an updated registration table* $r\vec{e}g$*.*

- $\sigma \leftarrow \texttt{Sign}(params, gpk, sk_i, \zeta, M, \Upsilon)$: *This algorithm takes* $params, gpk, sk_i$*, an attribute set* $\zeta \subseteq \mathcal{A}_i$*, message* M*, and the predicate* Υ *as an input and returns a group signature* σ *on* M*.*

- $0/1 \leftarrow \texttt{Verify}(params, gpk, M, \Upsilon, \sigma)$: *This is a deterministic algorithm verifies the validity of the group signature* σ *against gpk and returns* $1/0$*. If* 1 *then the algorithm claims that the* σ *is a valid group signature, otherwise,* σ *is invalid.*

- $i/\perp \leftarrow \texttt{OpenUser}(params, gpk, ok_{user}, \sigma, M, \Upsilon, r\vec{e}g)$*: This is a deterministic algorithm which takes as input* $params, gpk, ok_{user}, \sigma, \Upsilon, M$ *and* $r\vec{e}g$*, and returns either* $i \geq 1$ *or* $\perp$*. If* i*, the algorithm claims that the group member with identity* i *has produced* σ*, and if* $\perp$*, then no group member produced* σ*.*

- $\zeta/\perp \leftarrow \texttt{TraceAtt}(params, gpk, tk_{att}, \sigma, M, \Upsilon)$*: This is a deterministic algorithm which takes as input* $params, gpk, tk_{att}, \sigma, M$ *and* Υ*, and outputs either the attribute set* $\zeta \subseteq Att$ *or* $\perp$*. Here it claims that* ζ *is the attribute set that is used to satisfy* Υ *in producing* σ*. If* $\perp$*, then the algorithm claims that no attribute set is used to produce* σ*.*

Entities: There are several entities in ABGS scheme:

- The group manager `GM`, also known as *Issuer*, has issuing key ik using which he enrolls a user into the group by allotting some privileges (in terms of attributes) say $\mathcal{A}_i \subseteq Att$ and issuing a user's private key sk_i, by running interactive `Join` algorithm with the user.

- The *opener* has user opening key ok_{user} by which he is able to open the signature and reveal the user identity through `OpenUser` algorithm.

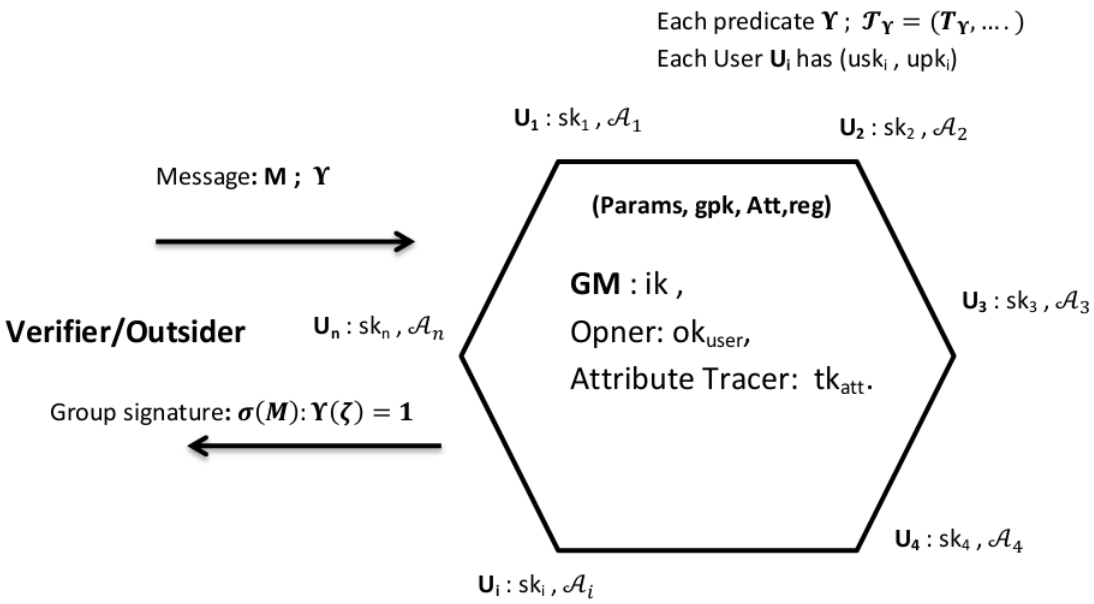


Figure 2.1: Attribute-Based Group Signature

- The *attribute tracer* has the attribute tracing key tk_{att} by which he can trace the attribute set ζ from the group signature, which is used to satisfy the predicate Υ, by running the `TraceAtt` algorithm.
- Group members, or signers, who are having their private keys sk_i. They run `Sign` algorithm to produce a group signature on a document M with predicate Υ; if they possess valid attribute set $\mathcal{A}_i$which satisfies the predicate.
- Outsider or verifier who can seek a group signature for a document M with predicate Υ from group manager GM. He can also verify the group signature using the group public key, gpk.

Note: Normally the `Setup` and `KeyGen` algorithms are run by some trusted party and he will distribute the appropriate keys to the concerned entities.

System model: Intuitively, a user (with a pair of keys (usk_i, upk_i)) engage with the group manager `GM` (with issuing key ik) in `Join` protocol to join the group and get the group member secret key sk_i. An outsider or verifier approaches the group manager `GM` with message M along with the predicate Υ for group signature on M.

A group member who satisfy the predicate Υ will generate the group signature σ on M and verifier verifies the σ using group public key gpk. Later in case of any dispute or upon requirement, the opener reveals the signer's identity from the signature σ by using user opening key ok_{user}. The attribute tracer using attribute tracing key tk_{att} can reveal the set of attributes that the signer has used in generating σ.
Remarks:

- ABGS scheme can be used as group signature scheme with $|Att| = 1$, say $Att = \{att\}$, and allotting att to every member in the group. Thus the predicate contains only one literal att, which can be satisfied by any member in the group.
- ABGS scheme with attribute anonymity can be used as ABS. The ABGS scheme possesses an extra feature of revealing the signer's identity from the signature.
- One can design an ABGS scheme with attribute anonymity by using an ABS scheme and a group signature scheme. The group signature scheme provides a signer tracing feature. Therefore an ABGS scheme can be built by cascading ABS with group signature scheme. A user of ABS is also a group member in the group signature scheme. Each user holds two secret keys viz. attribute keys from the ABS and group private key from the group signature scheme. First the user signs the document associated with a predicate with his attribute keys and then he signs using his group private key. The cost of building this ABGS scheme is the combined cost of both the schemes.

2.5 Provable Security

In the early years after the invention of public key cryptography by Diffie and Hellman in 1976 [49], design and evaluation of public key cryptosystems has been done merely in an ad-hoc manner. But due to various successful attacks on the cryptosystems, the cryptographic community understood that this ad-hoc approach might not be good enough. Moreover, in the early days of public key cryptography, security considerations only dealt with the most basic attacks, i. e., cryptanalytic research

concentrated on inverting the scheme's underlying one-way functions. However, the cryptosystems succumbed to attacks due to different reasons. The paradigm of provable security is an attempt to address this. The goals of provable security are to define appropriate models of security on the one hand, and to develop cryptographic designs that can be proven to be secure within particular models on the other. There are two general approaches for structuring the security proof.

2.5.1 Game-based Approach or Sequence-of-games Approach

This approach is simple to understand and easy to analyze security of any complex cryptosystem. This approach was separately introduced by Shoup [99] and by Bellare et al. [19]. The notion of security for a scheme is defined as a game between an *adversary* and a *challenger*. If the adversary wins the game, the security of the scheme is compromised. Both the adversary and the challenger are modeled as probabilistic processes, so that the whole game is modeled as a probability space. The fact that the game is won by the adversary corresponds to a specific event S and the scheme is secure when $Pr[S]$ is close to some target probability. Usually, providing such a bound given the sole description of the initial game is hard. One thus constructs a sequence of games Game 0, Game 1,. . . , Game n, where Game 0 is the original game between the adversary and the challenger. Just as Game 0 defines an event $S_0 = S$, each Game i defines an event S_i such that $Pr[S_i]$ is negligibly close to $Pr[S_{i-1}]$ for $i = 1, ..., n$. Provided that $\Pr[S_n]$ is easy to compute and negligibly close to the target probability, we are done. The game-based approach is used in several cryptosystems to prove its security, including [24; 39; 46; 82] and many more. We use this approach for proving the security of ABGS scheme in Chapter 6 and Chapter 7.

2.5.2 Simulation-based Approach or Reductionist Approach

In this approach, a cryptosystem is called provably secure if there exists a polynomial reduction from an attack against the security of the cryptosystem to a well-established hard problem (such as the integer factorization problem) . Informally, this means that

if there is a polynomially bounded adversary breaking the scheme, then the problem assumed to be hard can also be solved in polynomial time. As this contradicts our assumption towards hard problem, no such adversary exists and hence the system is secure. Here breaking the scheme is related to the hardness of the complexity assumptions described in the Section 2.3. This technique is used in many cryptosystems to analyse the security, and the first scheme which uses this technique is Rabin's public key cryptosystem [94].

To analyze the security of the cryptographic system some *idealized* models are introduced viz. random oracle model, generic group model and standard model.

2.5.3 Random Oracle Model

In the random oracle model, introduced by Bellare and Rogaway [18], all parties - the legitimate ones as well as the adversary - have black-box access to functions which behave like truly random functions. Under this idealized assumption, it is possible to develop cryptosystems that are both efficient and provably secure. In concrete implementations, however, truly random functions are replaced by concrete objects like cryptographic hash functions. Thus, even a rigorously analyzed security proof in the random oracle model does not guaranty security in the real world.

A *hash function* $\mathcal{H}$ is a keyless algorithm that takes arbitrary-length inputs and outputs a fixed-length hash value, $\mathcal{H} : \{0,1\}^* \rightarrow \{0,1\}^\ell$. A hash function should exhibit several properties, including pre-image resistance (given a random element of the output set, it should be computationally infeasible to find a pre-image of that element) and collision resistance (it should be computationally infeasible to find two elements that have the same hash value), correlation intractability (it should be infeasible to find an input-output pairs that follows some particular relation).

It is widely believed that if the cryptosystem is secure in random oracle model then there is no "structural flaws" in it. And if any attack is possible against the cryptosystem which is proven secure in random oracle model then it must have taken advantage of implementation flaws of the cryptosystem.

2.5.4 Generic Group Model

This model was introduced by Victor Shoup [98]. The idea of the generic group model is to give a precise definition of what it means to have an algorithm that does not make use of any special features of the group.

For simplicity suppose that the multiplicative group $\mathbb{G}$ is cyclic of prime order q. In the generic group model one supposes that instead of formulas for the group operation we have an "oracle" that for any i will give us an "encoding" $\sigma(i)$. In addition, if we have two encodings $\sigma(i)$ and $\sigma(j)$ (but we do not necessarily know i or j), then the oracle will give us $\sigma(i \pm j) = \sigma(i)\sigma(j)^{\pm 1}$. By repeatedly querying the oracle, we can also efficiently determine $\sigma(ri + sj) = \sigma(i)r\sigma(j)s$ for any integers $0 \leq r, s < q$. Without loss of generality we may suppose that we are allowed to ask for the value $\sigma(ri + sj)$ in a single query. Thus, the oracle will tell us either the encoding of an integer $i, 0 \leq i < q$, or else the element $\sigma(i)^r\sigma(j)^s$ for $0 \leq r, s < q$ of our choice. However, the oracle reveals no other information. The way to ensure this is to stipulate that the oracle's encodings are randomly selected elements from some set of bitstrings. The only condition on the oracle's responses is that if the same group element is queried a second time, it must respond with the same encoding [76]. Boneh et al. provide the lower bound on the computational complexity of solving the q-SDH problem in generic group model [27]. Maji et al. have proved the security of their Attribute-Based Signature in generic group model [85].

2.5.5 Standard Model

In standard model, unlike in random oracle model, no cryptographic primitive is replaced by any idealized version. Cramer-Shoup encryption scheme [46] is the first encryption scheme which is proven secure in standard model.

It is observed that the cryptosystems which are proven secure in standard model are less efficient and more complex when compared to the cryptosystems proven secure in random oracle model. Therefore unless we succeed to design a cryptosystem that is secure in standard model and is as efficient as cryptosystem secure in random oracle model, the random oracle model will never cease to disappear. Currently there are many schemes which are proven secure in standard model [53; 82; 92].

2.6 Summary

In this chapter, we listed out the complexity assumptions and described briefly the cryptographic primitives which are used in construction of the proposed schemes. We described two instantiations of Groth-Sahai proof systems. We introduced group signature, ABS and ABGS schemes. We also described the security notions of these schemes. We described different methods to prove the security of a cryptosystem under various models viz. random oracle model, generic group model and standard model. In the next chapter, we propose an ABGS scheme with attribute anonymity in the random oracle model.

Chapter 3

An ABGS Scheme with Attribute Anonymity and Attribute Tracing in the Random Oracle Model

In this chapter, we present an ABGS scheme with attribute anonymity and prove that it is secure under random oracle model with DL, q-SDH, DLDH and XDH assumptions. Moreover, the scheme provides revocation feature which allows to revoke multiple group members at anytime.

3.1 Introduction

Khader in [73] has suggested that the *attribute anonymity* - the verifier should be able to verify whether the signer has required attributes without learning which set of attributes he used for signing, is a desirable feature to achieve. Later Khader proposed an ABGS scheme [72] with member revocation feature without addressing attribute anonymity. We propose first ABGS scheme [2] with attribute anonymity feature. The proposed scheme's signature length is constant and also has membership revocation facility. We use the scheme proposed by Emura et al.'s ABGS scheme [51]

as a base scheme [2]. We prove that the proposed ABGS scheme is secure under random oracle model with DL, q-SDH, DLDH and XDH assumptions. We use the membership certificate format of [48] that makes the scheme *non-frameable*. We have also provided independent opening of the signer's identity and opening of the attribute set identity from the signature, thus these tasks can be assigned to two independent authorities and it is also useful when anyone wants to know the privileges of the signer rather than its identity. It allows group manager to revoke multiple members from the group at anytime. We also given a short ABGS scheme whose signature length is extremely short irrespective of number of attributes.

In Section 3.2, we present the proposed ABGS scheme with attribute anonymity and the related security definitions. The construction of the proposed ABGS scheme is described in Section 3.3. Its security analysis is given in Section 3.4. The short ABGS is given in Section 3.5 followed by comparison with previous schemes in Section 3.6. Finally we summarize in Section 3.7.

3.2 Proposed Scheme

In this section, we propose an ABGS scheme with attribute anonymity. Let $\mathrm{U}_1, \mathrm{U}_2, ..., \mathrm{U}_n$ be the members of a group. Let k be the security parameter, $params$ the system parameters, Att the universal set of attributes, Υ used to denote a predicate, $\Upsilon(\zeta) = 1$ denotes that the attribute set $\zeta \subseteq Att$ satisfies the predicate Υ, gpk the group public key, ik the issuing key used for issuing private keys to the users, ok_{user} the user opening key used to open the user identity of the group signature, tk_{att} the attribute tracing key used to trace the attributes of the group signature, $\mathcal{A}_i \subseteq Att$ the set of attributes assigned to the user U_i, A_i represents the membership certificate of U_i, sk_i denotes the private key for the member U_i implicitly includes A_i and $r\vec{e}g$ be the registration table with the group manager where the current group members membership certificates $\{A_i\}_{i=1}^{n}$ are stored.

Each document M is associated with some attribute relationships satisfying a predicate Υ whose access tree is denoted by T_Υ. The user U_i can make a group

signature on the document M if there exists a set of attributes $\zeta \subseteq \mathcal{A}_i$ with the user such that $\Upsilon(\zeta) = 1$.

Definition 3.2.1 (ABGS) *An ABGS scheme consists of following algorithms. Unless otherwise indicated, algorithms are randomized.*

- $params \leftarrow \texttt{Setup}(1^k)$: *This algorithm takes the security parameter k as an input and returns the system parameter params.*
- $(gpk, ik, ok_{user}, tk_{att}) \leftarrow \texttt{KeyGen}(params)$: *This algorithm takes the system parameter params, and returns a group public key gpk, an issuing key ik, a user opening key ok_{user} and an attribute tracing key tk_{att}.*
- $sk_i \leftarrow \texttt{Join}(\langle params, gpk, ik, upk_i, \mathcal{A}_i\rangle, \langle params, gpk, upk_i, usk_i\rangle)$: *This is an interactive group joining protocol between a user U_i (using his secret key usk_i) and the **GM** (using the issuing key ik and the attributes $\mathcal{A}_i \subseteq Att$ for U_i). In the protocol U_i ends with a member private key sk_i and **GM** ends with an updated registration table $\vec{reg}$.*
- $\sigma \leftarrow \texttt{Sign}(params, gpk, sk_i, \zeta, M, \Upsilon)$: *This algorithm takes $params, gpk, sk_i$, an attribute set $\zeta \subseteq \mathcal{A}_i$, message M, and the predicate Υ as an input and returns a group signature σ on M.*
- $0/1 \leftarrow \texttt{Verify}(params, gpk, M, \Upsilon, \sigma)$: *This is a deterministic algorithm verifies the validity of the group signature σ against gpk and returns $1/0$. If 1 then the algorithm claims that the σ is a valid group signature, otherwise, σ is invalid.*
- $i/\perp \leftarrow \texttt{OpenUser}(params, gpk, ok_{user}, \sigma, M, \Upsilon, \vec{reg})$*: This is a deterministic algorithm which takes as input $params, gpk, ok_{user}, \sigma, \Upsilon, M$ and $\vec{reg}$, and returns either $i \geq 1$ or $\perp$. If i, the algorithm claims that the group member with identity i has produced σ, and if $\perp$, then no group member produced σ.*
- $\zeta/\perp \leftarrow \texttt{TraceAtt}(params, gpk, tk_{att}, \sigma, M, \Upsilon)$*: This is a deterministic algorithm which takes as input $params, gpk, tk_{att}, \sigma, M$ and Υ, and outputs either*

the attribute set $\zeta \subseteq Att$ or $\perp$. Here it claims that ζ is the attribute set that is used to satisfy Υ in producing σ. If $\perp$, then the algorithm claims that no attribute set is used to produce σ.

- Revoke$(params, gpk, ik, i)$: *This algorithm takes $params, gpk, ik$ and i as an input and revokes the group member with identity i.*

Entities: There are several entities in ABGS scheme:

- The group manager `GM`, also known as *issuer*, has issuing key ik using which he enrolls a user into the group by allotting some privileges (in terms of attributes) say $\mathcal{A}_i \subseteq Att$ and issuing a user's private key sk_i, by running interactive `Join` algorithm with the user.
- The *opener* has user opening key ok_{user} by which he is able to open the signature and reveal the user identity through `OpenUser` algorithm.
- The *attribute tracer* has the attribute tracing key tk_{att} by which he can trace the attribute set ζ from the group signature, which is used to satisfy the predicate Υ, by running the `TraceAtt` algorithm.
- Group members or signers who are having their private keys sk_i. They run `Sign` algorithm to produce a group signature on a document M with predicate Υ if they possess valid attribute set $\mathcal{A}_i$ which satisfies the predicate.
- Outsider or verifier who can seek a group signature for a document M with predicate Υ from group manager `GM`. He can also verify the group signature using the group public key, gpk.

Note Normally the `Setup` and `KeyGen` algorithms are run by some trusted party and he will distribute the keys to the concerned entities.

ABGS scheme is correct if each group member produces his signature using his signing key and his attributes.

Definition 3.2.2 (`Correctness`) *We say an ABGS scheme is* correct *if and only if honestly-generated signatures verify correctly. That is,*

$$\mathsf{Verify}(params, gpk, M, \mathsf{Sign}(params, gpk, sk_i, \zeta, M, \Upsilon), \Upsilon) \;\; \rightarrow \;\; 1$$

such that $\mathrm{U}_i \in r\vec{e}g, \zeta \subseteq \mathcal{A}_i$ *and* $\Upsilon(\zeta) = 1$.

In ABGS scheme a group member may have multiple attribute sets to satisfy the predicate and he can produce a group signature using one of them. An ABGS scheme preserves attribute anonymity if it is computationally difficult to identify with what attribute set he produces the signature.

Definition 3.2.3 (`Attribute anonymity`) *We say that the ABGS scheme preserves* attribute anonymity *if for all PPT* $\mathcal{A}$*, the probability that* $\mathcal{A}$ *wins the following game is negligible.*

- `Setup` : *The challenger runs* $\mathtt{KeyGen}(params)$*, and obtains* gpk, ik, ok_{user} *and* tk_{att}*. Challenger gives* $params, gpk, ok_{user}$ and ik to $\mathcal{A}$.
- `Phase1` : $\mathcal{A}$ *can send following queries to the challenger,*
  - `Join` : $\mathcal{A}$ *can request the challenger (to run), the Join procedure for any honest member of her choice.* $\mathcal{A}$ *plays the role of corrupt* ***GM*** *on these queries.*
  - `Signing` : $\mathcal{A}$ *can request a group signature* σ *on any, message* M*, predicate* Υ_i*,* U_i *and a set of attributes* $\zeta_i \subseteq \Upsilon_i$ *such that* $\Upsilon(\zeta_i) = 1$*, of her choice.*
  - `Corruption` : $\mathcal{A}$ *can request the secret key* sk_i *of the members* U_i *of her choice.*
  - `TraceAtt` : $\mathcal{A}$ *can request the attribute set of some valid group signatures* σ.

- `Challenge` : $\mathcal{A}$ *outputs* M^*, Υ^*, *a non-corrupted user* U_i (*which is not queried in* `Corruption` *queries*) *such that there exists two attribute sets* $\zeta_{i_0}, \zeta_{i_1} \subseteq \mathcal{A}_i$ *and* $\Upsilon(\zeta_{i_0}) = 1, \Upsilon(\zeta_{i_1}) = 1$ *holds. Then the challenger uniformly selects* $b \in_R \{0, 1\}$, *uses* ζ_{i_b} *to make a group signature* σ^* *on* M^* *and returns* σ^* *to* $\mathcal{A}$.

- `Phase2` : $\mathcal{A}$ *can make the* `Signing`, `Corruption`, `Join` *and* `TraceAtt` *queries. Note that* `Corruption` *query includes* U_i *and* `TraceAtt` *query does not include* σ^*.

- `Output` : $\mathcal{A}$ *outputs a bit* b', *and wins if* $b' = b$.

The advantage of $\mathcal{A}$ *is defined as* $Adv^{att-anon}(\mathcal{A}) = |Pr(b = b') - \frac{1}{2}|$.

In `Join` queries, $\mathcal{A}$ can play the role of corrupted `GM` (same as `SndToU` oracle in [20]).

ABGS scheme preserves user anonymity if there are at least two group members possessing valid attribute sets and one of them produces the group signature then it should be computationally hard to identify who produced the group signature among them, even if their secret keys are revealed afterwards.

Definition 3.2.4 (`User anonymity`) *We say that the ABGS scheme preserves* user anonymity *if for all PPT* $\mathcal{A}$, *the probability that* $\mathcal{A}$ *wins the following game is negligible.*

- `Setup` : *The challenger runs* `KeyGen`$(params)$, *and obtains* gpk, ik, ok_{user} *and* tk_{att}. *Challenger gives* $params, gpk, tk_{att}$ and ik to $\mathcal{A}$.

- `Phase1` : $\mathcal{A}$ *can issue the* `Signing`, `Corruption`, `Join` *and* `OpenUser` *queries. All queries are the same as in* attribute anonymity *game. Instead of* `TraceAtt` *queries in the previous game,* $\mathcal{A}$ *requests* `OpenUser` *queries as follows,*

  - `OpenUser` : $\mathcal{A}$ *can request the signer's identity of some valid group signatures* σ.

- `Challenge` : $\mathcal{A}$ *outputs* M^*, Υ^*, *and non-corrupted users* $\mathrm{U}_{i_0}, \mathrm{U}_{i_1}$ *and* ζ[1]. *Note that* $\zeta \subseteq \mathcal{A}_{i_0}, \zeta \subseteq \mathcal{A}_{i_1}$ *and* $\Upsilon^*(\zeta) = 1$. *The challenger randomly selects* $b \in_R \{0,1\}$ *and responds with a group signature* σ^* *on* M^* *of group member* U_{i_b}.

- `Phase2` : $\mathcal{A}$ *can make the* `Signing`, `Corruption`, `Join` *and* `OpenUser` *queries. Note that* `Corruption` *query includes both* $\mathrm{U}_{i_0}, \mathrm{U}_{i_1}$ *and* `OpenUser` *query does not include* σ^*.

- `Output` : $\mathcal{A}$ *outputs a bit* b', *and wins if* $b' = b$.

The advantage of $\mathcal{A}$ *is defined as* $Adv^{usr-anon}(\mathcal{A}) = |Pr(b = b') - \frac{1}{2}|$.

Following definitions of `traceability`, `non-frameability` and `collusion resistance of attribute certificates` are similar to the one given in [51].

ABGS scheme preserves traceability if it is possible to trace the valid group signature to its signer with the help of group opening key.

Definition 3.2.5 (`Traceability`) *We say that the ABGS scheme preserves* traceability *if for all PPT* $\mathcal{A}$, *the probability that* $\mathcal{A}$ *wins the following game is negligible.*

- `Setup` : *The challenger runs* `KeyGen`$(params)$, *and obtains* gpk, ik, ok_{user} *and* tk_{att}. *Challenger gives* $params, gpk, ok_{user}$ and tk_{att} *to* $\mathcal{A}$.

- `Queries` : $\mathcal{A}$ *can issue the* `Signing`, `Corruption` *and* `Join` *queries. All queries are the same as in the* attribute anonymity *game, except the* `Join` *queries.*

  - `Join` : *Here* $\mathcal{A}$ *requests the challenger the Join procedure for corrupted member* U_i.

- `Output` : $\mathcal{A}$ *outputs a message* M^*, *and a group signature* σ^*. Υ^* *is the predicate in this phase.*

[1]Here ζ is common to both users since we are only concerned about user anonymity as attribute anonymity is separately considered in *attribute anonymity* definition.

$\mathcal{A}$ *wins if* (1) $\texttt{Verify}(params, gpk, M^*, \sigma^*, \Upsilon^*) = 1$, (2) $\texttt{OpenUser}(params, gpk, ok_{user}, \sigma^*, M^*, \Upsilon^*, r\vec{e}g) = 0$. *The advantage of* $\mathcal{A}$ *is defined as the probability that* $\mathcal{A}$ *wins.*

In `Join` queries, $\mathcal{A}$ can play the role of corrupted user (same as `SndToI` oracle in [20]).

ABGS scheme preserves non-frameability if it is difficult to produce a valid group signature which traces back to a group member who has not produce it, even with the help of group manager's secret key.

Definition 3.2.6 (`Non-frameability`) *We say that the ABGS scheme preserves* non-frameability *if for all PPT* $\mathcal{A}$, *the probability that* $\mathcal{A}$ *wins the following game is negligible.*

- `Setup` : *The challenger runs* $\texttt{KeyGen}(params)$, *and obtains* gpk, ik, ok_{user} *and* tk_{att}. *Challenger gives* $params, gpk, ik, ok_{user}$ and tk_{att} *to* $\mathcal{A}$.
- `Queries` : $\mathcal{A}$ *can issue the* `Signing`, `Corruption` *and* `Join` *queries. All queries are the same as in the* attribute anonymity *game.*
- `Output` : *Finally,* $\mathcal{A}$ *outputs a message* M^*, *an honest member* U_{i^*} *and a group signature* σ^*. Υ^* *is the predicate in this phase.*

$\mathcal{A}$ *wins if* (*1*) $\texttt{Verify}(params, gpk, M^*, \sigma^*, \Upsilon^*) = 1$,
(*2*) $\mathsf{OpenUser}(params, gpk, ok_{user}, \sigma^*, M^*, \Upsilon^*, r\vec{e}g) = i^*$, *a honest member* U_{i^*},
(*3*) $\mathcal{A}$ *has not obtained* σ^* *in* `Signing` *queries on* M^*, U_{i^*} *and with* Υ^*, *and*
(*4*) $\mathcal{A}$ *has not obtained* sk_{i^*} *in* `Corruption` *queries on* U_{i^*}.
The advantage of $\mathcal{A}$ *is defined as the probability that* $\mathcal{A}$ *wins.*

ABGS scheme preserves collusion resistance of attribute certificates if it is computationally hard for group members to collude by pooling their attribute certificates to satisfy the predicate and to produce a valid group signature.

Definition 3.2.7 (`Collusion resistance of attribute certificates`) *We say that the ABGS schemes preserve* collusion resistance of attribute certificates *if for all PPT $\mathcal{A}$, the probability that $\mathcal{A}$ wins the following game is negligible.*

- `Setup` : *The challenger runs* `KeyGen`$(params)$*, and obtains* gpk, ik, ok_{user} *and* tk_{att}*. Challenger gives params* and *gpk to* $\mathcal{A}$*.*
- `Queries` : $\mathcal{A}$ *can issue the* `Signing`, `Corruption` *and* `Join` *queries. All queries are the same as in the* attribute anonymity *game, except the* `Join` *queries.*
  - `Join` : *Here* $\mathcal{A}$ *requests the challenger the Join procedure for corrupted member* U_i*.*
- ***`Output:`*** *Finally,* $\mathcal{A}$ *outputs a message* M^**, and a group signature* σ^**.* Υ^* *is the predicate in this phase.*

$\mathcal{A}$ *wins if (1)* `Verify`$(params, gpk, M^*, \sigma^*, \Upsilon^*) = 1$*, and (2)* $\mathcal{A}$ *has not obtained attribute certificates associated with any* ζ^**, such that* $\Upsilon(\zeta^*) = 1$*, corresponding to a single user.*

This property can be better explained by an example. Let the two users U_{i_0} and U_{i_1} be having attributes $\mathcal{A}_{i_0}$ and $\mathcal{A}_{i_1}$, respectively. We assume that $\zeta^* \nsubseteq \mathcal{A}_{i_0}, \zeta^* \nsubseteq \mathcal{A}_{i_1}$, but $\zeta^* \subseteq \mathcal{A}_{i_0} \cup \mathcal{A}_{i_1}$. Then U_{i_0} and U_{i_1} cannot make a valid group signature with ζ^* even if they both collude with each other.

3.3 Construction

Construction of an ABGS scheme with attribute anonymity having constant size signature is presented in this section. Our construction is based on the [51]'s ABGS scheme. We prove that the proposed construction is secure under random oracle model with DL, q-SDH, DLDH and XDH assumptions. We use the membership certificate format of [48] that makes the scheme non-frameable i.e. even colluded

group manager cannot forge the signature. Thus the proof of traceability and non-frameability are similar to one presented in [48]. For generating the public values of the access tree we use the bottom-up approach technique introduced by [51]. We achieve the attribute anonymity by proving the knowledge of corresponding attribute certificates combinedly instead of separately as in [51]. The proposed ABGS scheme achieves the constant size signature i.e. signature length is independent of the number of attributes involved. We have also provided independent opening of the signer's identity and opening of the attribute set from the signature. Thus these tasks can be assigned to two independent authorities and it is also useful when anyone wants to know the privileges of the signer rather than its identity. We adopt membership revocation mechanism from [51] and make it suitable to our scheme. This allows the group manager to revoke multiple members from the group at anytime. We also give a short ABGS scheme whose signature length is extremely short irrespective of number of attributes. For XDH assumption to hold, we assume that the instantiation of the bilinear groups are done using the Weil or Tate pairing over MNT curves, since in the supersingular curves the DDH problem is known to be easy on all cyclic subgroups [28; 60].

Let `NIZK` be a Non-Interactive Zero-Knowledge proof, `SPK` be the Signature Proof of Knowledge, and `Ext-Commit` be an extractable commitment scheme which uses the Pailler's encryption scheme [93], which is required in the security proof of the traceability. Let $\{T_{i,j}\}_{\mathrm{att}_j \in \mathcal{A}_i}$ the attribute certificates of U_i, it is implicitly include in sk_i.

- `Setup`(1^k):
  - (i) Define the cyclic groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3$ of prime order p, where $|p| = O(k)$, an one-way isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$, a bilinear maps $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$, and a hash function $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_p^*$.
  - (ii) Define the attributes $Att = \{\mathrm{att}_1, \mathrm{att}_2, ..., \mathrm{att}_m\}$, $m \leq |\mathbb{Z}_p^*|$.
  - (iii) Outputs the system parameters, $params = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3, e, \psi, \mathcal{H}, Att)$.
- `KeyGen`$(params)$:

(i) Select the generators $g_1 \in \mathbb{G}_1$, and $g_2 \in \mathbb{G}_2 : g_1 = \psi(g_2)$.

(ii) Selects $\gamma \in_R \mathbb{Z}_p^*$, and computes $w = g_2^{\gamma}$.

(iii) Selects $\mu \in_R \mathbb{Z}_p^*$, and computes $\mathcal{U} = g_2^{\frac{1}{\mu}}$.

(iv) Selects $u_1, u_2 \in_R \mathbb{G}_1, x_1', x_2' \in_R \mathbb{Z}_p^*$, and computes $h_1 = u_1^{x_1'}$ and $h_2 = u_2^{x_2'}$. Also selects a random generator $\hat{h}_3 \in_R \mathbb{G}_2$ and exponents $x_3', y_3' \in_R \mathbb{Z}_p^*$, and set $\hat{u}_3 = \hat{h}_3^{1/x_3'}$ and $\hat{v}_3 = \hat{h}_3^{1/y_3'}$.

(v) For each $\text{att}_j \in Att$, selects $s_j \in_R \mathbb{Z}_p^*$, sets $S = \{s_j\}_{\text{att}_j \in Att}$, and computes $g_{\text{att}_j} = g_2^{s_j} (\forall \text{att}_j \in Att)$.

(vi) Outputs the user opening key, $ok_{user} = x_1'$, the attribute opening key, $tk_{att} = (x_3', y_3')$, the issuing key, $ik = (\gamma, \mu, S)$, and the group public key,

$$gpk = (g_1, g_2, u_1, h_1, u_2, h_2, \hat{u}_3, \hat{v}_3, \hat{h}_3, w, \mathcal{U}, \{g_{\text{att}_j}\}_{\text{att}_j \in Att}).$$

- `BuildTree`$(params, gpk, ik, \Upsilon)$:

  (i) Let T_Υ be the tree that represents the predicate Υ.

  (ii) `GM` runs $T^{ext} \leftarrow$ `AddDummyNode`(T_Υ) and `AssignedValue`(p, S, T^{ext}) and gets $(\{s_{d_j}\}_{d_j \in D_{T_\Upsilon}}, s_T)$.

  (iii) `GM` computes $v_T = g_2^{s_T}$.

  (iv) Outputs $\mathcal{T}_\Upsilon = (\{s_{d_j}\}_{d_j \in D_{T_\Upsilon}}, v_T, T^{ext})$.

  Normally the verifier with his predicate approaches the `GM` for a group signature and `GM` runs `BuildTree` algorithm to generate the public values of the predicate Υ and stores it in a public repository. Then anyone among the group members who are eligible will generate a group signature by using the predicate public value.

- `Join`$(< params, gpk, ik, upk_i, \mathcal{A}_i >, < params, gpk, upk_i, usk_i >)$:
  U_i gets sk_i[1] $= ((A_i, x_i, y_i), \{T_{i,j}\}_{\text{att}_j \in \mathcal{A}_i})$, where (A_i, x_i, y_i) is a membership certificate, $\{T_{i,j}\}_{\text{att}_j \in \mathcal{A}_i}$ is the set of attribute certificates and $\mathcal{A}_i$ is the set of U_i's attributes.

[1]Note that $x_i, T_{i,j}$ values can be stored in a public repository, so the size of $sk_i's$ can be reduced to two elements, i.e. $sk_i = (A_i, y_i)$.

(i) U_i picks $y_i \in_R \mathbb{Z}_p^*$ and computes $c_i = \texttt{Ext-Commit}(y_i)$, $F_i = h_1^{y_i}$ and $\pi_1 = \texttt{NIZK}\{y_i : F_i = h_1^{y_i} \wedge c_i = \texttt{Ext-Commit}(y_i)\}$.

(ii) U_i sends F_i, c_i and π_1 to GM.

(iii) GM checks π_1. If π_1 is not valid, then abort.

(iv) GM selects $x_i \in_R \mathbb{Z}_p^*$ and computes

$$\begin{aligned} A_i &= (g_1 F_i)^{1/(\gamma + x_i)}, \\ B_i &= e(g_1 F_i, g_2)/e(A_i, w), \\ D_i &= e(A_i, g_2), \\ T_{i,j} &= A_i^{s_j \mu} (\forall \mathrm{att}_j \in \mathcal{A}_i); \text{and} \\ \pi_2 &= \texttt{NIZK}\{x_i, \{s_j \mu, s_j\}_{(\mathrm{att}_j \in \mathcal{A}_i)} : B_i = D_i^{x_i} \wedge T_{i,j} = A_i^{s_j \mu} (\forall \mathrm{att}_j \in \mathcal{A}_i) \wedge \\ &\quad g_{\mathrm{att}_j} = g_2^{s_j} (\forall \mathrm{att}_j \in \mathcal{A}_i) \wedge e(T_{i,j}, \mathcal{U}) = e(A_i, g_{\mathrm{att}_j}) (\forall \mathrm{att}_j \in \mathcal{A}_i)\}. \end{aligned}$$

(v) GM sends $A_i, B_i, D_i, \{T_{i,j}\}_{\mathrm{att}_j \in \mathcal{A}_i}$ and π_2 to U_i.

(vi) U_i checks π_2. If π_2 is not valid, then abort.

(vii) U_i signs A_i with signature scheme $DSig$ producing signature $S_{i,A_i} = DSig_{usk_i}(A_i)$ and sends to GM.

(viii) GM verifies S_{i,A_i} with respect to upk_i and A_i. If S_{i,A_i} is valid, then GM sends x_i to U_i and adds (U_i, A_i) to $r\vec{e}g$.

(ix) U_i checks the relation $e(A_i, g_2)^{x_i} e(A_i, w) e(h_1, g_2)^{-y_i} \stackrel{?}{=} e(g_1, g_2)$ to verify whether $A_i^{(x_i+\gamma)} = g_1 h_1^{y_i}$.

GM chooses $s_{m+1} \in \mathbb{Z}_p^*$, and computes $g_{\mathrm{att}_{m+1}} = g_2^{s_{m+1}}$ when a new attribute att_{m+1} is added. Then GM computes $T_{i,m+1} = A_i^{s_{m+1}\mu}$ and $\pi_3 = \texttt{NIZK}\{s_{m+1} : T_{i,m+1} = A_i^{s_{m+1}\mu} \wedge g_{\mathrm{att}_{m+1}} = g_2^{s_{m+1}} \wedge e(T_{i,m+1}, \mathcal{U}) = e(A_i, g_{\mathrm{att}_{m+1}})\}$, sends $T_{i,m+1}$ and π_3 to U_i, and publish $g_{\mathrm{att}_{m+1}}$.

- $\texttt{Sign}(params, gpk, sk_i, \zeta, M, \Upsilon)$:

  A signer U_i signs a message $M \in \{0,1\}^*$ as follows:

(i) Get the public values of Υ, $\mathcal{T}_\Upsilon = (\{s_{d_j}\}_{d_j \in D_{T_\Upsilon}}, v_T, T^{ext})$, from the public repository[1].

(ii) U_i chooses $\zeta \subseteq \mathcal{A}_i$ as an input such that $\Upsilon(\zeta) = 1$.

(iii) U_i runs $\mathtt{MakeSimplifiedTree}(\zeta, T^{ext})$ and gets the corresponding $\Delta_{\mathrm{att}_j}(\forall \mathrm{att}_j \in \zeta)$, and $\Delta_{d_j}(\forall d_j \in D^\zeta_{T_\Upsilon})$.

(iv) Note that $\Sigma_{\mathrm{att}_j \in \zeta} \Delta_{\mathrm{att}_j} s_j + \Sigma_{d_j \in D^\zeta_T} \Delta_{d_j} s_{d_j} = s_T$. Let $s_{T_1} = \Sigma_{\mathrm{att}_j \in \zeta} \Delta_{\mathrm{att}_j} s_j$ and $s_{T_2} = \Sigma_{d_j \in D^\zeta_T} \Delta_{d_j} s_{d_j}$. Thus, $s_{T_1} + s_{T_2} = s_T$.

(v) U_i selects $\alpha_1, \alpha_2, \alpha_3, \beta_3 \in_R \mathbb{Z}^*_p$, and computes $C_1 = A_i h_1^{\alpha_1}$,
$C_2 = u_1^{\alpha_1}, C_3 = \Pi_{\mathrm{att}_j \in \zeta} T_{i,j}^{\Delta_{\mathrm{att}_j}} h_2^{\alpha_2} = A_i^{\mu s_{T_1}} h_2^{\alpha_2}, C_4 = u_2^{\alpha_2}$,
$C_5 = \Pi_{d_j \in D^\zeta_T} g_2^{s_{d_j} \Delta_{d_j}} \hat{h}_3^{\alpha_3+\beta_3} = g_2^{s_{T_2}} \hat{h}_3^{\alpha_3+\beta_3}$,
$C_6 = \hat{u}_3^{\alpha_3}, C_7 = \hat{v}_3^{\beta_3}$.

(vi) U_i sets $\tau = \alpha_1 x_i + y_i, \delta = \alpha_1 s_{T_2}$, and computes

$$V = \mathtt{SPK}\{(\alpha_1, \alpha_2, \alpha_3, x_i, \tau, \delta) : \frac{e(C_1, w)}{e(g_1, g_2)} = \frac{e(h_1, g_2)^\tau e(h_1, w)^{\alpha_1}}{e(C_1, g_2)^{x_i}} \quad (3.1)$$

$$\bigwedge C_2 = u_1^{\alpha_1} \bigwedge C_4 = u_2^{\alpha_2} \bigwedge C_6 = \hat{u}_3^{\alpha_3} \bigwedge C_7 = \hat{v}_3^{\beta_3} \bigwedge \quad (3.2)$$

$$\frac{e(C_3, \mathcal{U}) e(C_1, C_5)}{e(C_1, v_T)} = \frac{e(h_2, \mathcal{U})^{\alpha_2} e(C_1, \hat{h}_3)^{\alpha_3+\beta_3} e(h_1, g_2)^\delta}{e(h_1, v_T)^{\alpha_1}}\}(M) \quad (3.3)$$

(a) U_i chooses blinding values $r_{\alpha_1}, r_{\alpha_2}, r_{\alpha_3}, r_{\beta_3}, r_{x_i}, r_\tau, r_\delta \in_R \mathbb{Z}^*_p$.

(b) U_i computes $R_1 = \frac{e(h_1, g_2)^{r_\tau} e(h_1, w)^{r_{\alpha_1}}}{e(C_1, g_2)^{r_{x_i}}}$,
$R_2 = u_1^{r_{\alpha_1}}, R_3 = u_2^{r_{\alpha_2}}, R_4 = \hat{u}_3^{r_{\alpha_3}}, R_5 = \hat{v}_3^{r_{\beta_3}}$,
$R_{Att} = \frac{e(h_2, \mathcal{U})^{r_{\alpha_2}} e(C_1, \hat{h}_3)^{r_{\alpha_3} + r_{\beta_3}} e(h_1, g_2)^{r_\delta}}{e(h_1, v_T)^{r_{\alpha_1}}}$.

(c) U_i computes $c = \mathcal{H}(gpk, M, \{C_i\}_{i=1}^7, \{R_i\}_{i=1}^5, R_{Att})$.

(d) U_i computes $s_{\alpha_1} = r_{\alpha_1} + c\alpha_1, s_{\alpha_2} = r_{\alpha_2} + c\alpha_2, s_{\alpha_3} = r_{\alpha_3} + c\alpha_3, s_{\beta_3} = r_{\beta_3} + c\beta_3, s_\delta = r_\delta + c\delta, s_{x_i} = r_{x_i} + cx_i, s_\tau = r_\tau + c\tau$,.
Thus, $V = (c, s_{\alpha_1}, s_{\alpha_2}, s_{\alpha_3}, s_{\beta_3}, s_{x_i}, s_\tau, s_\delta)$.

(vii) Outputs $\sigma = (\{C_i\}_{i=1}^7, V) \in \mathbb{G}_1^4 \times \mathbb{G}_2^3 \times \mathbb{Z}_p^{*8}$.

[1] `GM` runs `BuildTree` algorithm to generate the public values of the predicate Υ and stores it in a public repository. Note that if the public values of the required predicate is present in the public repository then the user will not approach `GM`.

A signer U_i proves the knowledge of $(\alpha_1, \alpha_2, \alpha_3, \beta_3, x_i, \tau, \delta)$ which satisfies the above 6 relations 3.1 - 3.3 described in `SPK`. The first relation 3.1 captures whether a signer has a valid membership certificate issued by the `Join` algorithm or not. The last relation 3.3 captures whether a signer has valid attribute certificates or not. Note that the signature includes the 2 modules of ElGamal encryption scheme (in encrypting membership certificate - (C_1, C_2) and attribute certificates - (C_3, C_4)) and one module of Linear Encryption (`LE`) scheme [28] (in encrypting dummy nodes - (C_5, C_6, C_7)). Note that the first and the last relations collectively proves that the first 2 ElGamal encryption modules include the same membership certificate and last 2 ElGamal and `LE` encryption module encrypts the related plaintext which makes the scheme `CCA2` secure under random oracle model [56] and it helps to achieve full anonymity property. Also note that the signature is independent of number of attributes, thus its length is constant.

- `Verify`$(params, gpk, M, \sigma, \Upsilon)$:
  A verifier verifies the group signature σ as follows,

  (i) The verifier computes $\widetilde{R_1} = \frac{e(h_1,g_2)^{s_\tau}\, e(h_1,w)^{s_{\alpha_1}}}{e(C_1,g_2)^{s_{x_i}}} \left(\frac{e(g_1,g_2)}{e(C_1,w)}\right)^c$,
  $\widetilde{R_2} = u_1^{s_{\alpha_1}} \left(\frac{1}{C_2}\right)^c$, $\widetilde{R_3} = u_2^{s_{\alpha_2}} \left(\frac{1}{C_4}\right)^c$,
  $\widetilde{R_4} = \hat{u}_3^{s_{\alpha_3}} \left(\frac{1}{C_6}\right)^c$, $\widetilde{R_5} = \hat{v}_3^{s_{\beta_3}} \left(\frac{1}{C_7}\right)^c$,
  $\widetilde{R_{Att}} = \left(\frac{e(h_2,\mathcal{U})^{s_{\alpha_2}}\, e(C_1,\hat{h}_3)^{s_{\alpha_3+\beta_3}}\, e(h_1,g_2)^{s_\delta}}{e(h_1,v_T)^{s_{\alpha_1}}}\right) \left(\frac{e(C_1,v_T)}{e(C_3,\mathcal{U})e(C_1,C_5)}\right)^c$.

  (ii) The verifier checks whether
  $c \stackrel{?}{=} \mathcal{H}(gpk, M, \{C_i\}_{i=1}^7, \{\widetilde{R_i}\}_{i=1}^5, \widetilde{R_{Att}})$.

- `OpenUser`$(params, gpk, ok_{user}, \sigma, M, \Upsilon, r\vec{e}g)$:

  (i) `GM` verifies the validity of σ by using `Verify`$(param, gpk, M, \sigma, \Upsilon)$. If σ is not a valid signature, then `GM` outputs $\perp$.

  (ii) `GM` computes $A_i = \frac{C_1}{C_2^{x_1}}$.

  (iii) `GM` searches A_i in $r\vec{e}g$, and outputs identity i. If there is no entry in $r\vec{e}g$, then `GM` outputs 0.

- $\texttt{TraceAtt}(params, gpk, tk_{att}, \sigma, M, \Upsilon)$:

  (i) GM verifies the validity of σ by using $\texttt{Verify}(param, gpk, M, \sigma, \Upsilon)$. If σ is not a valid signature, then GM outputs $\perp$.

  (ii) GM computes $\widehat{g} = \frac{C_5}{{C_6}^{x'_3} {C_7}^{y'_3}}$.

  (iii) For all $\zeta_k : \Upsilon(\zeta_k) = 1$, GM checks $\widehat{g} \stackrel{?}{=} {g_2}^{s^k_{T_2}}$, where $s^k_{T_2} = \Sigma_{d_j \in D^{\zeta_k}_T} \Delta_{d_j} s_{d_j}$. If any such ζ_k exists then GM outputs it else outputs ϕ.

- $\texttt{Revoke}(params, gpk, ik, \{k_j\}_{j=1}^r)$:
  Here GM revokes the users, $\{\mathrm{U}_{k_1}, ..., \mathrm{U}_{k_r}\}$. First we show how GM revokes a single user U_k:

  (i) GM computes $\tilde{g}_2 = g_2^{\frac{1}{\gamma + x_k}}, \tilde{g}_1 = \psi(\tilde{g}_2), \tilde{u}_1 = u_1^{\frac{1}{\gamma + x_k}}, \tilde{h_1} = h_1^{\frac{1}{\gamma + x_k}}, \tilde{w} = \tilde{g}_2^{\gamma}, \tilde{\mathcal{U}} = \tilde{g}_2^{\frac{1}{\mu}}$ and $\tilde{g}_{\mathrm{att}_j} = \tilde{g}_2^{s_j} (\forall \mathrm{att}_j \in Att)$.

  (ii) GM sets new group public key, $\widetilde{gpk} = (\tilde{g}_1, \tilde{g}_2, \tilde{u}_1, \tilde{h}_1, u_2, h_2, \hat{u}_3, \hat{v}_3, \hat{h}_3, \tilde{w}, \tilde{\mathcal{U}}, \{\tilde{g}_{\mathrm{att}_j}\}_{\mathrm{att}_j \in Att})$ and adds it to the public repository.

  (iii) GM also computes $\tilde{\tilde{g}}_{\mathrm{att}_j} = \tilde{g}_2^{\mu s_j} (\forall \mathrm{att}_j \in Att)$ and $\tilde{\tilde{h}}_1 = \tilde{h_1}^{\mu}$.

  (iv) GM set auxiliary values, $Aux = (A_k, x_k, \tilde{g}_1, \tilde{h_1}, \tilde{\tilde{h}}_1, \{\tilde{\tilde{g}}_{\mathrm{att}_j}\}_{\mathrm{att}_j \in Att})$ and send it to all group members.

  (v) GM outputs $\widetilde{gpk}, Aux$.

  To revoke multiple users at a time, GM computes $\tilde{g}_2 = g_2^{\Pi_{j=1}^r \frac{1}{\gamma + x_{k_j}}}$, i.e. first compute the exponent value then perform exponentiation operation, and similarly other values, where $\{k_j\}_{j=1}^r$ are the user ids to be revoked.

- $\texttt{Update}(\widetilde{gpk}, Aux, sk_i)$:
  Unrevoked user U_i updates his member certificate and attribute certificates, when user U_k is revoked, as follows:

  (i)

$$\tilde{A}_i \;=\; \left(\frac{\tilde{g}_1 \tilde{h_1}^{y_i}}{A_i} \right)^{\frac{1}{x_i - x_k}}$$

$$
\begin{aligned}
&= \left(\frac{g_1^{\frac{1}{\gamma+x_k}} h_1^{\frac{1}{\gamma+x_k} y_i}}{(g_1 h_1^{y_i})^{\frac{1}{\gamma+x_i}}} \right)^{\frac{1}{x_i-x_k}} \\
&= \left(g_1^{\frac{1}{\gamma+x_k}-\frac{1}{\gamma+x_i}} h_1^{\left(\frac{1}{\gamma+x_k}-\frac{1}{\gamma+x_i}\right) y_i} \right)^{\frac{1}{x_i-x_k}} \\
&= \left(g_1^{\frac{x_i-x_k}{(\gamma+x_k)(\gamma+x_i)}} h_1^{\frac{x_i-x_k}{(\gamma+x_k)(\gamma+x_i)}} \right)^{\frac{1}{x_i-x_k}} \\
&= (\tilde{g}_1 \tilde{h_1}^{y_i})^{\frac{1}{\gamma+x_i}}.
\end{aligned}
$$

This is a valid member certificate.

(ii) Similarly for each $att_j \in \mathcal{A}_i$ compute $\tilde{T}_{i,j} = \left(\frac{\tilde{\tilde{g}}_1 \tilde{h_1}^{y_i}}{T_{i,j}} \right)^{\frac{1}{x_i-x_k}}$
$= (\tilde{g}_1 \tilde{h_1}^{y_i})^{\frac{s_j \mu}{\gamma+x_i}}$, a valid attribute certificate.

(iii) Outputs $\widetilde{sk}_i = (\tilde{A}_i, x_i, y_i, \{\tilde{T}_{i,j}\}_{att_j \in \mathcal{A}_i})$.

This revocation technique is adopted from [48] and made it suitable to the proposed scheme. If multiple users have been revoked, say revoked user ids are $\{k_j\}_{j=1}^{r}$, then user U_i repeats this process r times each with x_{k_j}, for $j \in [1, r]$. Thus this scheme supports concurrent join and revocation, which makes it more suitable for dynamic groups. In this revocation mechanism the user is revoked by the publishing the value Aux and consequently all the unrevoked users need to get these updates to update their secrets and verifiers need to use the updated group public key to verify the signature from there on.

3.4 Security Analysis

In this section, we show that our scheme satisfies attribute anonymity, user anonymity, traceability, non-frameability and collision resistance of attribute certificates. Let p, q_H and q_S be the order of bilinear groups, the number of hash queries and the number of signature queries, respectively.

Theorem 3.4.1 *The proposed ABGS scheme is correct.*

Proof SPK ensures that the scheme is correct. For this, we need to show that $R_i = \widetilde{R}_i$, for $i = \{1, ..., 5\}$ and $R_{Att} = \widetilde{R_{Att}}$. If all these equalities hold then
$\mathcal{H}(gpk, M, \{C_i\}_{i=1}^{7}, \{R_i\}_{i=1}^{5}, R_{Att})$
$= \mathcal{H}(gpk, M, \{C_i\}_{i=1}^{7}, \{\widetilde{R}_i\}_{i=1}^{5}, \widetilde{R_{Att}})$ holds and signature should be correctly verified.

$$
\begin{aligned}
\widetilde{R_1} &= \frac{e(h_1, g_2)^{s_\tau} e(h_1, w)^{s_{\alpha_1}}}{e(C_1, g_2)^{s_{x_i}}} \left(\frac{e(g_1, g_2)}{e(C_1, w)}\right)^c \\
&= \frac{e(h_1, g_2)^{r_\tau} e(h_1, w)^{r_{\alpha_1}}}{e(C_1, g_2)^{r_{x_i}}} \left(\frac{e(h_1, g_2)^{\tau} e(h_1, w)^{\alpha_1}}{e(C_1, g_2)^{x_i}} \frac{e(g_1, g_2)}{e(C_1, w)}\right)^c \\
&= R_1 \left(\frac{e(h_1, g_2)^{\alpha_1 x_i + y_i} e(h_1, g_2^{\gamma})^{\alpha_1}}{e(A_i h_1^{\alpha_1}, g_2)^{x_i}} \frac{e(g_1, g_2)}{e(C_1, w)}\right)^c \\
&= R_1 \left(\frac{e(h_1, g_2)^{\alpha_1 x_i + y_i + \gamma\alpha_1}}{e((g_1 h_1^{y_i})^{\frac{1}{x_i+\gamma}}, g_2)^{x_i} e(h_1, g_2)^{\alpha_1 x_i}} \frac{e(g_1, g_2)}{e(C_1, w)}\right)^c \\
&= R_1 \left(\frac{e(h_1, g_2)^{\alpha_1 x_i + y_i + \gamma\alpha_1 - \alpha_1 x_i}}{e(g_1, g_2)^{\frac{x_i}{x_i+\gamma}} e(h_1, g_2)^{\frac{y_i x_i}{x_i+\gamma}}} \frac{e(g_1, g_2)}{e(C_1, w)}\right)^c \\
&= R_1 \left(e(h_1, g_2)^{\gamma\alpha_1 + y_i - \frac{y_i x_i}{x_i+\gamma}} \frac{e(g_1, g_2)^{1-\frac{x_i}{x_i+\gamma}}}{e(C_1, w)}\right)^c \\
&= R_1 \left(\frac{e(h_1, g_2)^{\gamma\alpha_1 + \frac{y_i\gamma}{x_i+\gamma}} e(g_1, g_2)^{\frac{\gamma}{x_i+\gamma}}}{e(C_1, w)}\right)^c \\
&= R_1 \left(\frac{e(h_1, w)^{\alpha_1 + \frac{y_i}{x_i+\gamma}} e(g_1, w)^{\frac{1}{x_i+\gamma}}}{e(C_1, w)}\right)^c \\
&= R_1 \left(\frac{e(h_1, w)^{\alpha_1} e(h_1^{y_i}, w)^{\frac{1}{x_i+\gamma}} e(g_1, w)^{\frac{1}{x_i+\gamma}}}{e(C_1, w)}\right)^c \\
&= R_1 \left(\frac{e(h_1^{\alpha_1}, w) e(A_i, w)}{e(C_1, w)}\right)^c \\
&= \frac{e(h_1, g_2)^{r_\tau} e(h_1, w)^{r_{\alpha_1}}}{e(C_1, g_2)^{r_{x_i}}} (1)^c \\
&= R_1,
\end{aligned}
$$

$$\widetilde{R_2} = u_1^{s_{\alpha_1}} \left(\tfrac{1}{C_2}\right)^c = u_1^{r_{\alpha_1}} \left(u_1^{\alpha_1} \tfrac{1}{C_2}\right)^c = R_2,$$

$$\widetilde{R_3} = u_2^{s_{\alpha_2}} \left(\frac{1}{C_4}\right)^c = u_2^{r_{\alpha_2}} \left(u_2^{\alpha_2} \frac{1}{C_4}\right)^c = R_3,$$

$$\widetilde{R_4} = \hat{u}_3^{s_{\alpha_3}} \left(\frac{1}{C_6}\right)^c = \hat{u}_3^{r_{\alpha_3}} \left(\hat{u}_3^{\alpha_3} \frac{1}{C_6}\right)^c = R_4,$$

$$\widetilde{R_5} = \hat{v}_3^{s_{\beta_3}} \left(\frac{1}{C_7}\right)^c = \hat{v}_3^{r_{\beta_3}} \left(\hat{v}_3^{\beta_3} \frac{1}{C_7}\right)^c = R_5,$$

$$\begin{aligned}
\widetilde{R_{Att}} &= \left(\frac{e(h_2,\mathcal{U})^{s_{\alpha_2}} e(C_1,\hat{h}_3)^{s_{\alpha_3}+s_{\beta_3}} e(h_1,g_2)^{s_\delta}}{e(h_1,v_T)^{s_{\alpha_1}}}\right)\left(\frac{e(C_1,v_T)}{e(C_3,\mathcal{U})e(C_1,C_5)}\right)^c \\
&= \left(\frac{e(h_2,\mathcal{U})^{r_{\alpha_2}} e(C_1,\hat{h}_3)^{r_{\alpha_3}+r_{\beta_3}} e(h_1,g_2)^{r_\delta}}{e(h_1,v_T)^{r_{\alpha_1}}}\right) \\
&\quad \left(\frac{e(h_2,\mathcal{U})^{\alpha_2} e(C_1,\hat{h}_3)^{\alpha_3+\beta_3} e(h_1,g_2)^{\delta}}{e(h_1,v_T)^{\alpha_1}}\right)^c \left(\frac{e(C_1,v_T)}{e(C_3,\mathcal{U})e(C_1,C_5)}\right)^c \\
&= R_{Att}\left(\frac{e(h_2,\mathcal{U})^{\alpha_2} e(C_1,\hat{h}_3)^{\alpha_3+\beta_3} e(h_1,g_2)^{\delta}}{e(h_1,v_T)^{\alpha_1}} \frac{e(C_1,v_T)}{e(A_i^{\mu s_{T_1}} h_2^{\alpha_2},\mathcal{U})e(C_1,g_2^{s_{T_2}}\hat{h}_3^{\alpha_3+\beta_3})}\right)^c \\
&= R_{Att}\left(\frac{e(h_2,\mathcal{U})^{\alpha_2} e(C_1,\hat{h}_3)^{\alpha_3+\beta_3} e(h_1,g_2)^{\delta} \times e(C_1,v_T)}{e(h_1,v_T)^{\alpha_1} \times e(A_i^{\mu s_{T_1}},g_2^{1/\mu}) e(h_2^{\alpha_2},\mathcal{U}) e(C_1,g_2^{s_{T_2}}) e(C_1,\hat{h}_3^{\alpha_3+\beta_3})}\right)^c \\
&= R_{Att}\left(\frac{e(h_1,g_2)^{\delta}}{e(h_1,v_T)^{\alpha_1}} \frac{e(C_1,v_T)}{e(A_i^{s_{T_1}},g_2) e(A_i h_1^{\alpha_1}, g_2^{s_{T_2}})}\right)^c \\
&= R_{Att}\left(\frac{e(h_1,g_2)^{\delta} e(C_1,v_T)}{e(h_1^{\alpha_1},v_T) e(A_i^{s_{T_1}},g_2) e(A_i,g_2^{s_{T_2}}) e(h_1^{\alpha_1},g_2^{s_{T_2}})}\right)^c \\
&= R_{Att}\left(\frac{e(h_1,g_2)^{\delta} e(C_1,v_T)}{e(h_1^{\alpha_1},v_T) e(A_i,g_2^{s_{T_1}+s_{T_2}}) e(h_1,g_2)^{\delta}}\right)^c \\
&= R_{Att}\left(\frac{e(h_1,g_2)^{\delta} e(C_1,v_T)}{e(A_i h_1^{\alpha_1},v_T)}\right)^c \\
&= R_{Att}(1)^c = R_{Att}.
\end{aligned}$$

□

Theorem 3.4.2 *The proposed ABGS scheme satisfies the attribute anonymity with CCA2 secure in the random oracle model under DLDH and DDH assumption.*

Proof The following Lemma implies the Theorem 3.4.2. □

Lemma 3.4.3 *Suppose an adversary $\mathcal{A}$ breaks the attribute anonymity of the proposed scheme with the advantage ϵ. Then, we can construct an algorithm $\mathcal{B}$ that breaks the DLDH assumption on $\mathbb{G}_2$ with the advantage $\frac{1}{2}\left(\frac{\epsilon}{4} - \frac{q_S+q_H}{p}\right)$ and breaks the XDH assumption (namely DDH assumption over $\mathbb{G}_1$) with the advantage $\frac{1}{2}\left(\frac{\epsilon}{4} - \frac{q_S+q_H}{p}\right)$.*

Proof The input of $\mathcal{B}$ is a tuple $(\tilde{u}, \tilde{v}, \tilde{h}, \tilde{U} = \tilde{u}^a, \tilde{V} = \tilde{v}^b, \tilde{H} = \tilde{h}^c) \in \mathbb{G}_2^6$, where $a, b, c \in \mathbb{Z}_p^*$ and either $c = a + b$ (DLDH tuple) or $c \in_R \mathbb{Z}_p^*$(random tuple) and $(u, h, U = u^{a'}, H = h^{b'}) \in \mathbb{G}_1^4$, where $a', b' \in \mathbb{Z}_p^*$ and either $b' = a'$(DDH tuple) or b' is random (random tuple). From such a tuple, using the classical random self-reducibility, one can derive many independent tuples: $(\tilde{u}, \tilde{v}, \tilde{h}, \tilde{U}_i = \tilde{U}^{a_i}\tilde{u}^{b_i}, \tilde{V}_i = \tilde{V}^{a_i}\tilde{v}^{c_i}, \tilde{H}_i = \tilde{H}^{a_i}\tilde{h}^{b_i}\tilde{h}^{c_i})$, where $a_i, b_i, c_i \in_R \mathbb{Z}_p^*$ and $(u, h, U_i = U^{a'_i}u^{b'_i}, H_i = H^{a'_i}h^{b'_i})$, where $a'_i, b'_i \in_R \mathbb{Z}_p^*$. Let the challenged group signature be denoted by $\sigma^* = (\{C_i^*\}_{i=1}^7, V^*)$.

Setup. Algorithm $\mathcal{B}$ simulates the proposed ABGS scheme as follows:

(i) $\mathcal{B}$ generates system parameters, $params = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3, e, \psi, \mathcal{H}, Att)$.

(ii) $\mathcal{B}$ selects $d \in_R \{0, 1\}$. The value of d decides which type of adversary $\mathcal{A}$ is.

(iii) If $d = 0$, $\mathcal{B}$ sets $u_2 = u, h_2 = h$, thus x'_2 is unknown for $\mathcal{B}$, and selects $x'_3, y'_3 \in_R \mathbb{Z}_p^*, \hat{h}_3 \in_R \mathbb{G}_2$ and computes $\hat{u}_3 = \hat{h}_3^{1/x'_3}$ and $\hat{v}_3 = \hat{h}_3^{1/y'_3}$.
If $d = 1$, $\mathcal{B}$ sets $\hat{u}_3 = \tilde{u}, \hat{v}_3 = \tilde{v}, \hat{h}_3 = \tilde{h}$, thus (x'_3, y'_3) is unknown for $\mathcal{B}$, and selects $x'_2 \in_R \mathbb{Z}_p^*, u_2 \in_R \mathbb{G}_1$ and sets $h_2 = u_2^{x'_2}$.

(iv) $\mathcal{B}$ selects $\gamma, \mu, x'_1, \{s_j\}_{\text{att}_j \in Att} \in_R \mathbb{Z}_p^*$, $g_2 \in_R \mathbb{G}_2$ and computes $g_1 = \psi(g_2), w = g_2^{\gamma}, \mathcal{U} = g_2^{\frac{1}{\mu}}, g_{\text{att}_1} = g_2^{s_1}, ..., g_{\text{att}_m} = g_2^{s_m}$, where $m = |Att|$, selects $u_1 \in_R \mathbb{G}_1$ and computes $h_1 = u_1^{x'_1}$.

(v) $\mathcal{B}$ sets a group public key, $gpk = (g_1, g_2, u_1, h_1, u_2, h_2, \hat{u}_3, \hat{v}_3, \hat{h}_3, w, \mathcal{U}, \{g_{\text{att}_j}\}_{\text{att}_j \in Att})$, user opening key, $ok_{user} = x'_1$, and an issuing key, $ik = (\gamma, \mu, S)$.

(vi) $\mathcal{B}$ gives gpk, ik, ok_{user} to $\mathcal{A}$.

Hash queries. At any time, $\mathcal{A}$ can query the hash function $\mathcal{H}$. $\mathcal{B}$ responds with random values with consistency.

Phase 1. $\mathcal{A}$ requests the queries as given in `attribute anonymity` game. $\mathcal{B}$ answers to these queries as the real settings of ABGS scheme, since $\mathcal{B}$ knows all the values. For $d = 1$, (x'_3, y'_3) is unknown, $\mathcal{B}$ uses the following procedure to answer the `TraceAtt` query on any given σ; $\mathcal{B}$ computes A_i by following the procedure in `OpenUser` algorithm and finds $A_i^{\mu s_{T_1}} = \frac{C_3}{C_4^{x'_2}}$. For all $\zeta_k : \Upsilon(\zeta_k) = 1, \zeta_k \subseteq \mathcal{A}_i$, $\mathcal{B}$ gets $\widehat{g}_k = \Pi_{d_j \in D_T^{\zeta_k}} g_{d_j}^{\Delta_{d_j}}$ and checks $e(A_i^{\mu s_{T_1}}, \mathcal{U}) \stackrel{?}{=} e(A_i, v_T/\widehat{g}_k)$. If any such ζ_k exists then $\mathcal{B}$ outputs it, else outputs ϕ. This makes the scheme `CCA2` secure, since opening oracle `TraceAtt` is provided to $\mathcal{A}$.

Challenge. $\mathcal{A}$ outputs M^*, Υ^* and non-corrupted user U_i such that there exists two attribute sets $\zeta_0, \zeta_1 \subseteq \mathcal{A}_i$ which satisfies the predicate Υ^*, i.e. $\Upsilon^*(\zeta_0) = 1$ and $\Upsilon^*(\zeta_1) = 1$ holds to be challenged. $\mathcal{T}_{\Upsilon^*}$ is a public value of the predicate Υ^* and $v_{T^*} = g_2^{s_{T^*}}$. $\mathcal{B}$ picks $\rho \in_R \{0, 1\}$. $\mathcal{B}$ tries to simulate the challenged signature $\sigma^* \longleftarrow \mathtt{Sign}(params, gpk, sk_i, \zeta_\rho, M^*, \Upsilon^*)$ from U_i as follows,

- the encryption: according to d, by choosing an additional random bit d'.
  - if $d = 0$, $C_3^* = \Pi_{\mathrm{att}_j \in \zeta_\rho} T_{i,j}^{\Delta_{\mathrm{att}_j}} H_i = A_i^{\mu s_{T_1}} H_i, C_4^* = U_i$, $C_5^* = \Pi_{d_j \in D_T^{\zeta_{d'}}} g_2^{\Delta_{d_j} s_{d_j}} \hat{h}_3^{\alpha_3 + \beta_3} = g_2^{s_{T_2}} \hat{h}_3^{\alpha_3 + \beta_3}, C_6^* = \hat{u}_3^{\alpha_3}$ and $C_7^* = \hat{v}_3^{\beta_3}$, for a random α_3, β_3, and the other values are computed as the real settings;
  - if $d = 1$, $C_3^* = \Pi_{\mathrm{att}_j \in \zeta_{d'}} T_{i,j}^{\Delta_{\mathrm{att}_j}} h_2^{\alpha_2} = A_i^{\mu s_{T_1}} h_2^{\alpha_2}, C_4^* = u_1^{\alpha_2}$, $C_5^* = \Pi_{d_j \in D_T^{\zeta_{d'}}} g_2^{\Delta_{d_j} s_{d_j}} \tilde{H}_i = g_2^{s_{T_2}} \tilde{H}_i, C_6^* = \tilde{U}_i$ and $C_7^* = \tilde{V}_i$, for a random α_2 and the other values are computed as the real settings.
- the proof of validity `SPK` V^* is simulated by selecting a random $c^*, s^*_{\alpha_1}, s^*_{\alpha_2}, s^*_{\alpha_3}, s^*_{\beta_3}, s^*_{x_i}, s^*_\tau, s^*_\delta \in_R \mathbb{Z}_p^*$ and computing the corresponding $(\{R_i^*\}_{i=1}^5, R^*_{Att})$ by following the procedure given in `Verify` algorithm and patching the hash oracle at $\mathcal{H}(gpk, M^*, \{C_i^*\}_{i=1}^7, \{R_i^*\}_{i=1}^5, R^*_{Att})$ to c^*. If this backpatch fails then $\mathcal{B}$ outputs a random bit and aborts. This probability of failure is $(q_S + q_H)/p$.

In case of failure, $\mathcal{B}$ outputs a random bit and aborts, otherwise, $(\{C_i^*\}_{i=1}^7, V^*)$ is the signature on M^* given back to $\mathcal{A}$.

Phase 2. $\mathcal{A}$ requests the queries as given in `attribute anonymity` game and $\mathcal{B}$ answers it similar to phase 1.

Output. $\mathcal{A}$ outputs its guess $\rho' \in \{0,1\}$ with advantage ϵ. Our algorithm $\mathcal{B}$ outputs 0 if $\rho = \rho'$ (for $d = 0$ indicating that $H = h^{a'}$, DDH tuple, for $d = 1$ indicating that $\tilde{H} = \tilde{h}^{a+b}$, DLDH tuple); otherwise $\mathcal{B}$ outputs 1 (indicating that is a random tuple). If it is a DLDH tuple, and $d' = \rho$, then for $d = \{1\}$ the ElGamal encryption component(C_3^*, C_4^*) as well as LE encryption component(C_5^*, C_6^*, C_7^*) always uses ζ_ρ for encryption and this is a valid signature (the advantage of $\mathcal{A}$ in guessing ρ is ϵ). However, if $d' \neq \rho$, both ζ's are encrypted, $\mathcal{A}$ has thus no advantage in guessing ρ. If this is a random tuple then the signature is independent of ρ and then the adversary's advantage is 0. As a consequence, it has an advantage $\epsilon/4$ in distinguishing DLDH tuples.
Similarly, if it is a DDH tuple, for $d = 0$, it has an advantage $\epsilon/4$ in distinguishing DDH tuples.
Therefore the advantage of $\mathcal{B}$ in breaking DLDH assumption when there is no failure and when $\mathcal{B}$ guesses correctly which type of adversary $\mathcal{A}$ is, is at least $\frac{1}{2}\left(\frac{\epsilon}{4} - \frac{q_S+q_H}{p}\right)$. And the advantage of $\mathcal{B}$ in breaking DDH assumption when there is no failure and when $\mathcal{B}$ guesses correctly which type of adversary $\mathcal{A}$ is, is at least $\frac{1}{2}\left(\frac{\epsilon}{4} - \frac{q_S+q_H}{p}\right)$. □

Theorem 3.4.4 *The proposed ABGS scheme satisfies the user anonymity with CCA2 secure in the random oracle model under DDH assumption.*

Proof The following Lemma implies the Theorem 3.4.4. □

Lemma 3.4.5 *Suppose an adversary $\mathcal{A}$ breaks the user anonymity of the proposed scheme with the advantage ϵ. Then, we can construct an algorithm $\mathcal{B}$ that breaks the XDH assumption (namely DDH assumption over $\mathbb{G}_1$) with the advantage $\frac{\epsilon}{4} - \frac{q_S+q_H}{p}$.*

Proof The input of $\mathcal{B}$ is a tuple $(u, h, U = u^{a'}, H = h^{b'}) \in \mathbb{G}_1^4$, where $a', b' \in \mathbb{Z}_p^*$ and either $b' = a'$(DDH tuple) or b' is random (random tuple). From such a tuple, using the classical random self-reducibility, one can derive many independent tuples: $(u, h, U_i = U^{a'_i} u^{b'_i}, H_i = H^{a'_i} h^{b'_i})$, where $a'_i, b'_i \in_R \mathbb{Z}_p^*$. Let the challenged group signature be denoted by $\sigma^* = (\{C_i^*\}_{i=1}^7, V^*)$.

Setup. Algorithm $\mathcal{B}$ simulates the proposed ABGS scheme as follows,

(i) $\mathcal{B}$ generates system parameters, $params = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3, e, \psi, \mathcal{H}, Att)$.

(ii) $\mathcal{B}$ selects $d \in_R \{0, 1\}$. The value of d decides which type of adversary $\mathcal{A}$ is.

(iii) If $d = 0$, $\mathcal{B}$ sets $u_1 = u, h_1 = h$, thus x'_1 is unknown for $\mathcal{B}$, and selects $x'_2 \in_R \mathbb{Z}_p^*, u_2 \in_R \mathbb{G}_1$ and computes $h_2 = u_2^{x'_2}$.
If $d = 1$, $\mathcal{B}$sets $u_2 = u, h_2 = h$, thus x'_2 is unknown for $\mathcal{B}$, and selects $x'_1 \in_R \mathbb{Z}_p^*, u_1 \in_R \mathbb{G}_1$ and computes $h_1 = u_1^{x'_1}$.

(iv) $\mathcal{B}$ selects $\gamma, \mu, x'_3, y'_3, \{s_j\}_{att_j \in Att} \in_R \mathbb{Z}_p^*$, $g_2 \in_R \mathbb{G}_2$ and computes $g_1 = \psi(g_2), w = g_2^{\gamma}, \mathcal{U} = g_2^{\frac{1}{\mu}}, g_{att_1} = g_2^{s_1}, ..., g_{att_m} = g_2^{s_m}$, where $m = |Att|$, selects $\hat{h}_3 \in_R \mathbb{G}_2$, and computes $\hat{u}_3 = \hat{h}_3^{1/x'_3}$ and $\hat{v}_3 = \hat{h}_3^{1/y'_3}$.

(v) $\mathcal{B}$ sets a group public key, $gpk = (g_1, g_2, u_1, h_1, u_2, h_2, \hat{u}_3, \hat{v}_3, \hat{h}_3, w, \mathcal{U}, \{g_{\text{att}_j}\}_{\text{att}_j \in Att})$, Attribute opening key, $tk_{att} = (x'_3, y'_3)$, and an issuing key, $ik = (\gamma, \mu, S)$.

(vi) $\mathcal{B}$ gives gpk, ik, tk_{att} to $\mathcal{A}$.

Hash queries. At any time, $\mathcal{A}$ can query the hash function $\mathcal{H}$. $\mathcal{B}$ responds with random values with consistency.

Phase 1. $\mathcal{A}$ requests the queries as given in `user anonymity` game. $\mathcal{B}$ answers to these queries as the real settings of ABGS scheme, since $\mathcal{B}$ knows all the values. For $d = 0$, x'_1 is unknown, $\mathcal{B}$ uses the following procedure to answer the `OpenUser` query on any given σ; $\mathcal{B}$ gets $\widehat{g} = g_2^{s_{T_2}}$ by following the procedure in `TraceAtt` algorithm, computes $A_i^{\mu s_{T_1}} = \frac{C_3}{C_4^{x'_2}}$, computes $g_2^{s_{T_1}} = v_T / \widehat{g}$ and compare $e(A_i^{\mu s_{T_1}}, \mathcal{U}) \stackrel{?}{=} e(A_j, g_2^{s_{T_1}}), \forall A_j$ in $r\vec{e}g$ and responds to $\mathcal{A}$ with j. This makes the scheme `CCA2` secure, since opening

oracle `OpenUser` is provided to $\mathcal{A}$.

Challenge. $\mathcal{A}$ outputs a message M^*, a predicate Υ^*, an attribute set ζ and two uncorrupted members i_0, i_1, such that $\Upsilon^*(\zeta) = 1, \zeta \subseteq \mathcal{A}_{i_0}$ and $\zeta \subseteq \mathcal{A}_{i_1}$, to be challenged. $\mathcal{T}_{\Upsilon^*}$ is a public values of the predicate Υ^*. $\mathcal{B}$ picks $\rho \in_R \{0,1\}$. $\mathcal{B}$ try to simulate the challenged signatures σ^* from A_{i_ρ} as follows,

- the encryption: according to d, by choosing an additional random bit d'.
  - if $d = 0$, $\mathcal{B}$ computes $C_1^* = A_{i_\rho} H_i, C_2^* = U_i, C_3^* = \Pi_{\text{att}_j \in \zeta} T_{i_{d'},j}^{\Delta_{\text{att}_j}} h_2^{\alpha_2} = A_{i_{d'}}^{\mu s_{T_1}} h_2^{\alpha_2}, C_4^* = u_2^{\alpha_2}$, for a random α_2 and computes the other values as the real settings;
  - if $d = 1$, $\mathcal{B}$ computes $C_1^* = A_{i_{d'}} h_1^{\alpha_1}, C_2^* = u_1^{\alpha_1}, C_3^* = \Pi_{\text{att}_j \in \zeta} T_{i_\rho,j}^{\Delta_{\text{att}_j}} H_i = A_{i_\rho}^{\mu s_{T_1}} H_i, C_7^* = U_i$, for a random α_1 and computes the other values as the real settings.
- the proof of validity `SPK` V^* is simulated by selecting a random $c^*, s_{\alpha_1}^*, s_{\alpha_2}^*, s_{\alpha_3}^*, s_{\beta_3}^*, s_{x_i}^*, s_\tau^*, s_\delta^* \in_R \mathbb{Z}_p^*$ and computing the corresponding $(\{R_i^*\}_{i=1}^5, R_{Att}^*)$ by following the procedure given in `Verify` algorithm and patching the hash oracle at $\mathcal{H}(gpk, M^*, \{C_i^*\}_{i=1}^7,$
  $\{R_i^*\}_{i=1}^5, R_{Att}^*)$ to c^*. If this backpatch fails then $\mathcal{B}$ outputs a random bit and aborts. This probability of failure is $(q_S + q_H)/p$.

In case of failure, $\mathcal{B}$ outputs a random bit and abort, otherwise, $(\{C_i^*\}_{i=1}^7, V^*)$ is the signature on M^* given back to $\mathcal{A}$.

Phase 2. $\mathcal{A}$ requests the queries as given in `user anonymity` game and $\mathcal{B}$ answers it similar to phase 1.

Output. $\mathcal{A}$ outputs its guess $\rho' \in \{0,1\}$ with advantage ϵ. Our algorithm $\mathcal{B}$ outputs 0 if $\rho = \rho'$ (indicating that $H = h^{a'}$,DDH tuple); otherwise $\mathcal{B}$ outputs 1 (indicating that is a random tuple).

If it is a DDH tuple, and $d' = \rho$, then for $d = \{0, 1\}$ the ElGamal encryption components(C_1^*, C_2^*) and (C_3^*, C_4^*) always encrypts A_ρ, this is a valid signature (the advantage of $\mathcal{A}$ in guessing ρ is ϵ). However, if $d' \neq \rho$, both certificates are encrypted, $\mathcal{A}$ has thus no advantage in guessing ρ. If this is a random tuple then the signature is independent of ρ and then the adversary's advantage is 0. As a consequence, it has an advantage $\epsilon/4$ in distinguishing DDH tuples.

Therefore the advantage of $\mathcal{B}$ in breaking DDH assumption when there is no failure and when $\mathcal{B}$ guesses correctly which type of adversary $\mathcal{A}$ is, is at least $\frac{1}{2}\left(\left(\frac{\epsilon}{4} - \frac{q_S+q_H}{p}\right) + \left(\frac{\epsilon}{4} - \frac{q_S+q_H}{p}\right)\right) \leq \frac{\epsilon}{4} - \frac{q_S+q_H}{p}$. □

Theorem 3.4.6 *We suppose an adversary $\mathcal{A}$ breaks the traceability of the proposed scheme with the advantage ϵ. Then, in the random oracle model, we can construct an algorithm $\mathcal{B}$ that breaks the q-SDH assumption with the advantage $\frac{1}{6}\epsilon$.*

Proof Since the membership certificate format is similar to the one proposed in [48], the proof is similar to the proof given in [48; 51]. The input of simulator $\mathcal{B}$ is $(g, g', g'_1, ..., g'_q) \in \mathbb{G}_1 \times \mathbb{G}_2^{q+1}$, where $g = \psi(g'), g'_i = (g')^{\xi^i}$ (for $i \in [1, q]$) and let $g'_0 = g'$. Let $q-1$ be the total number of members. $\mathcal{B}$ simulates KeyGen as follows:

(i) $\mathcal{B}$ selects $\nu, \mu, \{x_i\}_{i=1}^{q-1}, \{y_i\}_{i=1}^{q-1}, x'_1, x'_2, x'_3, y'_3, \{s_j\}_{(\forall \mathrm{att}_j \in Att)} \in_R \mathbb{Z}_p^*$, $u_2 \in_R \mathbb{G}_1$, and set $h_2 = u_2^{x'_2}$, and selects $\hat{h}_3 \in_R \mathbb{G}_2$, and set $\hat{u}_3 = \hat{h}_3^{1/x'_3}$ and $\hat{v}_3 = \hat{h}_3^{1/y'_3}$.

(ii) $\mathcal{B}$ selects a target user $\mathrm{U}_{i^\star} \in \{\mathrm{U}_1, ..., \mathrm{U}_{q-1}\}$, and sets $\gamma = \xi - x_{i^\star}$. $\mathcal{B}$ computes g_1, g_2, h_1 and w as follows:

- Let $f(y) = \prod_{i=1}^{q-1}(y + x_i)$. Therefore, $f(\gamma) = f(\xi - x_{i^\star}) = \prod_{i=1}^{q-1}(\xi - x_{i^\star} + x_i) = \sum_{i=0}^{q-1}(\alpha_i \xi^i)$, where $\alpha_0, ..., \alpha_{q-1} \in \mathbb{Z}_p$ are the coefficients of the polynomial $f(\gamma)$, are computable.
- Let $f_i(y) = f(y)/(y + x_i) = \prod_{j=1, j\neq i}^{q-1}(y + x_j)$. Thus, $f_i(\gamma) = f_i(\xi - x_{i^\star}) = \prod_{j=1, j\neq i}^{q-1}(\xi - x_{i^\star} + x_j) = \sum_{j=0}^{q-2}(\beta_j \xi^j)$, where $\beta_0, ..., \beta_{q-2}$ are the coefficients of the polynomial $f_i(\gamma)$, are computable.

- Note that, $g^{f_i(\gamma)} = (g^{f(\gamma)})^{\frac{1}{x_i+\gamma}}$ and $f_{i^\star}(\gamma) = f(\gamma)/(\xi - x_{i^\star} + x_{i^\star}) = f(\gamma)/\xi$.
- Set $g_2 = (g')^{\nu f(\gamma)}/(g')^{x'_1 y_{i^\star} f_{i^\star}(\gamma)} = \prod_{i=0}^{q-1}(g'_i)^{\nu\alpha_i} / \prod_{j=0}^{q-2}(g'_j)^{x'_1 y_{i^\star}\beta_j}$
- $u_1 = g^{f_{i^\star}(\gamma)} = \prod_{j=0}^{q-2}\psi(g'_j)^{\beta_j}$
- $g_1 = \psi(g_2) = u_1^{\nu\xi}/h_1^{y_{i^\star}}$
- $w = \left\{\prod_{i=0}^{q-1}(g'_{i+1})^{\alpha_i\nu} / \prod_{j=0}^{q-2}(g'_{j+1})^{\beta_j x'_1 y_{i^\star}}\right\} / g_2^{x_{i^\star}}$
  $= \left\{(g')^{\nu\xi f(\gamma)}/(g')^{x'_1 y_{i^\star}\xi f_{i^\star}(\gamma)}\right\} / g_2^{x_{i^\star}}$
  $= g_2^{\xi - x_{i^\star}} = g_2^{\gamma}$

Thus $\mathcal{B}$ can compute these values by using the $q-$SDH input instances.

(iii) $\mathcal{B}$ computes $h_1 = u_1^{x'_1}$ and other parameters as the real settings.

(iv) $\mathcal{B}$ makes $params = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3, e, \psi, \mathcal{H}, Att), ok_{user} = x'_1, tk_{att} = (x'_3, y'_3), ik = (\gamma, \mu, \{s_j\}_{\mathrm{att}_j \in Att})$ and $gpk = (g_1, g_2, u_1, h_1, u_2, h_2, \hat{u}_3, \hat{v}_3, \hat{h}_3, w, \mathcal{U}, \{g_{\mathrm{att}_j}\}_{\mathrm{att}_j \in Att})$. $params, gpk, ok_{user}$ and tk_{att} are given to $\mathcal{A}$.

In the Join queries, $\mathcal{B}$ can get a secret value y_i of a corrupted user by extracting the commitment value. $\mathcal{B}$ computes a group membership certificate as follows:
In the case of $i = i^\star : A_{i^\star} = u_1^{\nu} = (u_1^{\nu\xi})^{\frac{1}{\xi}} = (g_1 h_1^{y_{i^\star}})^{\frac{1}{\gamma + x_{i^\star}}}$.
In the case of $i \neq i^\star$:Compute A_i as follows:

$$A_i = \left(g^{x'_1 \prod_{j=1, j\neq i, i^\star}^{q-1}(\xi - x_{i^\star} + x_j)}\right)^{y_i - y_{i^\star}} g^{\nu f_i(\gamma)}$$

$$= g^{\frac{x'_1 y_i}{\xi + x_i - x_{i^\star}} \prod_{j=1, j\neq i^\star}^{q-1}(\xi - x_{i^\star} + x_j)} \times \left\{g^{\nu f(\gamma)} / g^{z y_{i^\star} \prod_{j=1, j\neq i^\star}^{q-1}(\xi - x_{i^\star} + x_j)}\right\}^{\frac{1}{\xi + x_i - x_{i^\star}}}$$

$$= (g_1 h_1^{y_i})^{\frac{1}{\gamma + x_i}}$$

$\mathcal{B}$ can compute $\{T_{i,j}\}_{\mathrm{att}_j \in \mathcal{A}_i} = \{A_i^{s_j \mu}\}_{\mathrm{att}_j \in \mathcal{A}_i}$. Now $\mathcal{B}$ can answer all the queries made by an adversary. Finally, $\mathcal{A}$ outputs a forged signature $\sigma^* = (\{C_i^*\}_{i=1}^7, c^*, s_x^*,$ $s_{\alpha_1}^*, s_{\alpha_2}^*, s_{\alpha_3}^*, s_{\beta_3}^*, s_\tau^*, s_\delta^*)$ with ϵ advantage.
By using the Forking Lemma, $\mathcal{B}$ can get the two valid signatures $(\{C_i^*\}_{i=1}^7, c^*, s_x^*,$ $s_{\alpha_1}^*, s_{\alpha_2}^*, s_{\alpha_3}^*, s_{\beta_3}^*, s_\tau^*, s_\delta^*)$ and $(\{C_i^*\}_{i=1}^7, c', s'_x, s'_{\alpha_1}, s'_{\alpha_2}, s'_{\alpha_3}, s'_{\beta_3}, s'_\tau, s'_\delta)$ with probability $\epsilon' \geq \frac{1}{5} - \frac{8q_h}{\eta 2^k}, \eta > \frac{240 q_H}{2^k}$ [48]. Let $c'' = c^* - c', s''_x = s_x^* - s'_x, \{s''_{\alpha_i} = s_{\alpha_i}^* - s'_{\alpha_i}\}_{i=1}^3, s''_{\beta_3} = s_{\beta_3}^* - s'_{\beta_3}, s''_\delta = s_\delta^* - s'_\delta$ and $s''_\tau = s_\tau^* - s'_\tau$. Let $\tilde{x} = s''_x/c'', \tilde{\alpha_1} = s''_{\alpha_1}/c'', \tilde{\tau} = s''_\tau/c'', \tilde{A} = C_1^*/h_1^{\tilde{\alpha_1}}$, and $\tilde{y} = \tilde{\tau} - \tilde{\alpha_1}\tilde{x}$.

Now $(\tilde{A}, \tilde{x}, \tilde{y})$ is a valid member certificate because $\frac{e(C_1^*, w)}{e(g_1, g_2)} = \frac{e(h_1, g_2)^{\tilde{r}} e(h_1, w)^{\tilde{\alpha}}}{e(C_1^*, g_2)^{\tilde{x}}}$ holds. From the success of the adversary in the attack game, we know that $\tilde{A}$ does not belong to $\{A_i\}_{i=1}^{q-1}$. We assume that $\tilde{x} \neq x_{i^\star}$.
Consider,

$$\begin{aligned}
\tilde{A} &= (g_1 h_1^{\tilde{y}})^{\frac{1}{\tilde{x}+\gamma}} \\
&= (u_1^{\nu\xi} h_1^{\tilde{y}-y_{i^\star}})^{\frac{1}{\tilde{x}+\gamma}} \\
&= u_1^{\frac{\nu\xi + x_1'(\tilde{y}-y_{i^\star})}{\tilde{x}+\gamma}} \\
&= \left(g^{(\nu\xi + x_1'(\tilde{y}-y_{i^\star})) \prod_{i=1, i\neq i^\star}^{q-1} (\xi + x_i - x_{i^\star})} \right)^{\frac{1}{\tilde{x}+\xi-x_{i^\star}}} \\
&= \left(g^{\sum_{i=0}^{q-1} z_i \xi^i} \right)^{\frac{1}{\tilde{x}+\xi-x_{i^\star}}} \text{ (can be written in this form)} \\
&= g^{\frac{\tilde{z}_0}{\tilde{x}+\xi-x_{i^\star}} + \sum_{i=1}^{q-1} \tilde{z}_i \xi^i} \text{(can be written in this form)}
\end{aligned}$$

The polynomial coefficients $z_0, ..., z_{q-1}, \tilde{z}_0, \tilde{z}_1, ..., \tilde{z}_{q-1}$ are computable. Let $x = \tilde{x} - x_{i^\star}$, then $(\tilde{A}/g^{\sum_{i=1}^{q-1} \tilde{z}_i \xi^i})^{\frac{1}{\tilde{z}_0}} = g^{\frac{1}{x+\xi}}$ holds. Therefore $(x, g^{\frac{1}{x+\xi}})$ is the new SDH tuple. If $\tilde{x} = x_{i^\star}$ then $(0, g^{\frac{1}{\xi}})$ will be the new SDH tuple. The advantage of $\mathcal{B}$ is $(\frac{1}{5} - \frac{8q_H}{\eta 2^k})\epsilon \geq \frac{1}{6}\epsilon$, since $\eta > \frac{240 q_H}{2^k}$. □

Theorem 3.4.7 *We suppose an adversary $\mathcal{A}$ breaks the non-frameability of the proposed scheme with the advantage ϵ. Then, we can construct an algorithm $\mathcal{B}$ that breaks the DL assumption with the advantage $\frac{1}{12}(1 + \frac{1}{n})\epsilon$, where n is the number of honest members.*

Proof The proof is similar to the one given in [48; 51]. The input of simulator $\mathcal{B}$ is $(g, g') \in \mathbb{G}_2 \times \mathbb{G}_2$, let $\xi = log_g g'$. We consider the two types of adversaries by the results of the `OpenUser` algorithm. We explain the details of classification of the adversary in the proof. Let q be the number of all members, n be the number of honest members, and $q_1 = q - n$ be the number of corrupt members. We assume that all initial members $\{U_1, ..., U_n\}$ are honest. $\mathcal{B}$ simulates KeyGen as follows:

(i) $\mathcal{B}$ selects $d \in_R \{0,1\}$. If $d=1$, then $\mathcal{B}$ selects a target user $\mathrm{U}_{i^\star} \in \{\mathrm{U}_1, ..., \mathrm{U}_n\}$. Note that $d=0$ means $\mathcal{B}$ guesses that $\mathcal{A}$ is Type 1 Adversary, and $d=1$ means $\mathcal{B}$ guesses that $\mathcal{A}$ is Type 2 Adversary.

(ii) $\mathcal{B}$ computes the group public key and member certificates as follows:

(a) $\mathcal{B}$ selects $\gamma, x_1', \{s_j\}_{(\forall \mathrm{att}_j \in Att)}, \{x_i, y_i\}_{i=1}^q \in_R \mathbb{Z}_p^*$. If $d=1$, then set $y_{i^\star} = \xi$.

(b) If $d=0$, then $\mathcal{B}$ sets $g_1 = \psi(g), g_2 = g$ and $u_1 = \psi(g')$.

(c) If $d=1$, then $\mathcal{B}$ selects $g_2 \in_R \mathbb{G}_2$ and sets $g_1 = \psi(g_2), u_1 = \psi(g)$ and $y_{i^\star} = \xi$.

(d) $\mathcal{B}$ computes $w = g_2^{\gamma}, h_1 = u_1^{x_1'}$.

(e) $\mathcal{B}$ computes member certificates $\{(A_i, x_i, y_i)\}_{i=1}^q$ by using γ. If $d=1$, then $A_{i^\star} = (g_1 \psi(g')^{x_1'})^{\frac{1}{x_{i^\star}+\gamma}} = (g_1 u_1^{y_{i^\star} x_1'})^{\frac{1}{x_{i^\star}+\gamma}} = (g_1 h_1^{y_{i^\star}})^{\frac{1}{x_{i^\star}+\gamma}}$.

(f) $\mathcal{B}$ computes other public values, and gets $params = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3, e, \psi, \mathcal{H}, Att)$, $ok_{user} = x_1', tk_{att} = (x_3', y_3'), ik = (\gamma, \mu, S)$, and $gpk = (g_1, g_2, u_1, h_1, u_2, h_2, \hat{u}_3, \hat{v}_3, \hat{h}_3, w, \mathcal{U}, \{g_{\mathrm{att}_j}\}_{\mathrm{att}_j \in Att})$.

(iii) $\mathcal{B}$ gives $params, gpk, ik, ok_{user}$ and tk_{att} to $\mathcal{A}$.

In Join queries, $\mathcal{A}$ knows (A_i, x_i), for $i \in [1, q]$, because $\mathcal{A}$ plays the role of corrupted GM. However, $\mathcal{A}$ cannot know secret key of a target user $y_i^\star$. For Signing queries, $\mathcal{B}$ makes a group signature by using (A_i, x_i, y_i), and return its signature, if $d=1$ and $i = i^\star$, then $\mathcal{B}$ aborts. For Corruption queries, $\mathcal{B}$ answers y_i, if $d=1$ and $i = i^\star$, then $\mathcal{B}$ aborts. Finally, $\mathcal{A}$ outputs the valid group signature for honest user, say U_k. We can get the member certificate $(\tilde{A}, \tilde{x}, \tilde{y})$ by using the same technique as for traceability. We define a Type 1 adversary $\mathcal{A}$, which is the case of $\tilde{A} = A_k \in \{A_i\}_{i=1}^n$ and $\tilde{x} \neq x_k$. We define a Type 2 adversary $\mathcal{A}$, which is the case of $(\tilde{A}, \tilde{x}) = (A_k, x_k)$.

- In the case of Type 1 : If $d \neq 0$, then $\mathcal{B}$ aborts. Otherwise $\tilde{A} = (g_1 h_1^{\tilde{y}})^{\frac{1}{\tilde{x}+\gamma}} = (g_1^{1+x_1'\xi\tilde{y}})^{\frac{1}{\tilde{x}+\gamma}}$ holds. As $\tilde{A} = A_k = (g_1 h_1^{y_k})^{\frac{1}{x_k+\gamma}} = (g_1^{1+x_1'\xi y_k})^{\frac{1}{x_k+\gamma}}$ holds. Therefore, $\mathcal{B}$ can compute $\xi = \frac{\tilde{x}-x_k}{x_1'\{\tilde{y}(x_k+\gamma)-y_k(\tilde{x}+\gamma))\}}$.

- In the case of Type 2 : If $d \neq 1$, then $\mathcal{B}$ aborts. If $k \neq i^\star$, then $\mathcal{B}$ aborts. Otherwise, $\tilde{A} = (g_1 h_1^{\tilde{y}})^{\frac{1}{\tilde{x}+\gamma}} = (g_1 \psi(g)^{x_1' \tilde{y}})^{\frac{1}{\tilde{x}+\gamma}}$ holds. Moreover, $\tilde{A} = A_{i^\star} = (g_1 h_1^{y_{i^\star}})^{\frac{1}{x_{i^\star}+\gamma}} = (g_1 \psi(g)^{x_1' y_{i^\star}})^{\frac{1}{x_{i^\star}+\gamma}}$ holds. Therefore $\mathcal{B}$ can get $\xi = \tilde{y}$.

The advantage of $\mathcal{B}$ is $(\frac{1}{2}(\frac{1}{5} - \frac{8q_H}{\eta 2^k})\epsilon + \frac{1}{2}\frac{1}{n}(\frac{1}{5} - \frac{8q_H}{\eta 2^k})\epsilon) > \frac{1}{12}(1 + \frac{1}{n})\epsilon$, since $\eta > \frac{240q_H}{2^k}$.□

Theorem 3.4.8 *The proposed scheme preserves collusion resistance of attribute certificates.*

Proof We prove the theorem using two lemmas. In lemma 3.4.9, we show that it is negligible to produce a group signature using forged attribute certificates. In Lemma 3.4.10, we show that it is impossible to produce a group signature by using appropriately the valid attribute certificates of colluding group members. □

Lemma 3.4.9 *The probability that a signature by forged attribute certificates passes the verification,* $\Pr(\mathsf{Verify}(params, gpk, M, \sigma, \Upsilon) = 1 \wedge \Upsilon(\zeta) \neq 1)$, *is at most* $1/p$.

Proof We assume that $\zeta_i = \{\mathrm{att}_1, \mathrm{att}_2, ..., \mathrm{att}_{\hat{m}}\}$, such that $\Upsilon(\zeta_i) = 1$, without limiting the generality of the forging. The equations used in the scheme to prove the knowledge of $(\alpha_1, \alpha_2, \alpha_3, \beta_3, x_i, \tau, \delta)$ are as follows:

$$\frac{e(C_1, w)}{e(g_1, g_2)} = \frac{e(h_1, g_2)^\tau e(h_1, w)^{\alpha_1+\beta_1}}{e(C_1, g_2)^{x_i}} \tag{3.4}$$

$$C_2 = u_1^{\alpha_1} \tag{3.5}$$

$$C_4 = u_2^{\alpha_2} \tag{3.6}$$

$$C_6 = \hat{u}_3^{\alpha_3} \tag{3.7}$$

$$C_7 = \hat{v}_3^{\beta_3} \tag{3.8}$$

$$\frac{e(C_3,\mathcal{U})e(C_1,C_5)}{e(C_1,v_T)} = \frac{e(h_2,\mathcal{U})^{\alpha_2}e(C_1,\hat{h}_3)^{\alpha_3+\beta_3}e(h_1,g_2)^{\delta}}{e(h_1,v_T)^{\alpha_1}} \tag{3.9}$$

In (3.4) , a signer proves that $C_1 = A_i h_1^{\alpha_1}$, where A_i is a valid membership certificate. Equation (3.5) to (3.8) obviously holds. We can change (3.9) into $\frac{e(C_3,g_2^{\frac{1}{\mu}})e(A_i h_1^{\alpha_1},C_5)}{e(A_i h_1^{\alpha_1},g_2^{s_T})} = \frac{e(h_2,g_2^{\frac{1}{\mu}})^{\alpha_2}e(A_i h_1^{\alpha_1},\hat{h}_3)^{\alpha_3+\beta_3}e(h_1,g_2)^{\delta}}{e(h_1,g_2^{s_T})^{\alpha_1}}$, since the validity of SPK C_1 has already been proven (namely (3.4)). $C_3 = A_i^{\mu s_{T_1}} h_2^{\alpha_2}$, $C_5 = g_2^{s_{T_2}}\hat{h}_3^{\alpha_3+\beta_3}$ and $\delta = \alpha_1 s_{T_2} : s_{T_1} + s_{T_2} = s_T$ holds and $s_{T_2} \neq s_T$ since $s_{T_1} \neq \phi$ because atleast one $T_{i,j}$ is needed to get ride of $\frac{1}{\mu}$ in $\mathcal{U}$ in the equation, i.e. $A_i^{s_j\mu}$ is needed. Let $s_{T_2} = \sum_{d_j \in D_T^{\zeta}} \Delta_{d_j} s_{d_j}$ and we assume that $C_3 = A_i^{s_{T_1}\mu} h_2^{\alpha_2} = \Pi_{\mathrm{att}_j \in \zeta} A_i^{t_j \Delta_j \mu} h_2^{\alpha_2}$, where $t_j \in \mathbb{Z}_p^*$. Then

$$\sum_{\mathrm{att}_j \in \zeta} \Delta_{\mathrm{att}_j} t_j + \sum_{d_j \in D_T^{\zeta}} \Delta_{d_j} s_{d_j} = s_T \tag{3.10}$$

should holds. If $t_j = s_j(\mathrm{att}_j \in \zeta)$, then (3.10) obviously holds. On the contrary, we assume that $t_j(\mathrm{att}_j \in \zeta)$ satisfies (3.10). We set the values of s_{T_1}, Δ_j as constants. We randomly choose $t_j \in \mathbb{Z}_p^*(\text{for } j = 1, 2, ..., \hat{m} - 1)$, and set $t_{\hat{m}} = (s_{T_1} - \Sigma_{\mathrm{att}_j \in \zeta \setminus \{\mathrm{att}_{\hat{m}}\}} \Delta_j t_j)/\Delta_{\hat{m}}$. Then $(t_1, t_2, ..., t_{\hat{m}})$ obviously satisfies (3.10). Therefore, the total number of solution vectors $(t_1, t_2, ..., t_{\hat{m}})$ is $p^{\hat{m}-1}$. Therefore, the probability that the randomly chosen vector $(t_1, t_2, ..., t_{\hat{m}})$ satisfying (3.10) is $p^{\hat{m}-1}/p^{\hat{m}} = 1/p$. This implies that, the probability that a signature made by forged attribute certificates (except the genuine certificates) satisfying (3.10) is $\frac{p^{\hat{m}-1}-1}{p^{\hat{m}}} = \frac{1}{p}(1 - \frac{1}{p^{\hat{m}-1}})$. Next, we consider $t_j = s_j(\text{for } j = 1, 2, ..., l)$, where $l < \hat{m}$. Let $l = \hat{m} - 1$, this means a signer has valid attribute certificates of $\zeta \setminus \{\mathrm{att}_{\hat{m}}\}$. We assume that a signature satisfies (3.9). Then $t_{\hat{m}} = (s_{T_1} - \Sigma_{\mathrm{att}_j \in \zeta \setminus \{\mathrm{att}_{\hat{m}}\}} \Delta_j t_j)/\Delta_{\hat{m}} = s_{\hat{m}}$ hold. This means that the signer has valid attribute certificates of ζ, and the signature is not a forged signature. Therefore, we set $l < \hat{m} - 1$. This means a signer has valid attribute certificates of $\zeta \setminus \{\mathrm{att}_{l+1}, ..., \mathrm{att}_{\hat{m}}\}$. Then there exist the number of $p^{\hat{m}-l-1} - 1$ pairs $(t_{l+1}, ..., t_{\hat{m}})$ such that $(s_1, ..., s_l, t_{l+1}, ..., t_{\hat{m}})$ satisfies (3.10) and $(t_{l+1}, ..., t_{\hat{m}}) \neq (s_{l+1}, ..., s_{\hat{m}})$. The total number of vectors $(t_{l+1}, ..., t_{\hat{m}})$ is $p^{\hat{m}-l}$. Therefore, the probability that a signature made by valid attribute certificates of $\zeta \setminus \{\mathrm{att}_{l+1}, ..., \mathrm{att}_{\hat{m}}\}$ and forged attribute certificates of $\{\mathrm{att}_{l+1}, ..., \mathrm{att}_{\hat{m}}\}$ satisfying (3.10) is $\frac{p^{\hat{m}-l-1}-1}{p^{\hat{m}-l}} = \frac{1}{p}(1 - \frac{1}{p^{\hat{m}-l-1}}) \leq \frac{1}{p}$.

So, a verifier can decide whether an anonymous signer has valid attribute certificates when a verifier is given a signature which satisfies (3.9). □

Lemma 3.4.10 *Even if some malicious participants* $\mathrm{U}_{i_1}, ..., \mathrm{U}_{i_k} (k > 1)$ *with the set of attributes* $\zeta_{i_1}, ..., \zeta_{i_k}$ *collude, they cannot make a valid signature associated with an predicate* Υ, *where* $(\cup_{j=1}^{k} \Upsilon(\zeta_{i_j}) = 1)$ *and* $\Upsilon(\zeta_{i_j}) \neq= 1 (j = 1, ..., k)$ *with non-negligible probability.*

Proof. Without loss of generality, we assume that U_0 with ζ_0 and U_1 with ζ_1 represent malicious participants. U_0 and U_1 attempt to make a valid signature associated with Υ which satisfies $\Upsilon(\zeta_0 \cup \zeta_1) = 1, \Upsilon(\zeta_0) \neq 1$ and $\Upsilon(\zeta_1) \neq 1$. They can make the SPK of $(\alpha, x_0, \tau, \delta)$ satisfy (3.4) to (3.8) because they have a valid membership certificate A_0. We assume that $A_0^t = A_1$, where $t \in \mathbb{Z}_p^*$. Note that the probability of $t = 1$ is negligible. Then, from (3.10), $\Sigma_{\mathrm{att}_j \in \zeta_0} \Delta_j s_j + \Sigma_{\mathrm{att}_j \in \zeta_1} t \Delta_j s_j \neq s_{T_1}$ holds since $t \neq 1$. This means that they cannot use $\{T_{i_0,j}\}_{\mathrm{att}_j \in \zeta_0}$ and $\{T_{i_1,j}\}_{\mathrm{att}_j \in \zeta_1}$ simultaneously. □

3.5 Short ABGS

In this section, we present a scheme with shorter signature than the previous one but opening of user anonymity and attribute anonymity are not independent. In this scheme the key tk_{att} can reveal the user identity of the signature. Except this the scheme preserves all the security features of the previous scheme. All the algorithms are same as the previous scheme except `Sign` algorithm and a few modifications to `KeyGen` algorithm.

`Sign`$(params, gpk, sk_i, \zeta, M, \Upsilon)$:

A signer U_i signs a message $M \in \{0,1\}^*$ as follows:

(i) Get the public values of Υ, $\mathcal{T}_\Upsilon = (\{s_{d_j}\}_{d_j \in D_{T_\Upsilon}}, v_T, T^{ext})$, from the public repository.

(ii) U_i chooses $\zeta \subseteq \mathcal{A}_i$ as an input such that $\Upsilon(\zeta) = 1$.

(iii) U_i runs $\texttt{MakeSimplifiedTree}(\zeta, T^{ext})$ and gets the corresponding $\Delta_{\text{att}_j}(\forall \text{att}_j \in \zeta)$, and $\Delta_{d_j}(\forall d_j \in D_T^{\zeta})$.

(iv) Note that $\Sigma_{\text{att}_j \in \zeta} \Delta_{\text{att}_j} s_j + \Sigma_{d_j \in D_T^{\zeta}} \Delta_{d_j} s_{d_j} = s_T$. Let $s_{T_1} = \Sigma_{\text{att}_j \in \zeta} \Delta_{\text{att}_j} s_j$ and $s_{T_2} = \Sigma_{d_j \in D_T^{\zeta}} \Delta_{d_j} s_{d_j}$. Thus, $s_{T_1} + s_{T_2} = s_T$.

(v) U_i selects $\alpha_1, \alpha_2, \alpha_3 \in_R \mathbb{Z}_p^*$, and computes $C_1 = A_i h_1^{\alpha_1}$,
$C_2 = u_1^{\alpha_1}, C_3 = \Pi_{\text{att}_j \in \zeta} T_{i,j}^{\Delta_{\text{att}_j}} h_2^{\alpha_2} = A_i^{\mu s_{T_1}} h_2^{\alpha_2}, C_4 = u_2^{\alpha_2}$,
$C_5 = \Pi_{d_j \in D_T^{\zeta}} A_i^{\Delta_{d_j} s_{d_j}} h_3^{\alpha_3} = A_i^{s_{T_2}} h_3^{\alpha_3}, C_6 = u_3^{\alpha_3}$.

(vi) U_i sets $\tau = \alpha_1 x_i + y_i$, and computes
$V = \texttt{SPK}\{(\alpha_1, \alpha_2, \alpha_3, x_i, \tau) : \frac{e(C_1, w)}{e(g_1, g_2)} = \frac{e(h_1, g_2)^{\tau} e(h_1, w)^{\alpha_1}}{e(C_1, g_2)^{x_i}} \bigwedge C_2 = u_1^{\alpha_1}$
$\bigwedge C_4 = u_2^{\alpha_2} \bigwedge C_6 = u_3^{\alpha_3} \bigwedge \frac{e(C_3, \mathcal{U}) e(C_5, g_2)}{e(C_1, v_T)} = \frac{e(h_2, \mathcal{U})^{\alpha_2} e(h_3, g_2)^{\alpha_3}}{e(h_1, v_T)^{\alpha_1}}\}(M)$.

(a) U_i chooses blinding values $r_{\alpha_1}, r_{\alpha_2}, r_{\alpha_3}, r_{x_i}, r_{\tau} \in_R \mathbb{Z}_p^*$.

(b) U_i computes $R_1 = \frac{e(h_1, g_2)^{r_{\tau}} e(h_1, w)^{r_{\alpha_1}}}{e(C_1, g_2)^{r_{x_i}}}$,
$R_2 = u_1^{r_{\alpha_1}}, R_3 = u_2^{r_{\alpha_2}}$,
$R_4 = u_3^{r_{\alpha_3}}, R_{Att} = \frac{e(h_2, \mathcal{U})^{r_{\alpha_2}} e(h_3, g_2)^{r_{\alpha_3}}}{e(h_1, v_T)^{r_{\alpha_1}}}$.

(c) U_i computes $c = \mathcal{H}(gpk, M, \{C_i\}_{i=1}^6, \{R_i\}_{i=1}^4, R_{Att})$.

(d) U_i computes $s_{\alpha_1} = r_{\alpha_1} + c\alpha_1, s_{\alpha_2} = r_{\alpha_2} + c\alpha_2, s_{\alpha_3} = r_{\alpha_3} + c\alpha_3, s_{x_i} = r_{x_i} + cx_i, s_{\tau} = r_{\tau} + c\tau,$.
Thus, $V = (c, s_{\alpha_1}, s_{\alpha_2}, s_{\alpha_3}, s_{x_i}, s_{\tau})$.

(vii) Outputs $\sigma = (\{C_i\}_{i=1}^6, V)$.

Here we can see that the signature algorithm contains 3 modules of ElGamal encryption scheme and signature contains 6 elements from $\mathbb{G}_1$ and 6 elements from $\mathbb{Z}_p^*$. One can derive the security analysis of this scheme similar to the previous scheme. Thus this scheme can be used where there is one authority for opening both user anonymity and attribute anonymity.

Table 3.1: Comparison of ABGS scheme with other schemes

	Reference [72]	Reference [51]	Our Scheme
Non-frameability	no	yes	yes
CCA - user anonymity	no	yes	yes
Attribute anonymity	no	no	yes
User revocation	yes	no	yes
Signature Length	O(Φ)	O(Φ)	O(1)
User's signing key length	$(\hat{m}+1)\|\mathbb{G}_1\| + \|\mathbb{Z}_p^*\|$	$(\hat{m}+1)\|\mathbb{G}_1\| + 2\|\mathbb{Z}_p^*\|$	$\|\mathbb{G}_1\| + \|\mathbb{Z}_p^*\|$
Assumption	DLDH, q-SDH	DDH, q-SDH, DL	DDH,DLDH, q-SDH, DL
Model	RO	RO	RO
Signing	$\binom{(7+2\Phi)\mathbb{G}_1+(5+\Phi)\mathbb{G}_3}{+(\Phi+1)e}$	$\binom{(9+3\Phi)\mathbb{G}_1+(1+\Phi)\mathbb{G}_2}{+8\mathbb{G}_3+3e}$	$(8+\Phi)\mathbb{G}_1 + 7\mathbb{G}_2 + 12\mathbb{G}_3 + 7e$
Verification	$\binom{(6+2r)\mathbb{G}_1+(8+2\Phi)\mathbb{G}_3}{+(\Phi+2r+1)e}$	$\binom{(11+2\Phi)\mathbb{G}_1+(\Phi+1)\mathbb{G}_2}{+14\mathbb{G}_3+6e}$	$6\mathbb{G}_1 + 6\mathbb{G}_2 + 19\mathbb{G}_3 + 12e$

3.6 Comparison

The group signature of the proposed scheme contains 4 elements from $\mathbb{G}_1$, 3 elements from $\mathbb{G}_2$ and 8 elements from $\mathbb{Z}_p^*$. Using the MNT family of curves [88], as described in [30], one can take p to be a 170-bit prime and use groups $\mathbb{G}_1$ and $\mathbb{G}_2$ where each element is 171-bits. Thus, the total group signature length is 2558 bits (= 320 bytes). Similarly for Short ABGS it is 256 bytes.
Let $\Phi = |\zeta|$, where ζ be the set of attributes which is associated with a signature and $m = |Att|$. Let $\hat{m}$ be the average number of attributes assigned to any user. In Table 3.1, we compare the efficiency of our scheme with the other schemes proposed in [72] and [51]. In this table, RO means Random oracle model, e represents the paring operation and r is the number of revoked members. Note that the verification cost of the proposed scheme is constant.

3.7 Summary

We have proposed an ABGS scheme with attribute anonymity and the constant size signature, and proven that it is secure under random oracle model. We have also included the revocation mechanism which makes the scheme attractive and practical. In our scheme the signature length is shorter and it is CCA2 secure. Moreover it is having independent opening of attribute anonymity and user anonymity.

Chapter 4

A VLR-ABGS Scheme with Backward Unlinkability and Attribute Anonymity in the Random Oracle Model

In the last chapter, we proposed an ABGS scheme with attribute anonymity. In this chapter, we propose an ABGS scheme with Verifier-Local Revocation (VLR) and backward unlinkability. In VLR schemes, only verifiers are involved in the revocation of a member, while signers are not. We prove that the scheme is secure under random oracle model with DL, q-SDH, DLIN and KEA1 assumptions.

4.1 Introduction

Khader proposed an ABGS scheme with VLR feature but does not address attribute anonymity [72]. Afterwards Emura et al. in [52] have proposed an ABGS scheme, but this scheme neither addresses the attribute anonymity issue nor provides the revocation feature. To the best of our knowledge there is only one ABGS scheme

with VLR feature proposed by Khader in [72] but the scheme does not have *backward unlinkability* feature nor addressed attribute anonymity. Moreover, in this scheme the signature length is linear in terms of the number of attributes. The *backward unlinkability* feature gives provision to suspend a group member for a certain period.

We propose an ABGS scheme with attribute anonymity and VLR feature. Furthermore, we add one more security feature namely, attribute unforgeability. *Attribute unforgeability* is the special case of collusion resistance security feature of [52], which means that it should be impossible for any individual member to satisfy the predicate with invalid set of attributes. Apart from ABGS schemes many VLR group signature (VLR-GS) schemes are proposed either in random oracle model [31; 38; 91] or in the standard model [82]. We note that to build a VLR-ABGS scheme with attribute anonymity in the standard model one can also combine an ABS scheme [70] with a VLR-GS scheme [82], but it incurs combined cost of both the schemes.

In Section 4.2, we present the proposed VLR-ABGS scheme with attribute anonymity and the related security definitions. The construction of the proposed VLR-ABGS scheme is given in Section 4.3. Its security analysis is given in Section 4.4. followed by comparison with previous schemes in Section 4.5. Finally we summarize in Section 4.6.

4.2 Proposed Scheme

In this section, we propose a VLR-ABGS scheme with backward unlinkability and attribute anonymity secure under the random oracle model. In verifier-local revocation group signatures (VLR-GS), originally suggested by Chaum in [42] and formalized by Boneh et al. in [31], the group manager maintains a periodically updated revocation list (RL) which is used by all verifiers to perform the revocation test and it makes sure that the signatures are not produced by a revoked member. Let k be the security parameter, $\mathbb{T}$ be the number of time intervals (it is polynomially bounded by k), A_i the membership certificate for U_i, $\{T_{i,j}\}_{\mathrm{att}_j \in \mathcal{A}_i}$ the attribute certificates of U_i, the $T_{i,j}$ is the attribute certificate of $att_j \in Att$ to U_i, sk_i denotes the group private key for the

member U_i which includes both A_i and $\{T_{i,j}\}_{\mathrm{att}_j \in \mathcal{A}_i}$. The signature verification algorithm is provided with an additional argument called the Revocation List (RL). The RL contains a token for each revoked user. The verification algorithm accepts all signatures issued by unrevoked users and reveals no information about which unrevoked user issued the signature. However, if a user is ever revoked by having his revocation token added to the RL, signatures from that user are no longer accepted. Let RL_t be a set, which contains the revocation tokens of the revoked users at the interval t, $RL = \{RL_t\}_{t=1}^{\mathbb{T}}$ be the public revocation list, n be the number of group members and $\vec{grt}$ be the $(n \times \mathbb{T})$-vector of revocation tokens, $\vec{grt} = \{grt[1][1], ..., grt[n][\mathbb{T}]\}$, where $grt[i][t]$ denotes the token of member U_i at interval t. Thus, $RL_t = \{grt[i][t]\}_{i \in \{1,...,n\}}$ it contains only revoked user tokens at the interval t. Note that $\vec{grt}$ is not publicly accessible.

The user U_i can make a group signature on a document M with the predicate Υ during the interval t if there exists a set of attributes $\zeta \subseteq \mathcal{A}_i$ with the user such that $\Upsilon(\zeta) = 1$ and $grt[i][t] \notin RL_t$.

Definition 4.2.1 (VLR-ABGS) *A VLR-ABGS scheme consists of the following algorithms. Unless otherwise indicated, algorithms are randomized.*

- $params \leftarrow \texttt{Setup}(1^k)$: *This algorithm takes the security parameter k as an input and returns the system parameter params.*
- $(gpk, ik, \vec{grt}) \leftarrow \texttt{KeyGen}(params)$: *This algorithm takes the system parameter params, and returns a group public key gpk, an issuing key ik and the revocation token vector, $\vec{grt}$.*
- $sk_i \leftarrow \texttt{Join}(\langle params, gpk, t, ik, upk_i, \mathcal{A}_i \rangle, \langle params, gpk, t, upk_i, usk_i \rangle)$: *This is the interactive group joining protocol. It takes as input params, gpk, the current time interval t, the issuing key ik, upk_i and U_i's attributes $\mathcal{A}_i \subseteq Att$ from* **GM**, *and $params, gpk, t, upk_i$, and usk_i from U_i. In the protocol U_i ends with a member secret key sk_i and* **GM** *ends with updated revocation token vector $\vec{grt}$ and registration table $\vec{reg}$.*

- $\sigma \leftarrow \texttt{Sign}(params, gpk, t, sk_i, \zeta, M, \Upsilon)$: *This algorithm takes* $params, gpk$, *the current time interval* t, sk_i, *an attributes set* $\zeta \subseteq \mathcal{A}_i$, *message* M, *and the predicate* Υ *as an input and returns a group signature* σ *on* M.

- $0/1 \leftarrow \texttt{Verify}(params, gpk, t, RL_t, M, \Upsilon, \sigma)$: *This is a deterministic algorithm which takes* $params, gpk$, *the interval value* t, M, σ, Υ *and a set of revocation tokens* RL_t *for the period* t *as an input and returns* $1/0$. *If* 1 *then the algorithm claims that the* σ *is a valid signature, otherwise,* σ *is invalid.*

Trace: As mentioned in [31], any such group signature scheme has an associated *implicit tracing* algorithm that traces a signature to the group member who generate it using the vector $\vec{grt}$: on input a valid message-predicate-signature tuple (M, Υ, σ) for period t, the opener can determine who was the author of σ by successively executing the verification algorithm on (M, Υ, σ) using the vector of revocation tokens (i.e., with $RL_t = \{grt[i][t]\}_{i \in \{1,...,n\}}$) and outputting the identity $i \in \{1, ..., n\}$ for which the verification algorithm returns 1.

Revoke: When the member U_i is revoked in the interval t, the `GM` publishes (or adds) the secret tokens $grt[i][t], ..., grt[i][\mathbb{T}]$ into the public lists $RL_t, RL_{t+1}, ..., RL_{\mathbb{T}}$, respectively.

Remark When a new attribute att_{m+1} is added, then an attribute certificate corresponding to that attribute att_{m+1} needs to be issued for the eligible user(s) only.

Entities: Following are the entities in VLR-ABGS scheme:

- The group manager `GM`, also known as *issuer*, has issuing key ik using which he enrolls a user into the group by allotting some privileges (in terms of attributes) say $\mathcal{A}_i \subseteq Att$ and issuing a user's private key sk_i, by running interactive `Join` algorithm with the user. Issuer revokes a group member by publishing the revocation token of the member and also can reveal the signer's identity from the group signature by using $\vec{grt}$.

- Group members or signers with their private keys sk run `Sign` algorithm to produce a group signature on a document M with predicate Υ if they possess valid attribute set which satisfies the predicate.
- Outsider or verifier who can seek a group signature for a document M with predicate Υ from group manager `GM`. He can also verify the group signature using the group public key, gpk.

Note Normally the `Setup` and `KeyGen` algorithms are run by some trusted party and he will distribute the keys to the concerned entities.

ABGS scheme is correct if a group signature is produced by the unrevoked group member using his signing key and his attribute set.

Definition 4.2.2 (`Correctness`) Correctness *requires that for all* $params \leftarrow \texttt{Setup}(1^k)$, *all* $(gpk, ik, \vec{grt}) \leftarrow \texttt{KeyGen}(params)$, *all* $t \in [1, \mathbb{T}]$, *all* Υ, *all* $RL_t \in RL$, *all* $\zeta \subseteq Att$ *and all* $M \in \{0,1\}^*$,

$$\texttt{Verify}(params, gpk, t, RL_t, M, \texttt{Sign}(params, gpk, t, sk_i, \zeta, M, \Upsilon), \Upsilon) = 1 \iff grt[i][t] \notin RL_t$$

In ABGS scheme a group member may have multiple attribute sets to satisfy the predicate and he can produce a group signature using one of them. An ABGS scheme preserves attribute anonymity if it is computationally difficult to identify with what attribute set he produces the signature.

Definition 4.2.3 (`Attribute anonymity`) *We say that the VLR-ABGS scheme preserves* attribute anonymity *if, for all honestly generated* $(gpk, ik, \vec{grt}) \leftarrow \texttt{KeyGen}(params)$, *for all* $t \in [1, \mathbb{T}]$, *for all predicates* Υ, *for all attribute sets* $\mathcal{A}_i \subseteq Att$ *such that there exist* $\zeta_1, \zeta_2 \subseteq \mathcal{A}_i$ *and* $\Upsilon(\zeta_1) = \Upsilon(\zeta_2) = 1$, *for all* $sk_i \leftarrow \texttt{Join}(\langle params, gpk, t, ik, upk_i, \mathcal{A}_i \rangle, \langle params, gpk, t, upk_i, usk_i \rangle)$ *and all messages* M, *the distributions* $\texttt{Sign}(params, gpk, t, sk_i, \zeta_1, M, \Upsilon)$ *and* $\texttt{Sign}(params, gpk, t, sk_i, \zeta_2, M, \Upsilon)$ *are equal.*

In other words, even the computationally unbounded adversary cannot link a signature to a set of attributes used to generate it.

ABGS scheme preserves backward unlinkability - user anonymity if there are at least two unrevoked group members possessing valid attribute sets and one of them produces the group signature then it should be computationally hard to identify who produced the group signature among them, even if they are revoked afterwards.

Definition 4.2.4 (`BU-user anonymity`) *We say that the VLR-ABGS scheme preserves* BU-user anonymity *if for all PPT $\mathcal{A}$, the probability that $\mathcal{A}$ wins the following game is negligible.*

- ***Setup:*** *The challenger $\mathcal{C}$ runs $(gpk, ik, \vec{grt}) \leftarrow \mathtt{KeyGen}(params)$. $\mathcal{C}$ gives gpk to $\mathcal{A}$.*
- ***Queries:*** *At the beginning of each period, $\mathcal{C}$ increments a counter t and notifies $\mathcal{A}$ about it. During the current interval t, $\mathcal{A}$ can send the following queries to $\mathcal{C}$,*
  - `Phase 1` : *$\mathcal{A}$ can send following queries to the challenger,*
    - Join[1]*: $\mathcal{A}$ can request $\mathcal{C}$ (to run), the Join procedure for any honest member of her choice. $\mathcal{A}$ plays the role of corrupted `GM` on these queries.*
    - Signing : *$\mathcal{A}$ can request a group signature σ on any, message M, predicate Υ, U_i and set of attributes $\zeta_i \subseteq \mathcal{A}$ such that $\Upsilon(\zeta_i) = 1$, at the current interval t, of her choice.*
    - Corruption : *$\mathcal{A}$ can request the secret key sk_i of the honest member U_i of her choice.*
    - Revocation : *$\mathcal{A}$ can request the revocation of arbitrary member U_i of her choice. $\mathcal{C}$ responds with the updated revocation list RL_t.*

[1]Similar to `SndToU` [20] oracle.

- `Challenge` : *At some period* $t^* \in \{1, ..., \mathbb{T}\}$, $\mathcal{A}$ *outputs* M^*, Υ^*, *and uncorrupted users* $\mathrm{U}_{i_0}, \mathrm{U}_{i_1}$ (*not queried in* `Corruption`, `Join` *and* `Revocation` *queries so far*) *and,* $\zeta : \zeta \subseteq \mathcal{A}_{i_0}, \zeta \subseteq \mathcal{A}_{i_1}$ *and* $\Upsilon(\zeta)$[1]$= 1$. $\mathcal{C}$ *randomly selects* $\kappa \in_R \{0, 1\}$ *and responds with a group signature* σ^* *on* M^* *of group member* U_{i_κ}.
- `Phase 2` : $\mathcal{A}$ *can make queries similar to Phase* 1. *However* $\mathcal{A}$ *cannot make* `Corruption` *query on* U_{i_0} *and* U_{i_1} *at any time but can make* `Revocation` *query after the time interval* t^*.

- ***Output:*** *Finally,* $\mathcal{A}$ *outputs a bit* κ', *and wins if* $\kappa' = \kappa$.

The advantage of $\mathcal{A}$ *is defined as* $Adv^{BU-user-anon}(\mathcal{A}) = |Pr(\kappa = \kappa') - \frac{1}{2}|$.

Thus there should not exist any PPT adversary to link a group signature to a user with non negligible probability.

ABGS scheme preserves traceability if it is possible to trace the valid group signature to its signer with the help of group opening key.

Definition 4.2.5 (`Traceability`) *We say that the VLR-ABGS scheme preserves* traceability *if for all PPT* $\mathcal{A}$, *the probability that* $\mathcal{A}$ *wins the following game is negligible.*

- ***Setup:*** *The challenger* $\mathcal{C}$ *runs* $(gpk, ik, \vec{grt}) \leftarrow$ `KeyGen`$(params)$. $\mathcal{C}$ *gives* gpk and $\vec{grt}$ *to* $\mathcal{A}$.
- ***Queries:*** $\mathcal{A}$ *can issue the* `Signing`, `Corruption` *and* `Join` *queries. All queries are the same as in the* BU-user anonymity *game, except the* `Join` *query.*
  - `Join`[2]*: Here* $\mathcal{A}$ *requests* $\mathcal{C}$ *(to run), the Join procedure for corrupted member* U_i.

[1]Here ζ can be different for $\mathrm{U}_{i_0}, \mathrm{U}_{i_1}$ but we are concerned about user anonymity rather than attribute anonymity

[2]similar to `SndToI` oracle in [20]

- ***Output:*** *$\mathcal{A}$ outputs a message M^*, a predicate Υ^*, a group signature σ^*, a interval number t^* and a set of revocation tokens $RL_{t^*}^*$.*

$\mathcal{A}$ wins if
(1) $\texttt{Verify}(params, gpk, t^*, RL_{t^*}^*, M^*, \Upsilon^*, \sigma^*) = 1$
(2) σ^* traces (using the tracing algorithm above) to wrong user outside $RL_{t^*}^*$.
The advantage of $\mathcal{A}$ is defined as the probability that $\mathcal{A}$ wins.

Thus it should be impossible to produce an untraceable valid group signature by any PPT adversary.

ABGS scheme preserves non-frameability if it is difficult to produce a valid group signature which traces back to a group member who has not produce it, even with the help of group manager's secret key.

Definition 4.2.6 (`Non-frameability`) *We say that the VLR-ABGS scheme preserves* non-frameability *if for all PPT $\mathcal{A}$, the probability that $\mathcal{A}$ wins the following game is negligible.*

- ***Setup:*** *The challenger $\mathcal{C}$ runs $(gpk, ik, \vec{grt}) \leftarrow \texttt{KeyGen}(params)$. $\mathcal{C}$ gives gpk, ik* and *$\vec{grt}$ to $\mathcal{A}$.*
- ***Queries:*** *$\mathcal{A}$ can issue the* `Join`, `Signing` *and* `Corruption` *queries. All queries are the same as in the BU-user anonymity game.*
- ***Output:*** *Finally, $\mathcal{A}$ outputs a message M^*, an honest member U_{i^*}, a predicate Υ^*, a group signature σ^*, a period number t^* and a set of revocation tokens $RL_{t^*}^*$.*

$\mathcal{A}$ wins if
(1) $\texttt{Verify}(params, gpk, t^*, RL_{t^*}^*, M^*, \Upsilon^*, \sigma^*) = 1$,
(2) σ^* traces to an honest member U_{i^*},
(3) $\mathcal{A}$ has not obtained sk_{i^*} in `Corruption` *queries on U_{i^*}.*
The advantage of $\mathcal{A}$ is defined as the probability that $\mathcal{A}$ wins.

Thus even the group manager should not be able to forge a group signature which trace back to a honest member.

ABGS scheme preserves collusion resistance of attribute certificates if it is computationally hard to combine attribute certificates of different group members to satisfy the predicate and produce a valid group signature.

Definition 4.2.7 (`Attribute unforgeability`) *We say that the VLR-ABGS scheme preserves* attribute unforgeability *if for all PPT $\mathcal{A}$, the probability that $\mathcal{A}$ wins the following game is negligible.*

- ***Setup:*** *The challenger $\mathcal{C}$ runs $(gpk, ik, \vec{grt}) \leftarrow$ `KeyGen`$(params)$. $\mathcal{C}$ gives gpk to $\mathcal{A}$.*
- ***Queries:*** *$\mathcal{A}$ can issue the* `Join`, `Signing`, `Corruption` *and* `Revocation` *queries. All queries are the same as in the* BU-user anonymity *game, except the* `Join` *query.*
  - `Join`*: Here $\mathcal{A}$ requests $\mathcal{C}$ (to run), the Join procedure for corrupted member* U_i.
- ***Output:*** *$\mathcal{A}$ outputs a message M^*, a predicate Υ^*, a group signature σ^*, a period number t^* and a set of revocation tokens $RL_{t^*}^*$.*

$\mathcal{A}$ wins if
(1) `Verify`$(params, gpk, t, RL_t, M^*, \Upsilon^*, \sigma^*) = 1$,
(2) traces to i and
(3) $\nexists \zeta \in \mathcal{A}_i : \Upsilon(\zeta) = 1$.
The advantage of $\mathcal{A}$ is defined as the probability that $\mathcal{A}$ wins.

Thus it should be impossible for any PPT adversary to satisfy the predicate with invalid set of attributes.

ABGS scheme preserves collusion resistance of attribute certificates if it is computationally hard for group members to collude by pooling their attribute certificates to satisfy the predicate and to produce a valid group signature.

Definition 4.2.8 (`Collusion resistance of attributes`) *We say that the ABGS scheme preserves collusion resistance of attributes if for all PPT $\mathcal{A}$, the probability that $\mathcal{A}$ wins the following game is negligible.*

- ***Setup:*** *The challenger $\mathcal{C}$ runs $(gpk, ik, \vec{grt}) \leftarrow$ `KeyGen`$(params)$. $\mathcal{C}$ gives gpk to $\mathcal{A}$.*
- ***Queries:*** *$\mathcal{A}$ can issue the* `Join`, `Signing`, `Corruption` *and* `Revocation` *queries. All queries are the same as in the BU-user anonymity game, except the* `Join` *query.*
  - `Join`*: Here $\mathcal{A}$ requests $\mathcal{C}$ (to run), the Join procedure for corrupted member* U_i.
- ***Output:*** *$\mathcal{A}$ outputs a message M^*, a predicate Υ^*, a group signature σ^*, a period number t^* and a set of revocation tokens $RL^*_{t^*}$.*

$\mathcal{A}$ wins if
(1) `Verify`$(params, gpk, t, RL_t, M^*, \Upsilon^*, \sigma^*) = 1$, *and*
(2) $\mathcal{A}$ has obtained $sk_{i_1}, ..., sk_{i_k} : \Upsilon^*(\cup_{j=1}^{k} \mathcal{A}_{i_j}) = 1$ and $\Upsilon^*(\mathcal{A}_{i_j}) \neq 1$ for $j = 1, ..., k$.
The advantage of $\mathcal{A}$ is defined as the probability that $\mathcal{A}$ wins.

Thus the users with invalid set of attributes each, cannot collude with each other to pool a valid attribute set for producing a valid group signature.

4.3 Construction

A construction of VLR-ABGS scheme with backward unlinkability and attribute anonymity features is presented in this section. We prove that the proposed scheme

is secure under random oracle model with DL, q-SDH, DLIN and KEA1 assumptions. To build our scheme we use VLR-GS scheme of Nakanishi et al. [91] as the base scheme. We use the membership certificate format of [48] to make the scheme *non-frameable* i.e. even the group manager cannot forge signature of a trusted member. We use the bottom-up approach technique, introduced by Emura et al. in [52], for generating the public values of the access tree representing a predicate. We device an `BuildTree-Validity` algorithm which gives provision to publicly verify the correctness of the generated public values of the predicate and with this we reduce the trust on group manager in producing public keys of the predicates. We device an idea to achieve attribute anonymity. The proposed ABGS scheme achieves the better efficiency than the other schemes [52; 72] in terms of signing cost, verification cost , secret key length and signature length. When compare to VLR-GS scheme in standard model by Libert et al. [82] our scheme achieves an additional feature namely, non-frameability, and has shorter signature length. We emphasize that our scheme achieves constant signature size, independent of the number of attributes, when compare to other ABGS scheme in the literature.

- `Setup`(1^k): It generates the system parameters : $params = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \psi, \mathbb{T}, \mathcal{H}_0, \mathcal{H}, Att)$; where
  - (i) $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are the cyclic groups of prime order p, where $2^{k-1} < p < 2^k$, $\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$ is an isomorphism, $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a bilinear map.
  - (ii) $\mathcal{H}_0 : \{0,1\}^* \rightarrow \mathbb{G}_2$ and $\mathcal{H} : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$ are the hash functions.
  - (iii) $\mathbb{T} = O(k)$ is the total number of time periods.
  - (iv) $Att = \{\text{att}_1, ..., \text{att}_m\}$, for $m = O(k)$[1] is the universal set of attributes.
- `KeyGen`$(params)$: It takes an input system parameters $params$ and outputs a group public key gpk, an issuing key ik and the vector $\vec{grt}$ of revocation tokens.
  - (i) Select the generators $g_1, h \in \mathbb{G}_1$ and $g_2, \hat{h} \in \mathbb{G}_2 : g_1 = \psi(g_2)$ and $h = \psi(\hat{h})$.
  - (ii) Select $\gamma \in_R \mathbb{Z}_p^*$, and computes $w = g_2^{\gamma}$.

[1]For $m = 1$ it becomes a group signature scheme.

(iii) For each $\text{att}_j \in Att$, choose $s_j \in_R \mathbb{Z}_p^*$, sets $S = \{s_j\}_{\text{att}_j \in Att}$, and computes $\hat{h}_{\text{att}_j} = \hat{h}^{s_j}, \forall \text{att}_j \in Att$.

(iv) Select the vector of group elements, $(\hat{h}_1, ..., \hat{h}_{\mathbb{T}})^1 \in_R \mathbb{G}_2^{\mathbb{T}}$.

(v) Initialize the vector of revocation tokens, $\vec{grt}$, registration table, $\vec{reg}$ and the revocation lists, $\{RL_t\}_{t=1}^{\mathbb{T}}$ as empty.

(vi) Outputs an issuing key ik and a group public key gpk.

$$gpk = (g_1, h, g_2, \hat{h}, w, (\hat{h}_1, ..., \hat{h}_{\mathbb{T}}), \{\hat{h}_{\text{att}_j}\}_{\text{att}_j \in Att}), ik = (\gamma, S), \vec{grt}$$

- `BuildTree`$(params, gpk, ik, \Upsilon)$: It generates a public values for the predicate Υ.

  (i) Let T_Υ be the tree that represents the predicate Υ.

  (ii) Get extension tree $T^{\text{ext}} \leftarrow$ `AddDummyNode`(T_Υ).

  (iii) Get secret values for each dummy node and the secret value of root of T^{ext} using $(\{s_{d_j}\}_{d_j \in D_{T_\Upsilon}}, s_T) \leftarrow$ `AssignedVaule`(S, T^{ext}).

  (iv) Output the predicate public values, $\mathcal{T}_\Upsilon = (\{s_{d_j}\}_{d_j \in D_{T_\Upsilon}}, \hat{h}_T = \hat{h}^{s_T}, T^{\text{ext}})$.

  Here we relax the trust on the group manager in computing public values for the predicate. We give another algorithm, `BuildTree-Validity`, which allows a user to verify the validity of the public values of the predicate with the help of group public key gpk.

- `BuildTree-Validity`$(params, gpk, \mathcal{T}_\Upsilon)$:

  (i) Randomly choose an attribute set, $Leaves \subseteq Att : \Upsilon(Leaves) = 1$ And gets the corresponding $\Delta_{\text{att}_j}(\forall \text{att}_j \in Leaves)$, and $\Delta_{d_j}(\forall d_j \in D_{T_\Upsilon}^{Leaves})$ by running `MakeSimplifiedTree`$(Leaves, T^{ext})$.

  (ii) Compute $\hat{h}_{root} = \prod_{\text{att}_j \in Leaves} \hat{h}_{\text{att}_j}^{\Delta_{\text{att}_j}} \times \prod_{d_j \in D_{T_\Upsilon}^{Leaves}} \hat{h}_{d_j}^{\Delta_{d_j}}$
  $= \prod_{\text{att}_j \in Leaves} \hat{h}^{s_j \Delta_{\text{att}_j}} \times \prod_{d_j \in D_{T_\Upsilon}^{Leaves}} \hat{h}^{s_{d_j} \Delta_{d_j}}$
  $= \hat{h}^{\sum_{\text{att}_j \in Leaves} \Delta_{\text{att}_j} s_j + \sum_{d_j \in D_{T_\Upsilon}^{Leaves}} \Delta_{d_j} s_{d_j}} = \hat{h}^{s_T}$ from (2.6).

[1]It has provision to increase the time intervals by adding new group elements $\hat{h}_{\mathbb{T}+1}, ... \in_R \mathbb{G}_2$ and update the gpk without altering the other keys.

(iii) Verify whether $\hat{h}_{root} \stackrel{?}{=} \hat{h}_T$. If not then $\mathcal{T}_\Upsilon$ is the invalid public values of the predicate Υ.

- `Join`$(< params, gpk, t, ik, upk_i, \mathcal{A}_i >, < params, gpk, t, upk_i, usk_i >)$:
  This is the modified version of the `Join` protocol in [48; 52], which also includes revocation tokens. Let `NIZK` be a non-interactive zero-knowledge proof, `SPK` be the Signature Proof of Knowledge, and `Ext-Commit` be an extractable commitment scheme, the trapdoor of the commitment will not be known to anybody except to our simulator in the *traceability* proof. As a result of this protocol, U_i gets sk_i[1] $= ((A_i, x_i, y_i), \{T_{i,j}\}_{\mathrm{att}_j \in \mathcal{A}_i})$, where (A_i, x_i, y_i) is a membership certificate, $\{T_{i,j}\}_{\mathrm{att}_j \in \mathcal{A}_i}$ is the set of attribute certificates and $\mathcal{A}_i \subseteq Att$ is the set of U_i's attributes. And `GM` ends with the updated $\vec{grt}$ and $\vec{reg}$. The protocol begins as follows,

  (i) U_i picks $y_i \in_R \mathbb{Z}_p^*$, computes $c_i =$ `Ext-Commit`$(y_i), F_i = h^{y_i}$, $\pi_1 =$ `NIZK`$\{y_i : F_i = h^{y_i} \wedge c_i =$ `Ext-Commit`$(y_i)\}$ and, sends F_i, c_i and π_1 to `GM`.

  (ii) `GM` checks π_1 using the `NIZK` method. If π_1 is not valid, then abort. `GM` selects $x_i \in_R \mathbb{Z}_p^*$ and computes $A_i = (g_1 F_i)^{1/(\gamma + x_i)}, B_i = e(g_1 F_i, g_2)/e(A_i, w)$, $D_i = e(A_i, g_2), T_{i,j} = A_i^{s_j} (\forall \mathrm{att}_j \in \mathcal{A}_i)$ and $\pi_2 =$ `NIZK`$\{x_i, \{s_j\}_{(\mathrm{att}_j \in \mathcal{A}_i)} : B_i = D_i^{x_i} \wedge T_{i,j} = A_i^{s_j} (\forall \mathrm{att}_j \in \mathcal{A}_i) \wedge \hat{h}_{\mathrm{att}_j} = \hat{h}^{s_j} (\forall \mathrm{att}_j \in \mathcal{A}_i) \wedge e(T_{i,j}, \hat{h}) = e(A_i, \hat{h}_{\mathrm{att}_j}) (\forall \mathrm{att}_j \in \mathcal{A}_i)\}$ and, sends $A_i, B_i, D_i, \{T_{i,j}\}_{\mathrm{att}_j \in \mathcal{A}_i}$ and π_2 to U_i.

  (iii) U_i checks π_2. If π_2 is not valid, then abort. U_i makes $S_{i,A_i} = DSig_{usk_i}(A_i)$ and sends to `GM`.

  (iv) `GM` verifies S_{i,A_i} with respect to upk_i and A_i. If S_{i,A_i} is valid, then `GM` appends the tuple $(i, upk_i, A_i, x_i, F_i, S_{i,A_i})$ to $\vec{reg}$ and sends x_i to U_i. `GM` computes the revocation tokens, $\{B_{ij} = \psi(\hat{h}_j)^{x_i}\}_{j=t}^{\mathbb{T}}$ and it is added to the vector of revocation tokens $\vec{grt}$ but not to the $\{RL_j\}_{j=t}^{\mathbb{T}}$ lists.

  (v) U_i checks the relation $e(A_i, g_2)^{x_i} e(A_i, w) e(h, g_2)^{-y_i} \stackrel{?}{=} e(g_1, g_2)$ to verify whether $A_i^{(x_i+\gamma)} = g_1 h^{y_i}$.

[1]Note that $T_{i,j}$ values can be stored in a public repository, so the size of $sk_i's$ can be reduced to three elements, i.e. $sk_i = (A_i, x_i, y_i)$.

GM chooses $s_{m+1} \in \mathbb{Z}_p^*$, and computes $g_{\text{att}_{m+1}} = g_2^{s_{m+1}}$ when a new attribute att_{m+1} is added. Let U_i be issued $T_{i,m+1}$. Then GM computes $T_{i,m+1} = A_i^{s_{m+1}}$ and $\pi_3 = \text{NIZK}\{s_{m+1} : T_{i,m+1} = A_i^{s_{m+1}} \wedge \hat{h}_{\text{att}_{m+1}} = \hat{h}^{s_{m+1}} \wedge e(T_{i,m+1}, \hat{h}) = e(A_i, \hat{h}_{\text{att}_{m+1}})\}$, sends $T_{i,m+1}$ and π_3 to U_i, and opens $\hat{h}_{\text{att}_{m+1}}$.

- $\text{Sign}(params, gpk, t, sk_i, \zeta, M, \Upsilon)$: It generates a group signature σ on message $M \in \{0,1\}^*$ for the time interval t with the user private key sk_i who satisfy the predicate Υ with his subset of attributes $\zeta \subseteq \mathcal{A}_{\text{id}} : \Upsilon(\zeta) = 1$.

  (i) Get the public values of Υ from the public repository[1].

  (ii) Runs $\text{MakeSimplifiedTree}(\zeta, T^{ext})$ and get the corresponding $\Delta_{\text{att}_j}(\forall \text{att}_j \in \zeta)$, and $\Delta_{d_j}(\forall d_j \in D^{\zeta}_{T_\Upsilon})$.

  (iii) Compute $s_{T_2} = \Sigma_{d_j \in D^{\zeta}_{T_\Upsilon}} \Delta_{d_j} s_{d_j}$. Let $s_{T_1} = \Sigma_{\text{att}_j \in \zeta} \Delta_{\text{att}_j} s_j$. Note that from (2.6) $s_{T_1} + s_{T_2} = s_T$.

  (iv) Select $r \in_R \mathbb{Z}_p^*$, and computes $\hat{f} = \mathcal{H}_0(gpk||M||r) \in \mathbb{G}_2$.
  Then compute $f = \psi(\hat{f})$.

  (v) Select $\alpha, \beta \in_R \mathbb{Z}_p^*$, and compute

$$C_1 = A_i h^{\alpha}, C_2 = f^{\beta + x_i}, C_3 = \psi(\hat{h}_t)^{\beta},$$

$$C_4 \ = \Pi_{\text{att}_j \in \zeta} T_{i,j}^{\Delta_{\text{att}_j}} A_i^{s_{T_2}} \psi(\hat{h}_T)^{\alpha} = A_i^{s_{T_1} + s_{T_2}} h^{s_T \alpha} = C_1^{s_T}$$

  (vi) Set $\tau = \alpha x_i + y_i$ and compute
  $V = \text{SPK}\{(\alpha, \beta, x_i, \tau) : \frac{e(C_1, w)}{e(g_1, g_2)} = \frac{e(h, g_2)^{\tau} e(h, w)^{\alpha}}{e(C_1, g_2)^{x_i}} \bigwedge C_2 = f^{\beta + x_i} \bigwedge C_3 = \psi(\hat{h}_t)^{\beta}\}$
  $(M) = (c, s_\alpha, s_\beta, s_{x_i}, s_\tau)$.
  The computation of SPK is given in subsequent section.

  (vii) Outputs $\sigma = (\{C_i\}_{i=1}^4, V, r) \in \mathbb{G}_1^4 \times \mathbb{Z}_p^{*6}$.

  A signer proves the knowledge of $(\alpha, \beta, x_i, \tau)$ which satisfies the above 3 relations described in SPK. The first relation captures whether a signer has a valid

[1]GM runs BuildTree algorithm to generate the public values of the predicate Υ and stores it in a public repository. Note that if the public values of the required predicate is present in the public repository then the user will not approach GM.

membership certificate issued by the `Join` algorithm or not. The relation (4.1) in the verification algorithm captures whether a signer has valid attribute certificates or not and also proves the association of the membership certificate with attribute certificates. The second and third relations are needed for revocation check in verification. Note that the signature is independent of the number of attributes, thus its length is constant.

- `Verify`$(params, gpk, t, RL_t, M, \sigma, \Upsilon)$:
  A verifier verifies the group signature σ as follows,

  (i) *Signature check:* Checks that the σ is valid, by verifying the `SPK` V and the relation

  $$e(C_1, \hat{h}_T) = e(C_4, \hat{h}) \tag{4.1}$$

  The verification of `SPK` V is given in subsequent section.

  (ii) *Revocation check:* Check that the signer is not revoked at the interval t by checking

  $$e(C_2, \hat{h}_t) \neq e(B_{it}C_3, \hat{f}), \forall grt[i][t] \in RL_t. \tag{4.2}$$

  Consider $e(C_2, \hat{h}_t) = e(f^{\beta+x_i}, \hat{h}_t) = e(f, \hat{h}_t)^{\beta+x_i}$. On the other hand, $e(B_{it}C_3, \hat{f}) = e(\psi(\hat{h}_t)^{x_i}\psi(\hat{h}_t)^{\beta}, \hat{f}) = e(\psi(\hat{h}_t), \hat{f})^{\beta+x_i}$. Thus, the revoked user's signature can be detected.

Note that a user can be revoked for a certain time period, after which again he can join the group and generate the valid group signatures with same key. That is, if `GM` wants to revoke a user for a period from time interval t to time interval t',then he will add $t' - t + 1$ revocation tokens from interval t to t' to the respective revocation lists $RL_t, RL_{t+1}, ..., RL_{t'}$. Thus *suspension* of a member is possible.

4.3.1 Computation of `SPK`

In the signature algorithm U_i computes V as follows,

(i) U_i chooses blinding values $r_\alpha, r_\beta, r_{x_i}, r_\tau \in_R \mathbb{Z}_p^*$.

(ii) U_i computes $R_1 = \frac{e(h,g_2)^{r_\tau} e(h,w)^{r_\alpha}}{e(C_1,g_2)^{r_{x_i}}}$, $R_2 = f^{r_\beta + r_{x_i}}$, $R_3 = \psi(\hat{h}_t)^{r_\beta}$.

(iii) U_i computes $c = \mathcal{H}(gpk, t, M, \{C_i\}_{i=1}^4, \{R_i\}_{i=1}^3)$.

(iv) U_i computes $s_\alpha = r_\alpha + c\alpha$, $s_\beta = r_\beta + c\beta$, $s_{x_i} = r_{x_i} + cx_i$ and $s_\tau = r_\tau + c\tau$.

Thus, $V = (c, s_\alpha, s_\beta, s_{x_i}, s_\tau)$.

4.3.2 Verification of SPK

In verification algorithm the verifier verifies SPK V for the group signature $\sigma = (\{C_i\}_{i=1}^4, V, r)$ as follows,

(i) The verifier computes
$\hat{f} = \mathcal{H}_0(gpk||M||r), f = \psi(\hat{f})$,
$\widetilde{R_1} = \frac{e(h,g_2)^{s_\tau} e(h,w)^{s_\alpha}}{e(C_1,g_2)^{s_{x_i}}} \left(\frac{e(g_1,g_2)}{e(C_1,w)}\right)^c$,
$\widetilde{R_2} = f^{s_\beta + s_{x_i}} \left(\frac{1}{C_2}\right)^c$,
$\widetilde{R_3} = \psi(\hat{h}_t)^{s_\beta} \left(\frac{1}{C_3}\right)^c$.

(ii) The verifier checks whether
$c \stackrel{?}{=} \mathcal{H}(gpk, t, M, \{C_i\}_{i=1}^4, \{\widetilde{R_i}\}_{i=1}^3)$.

4.4 Security Analysis

Theorem 4.4.1 *The proposed ABGS scheme is correct.*

Proof SPK ensures that the scheme is correct. □

Theorem 4.4.2 *The proposed ABGS scheme preserves attribute anonymity.*

Proof According to the definition of attribute anonymity it is sufficient to show that for any predicate Υ and for any subset of attributes $\mathcal{A}_i : \exists \zeta_1, \zeta_2 \subseteq \mathcal{A}_i$ that satisfy the predicate i.e., $\Upsilon(\zeta_1) = \Upsilon(\zeta_2) = 1$, the output of $\texttt{Sign}(params, gpk, t, sk_i, \zeta_1, M, \mathcal{T}_\Upsilon)$ is indistinguishable from the output of $\texttt{Sign}(params, gpk, t, sk_i, \zeta_2, M, \mathcal{T}_\Upsilon)$, subject to the constraint that they passes verification algorithm.

For any group signature $\sigma = (\{C_i\}_{i=1}^4, V, r)$, the attribute certificates are hidden and used in C_4 computation and it is easy to observe that for any two given group signatures the value $C_4 = (C_1)^{s_T}$ will not distinguish among themselves, since both are identical because of same s_T value. Thus it will not reveal the underlying subset of attributes, but only it proves that it satisfies the predicate. Thus the proposed scheme preserve attribute anonymity. □

Theorem 4.4.3 *The proposed ABGS scheme satisfies the BU-user anonymity in the random oracle model under DLIN assumption.*

Proof Lemma 4.4.4 implies the Theorem 4.4.3. □

Lemma 4.4.4 *Suppose an adversary $\mathcal{A}$ breaks the BU-user anonymity of the proposed scheme with the advantage ϵ. Then, we can construct an algorithm $\mathcal{B}$ that breaks the DLIN assumption on $\mathbb{G}_2$ with the advantage $\left(\frac{1}{n\mathbb{T}} - \frac{q_S+q_H}{p}\right)\epsilon$, where n is the total number of members, $\mathbb{T}$ is the number of time intervals, q_S is the total number of signature queries, q_H is the total number of hash queries and p is large prime.*

Proof The input of $\mathcal{B}$ is an instance of DLIN problem, $(\hat{u}, \hat{v}, \hat{h}, \hat{U} = \hat{u}^a, \hat{V} = \hat{v}^b, \hat{Z}) \in \mathbb{G}_2^6$, where $a, b \in_R \mathbb{Z}_p^*$ and either $\hat{Z} = \hat{h}^{a+b}$ or $\hat{Z} = \hat{h}^c$ for $c \in_R \mathbb{Z}_p^*$. $\mathcal{B}$ decides which $\hat{Z}$ it is given by communicating with $\mathcal{A}$. Let the challenged group signature be $\sigma^* = (\{C_i^*\}_{i=1}^4, V^*, r^*)$.

Setup. Algorithm $\mathcal{B}$ simulates an VLR-ABGS scheme as follows,

(i) $\mathcal{B}$ generates system parameters, *params* and set $\mathcal{H}_0$ and $\mathcal{H}$ as hash oracles.

(ii) $\mathcal{B}$ selects $i^\star \in_R [1, n]$ and $t^\star \in_R [1, \mathbb{T}]$. $\mathcal{B}$ sets $g_2 = \hat{u}$, and $g_1 = \psi(g_2)$.

(iii) $\mathcal{B}$ selects $\hat{h} \in_R \mathbb{G}_2$ and computes $h = \psi(\hat{h})$. Also chooses $\gamma, s_1, ..., s_m, r_1, ..., r_{\mathbb{T}} \in_R \mathbb{Z}_p^*$. And computes $w = g_2^{\gamma}, \{\hat{h}_{att_i} = \hat{h}^{s_i}\}_{i=1}^{m}$ and $\hat{h}_t = g_2^{r_t}$ for all $t \in [1, \mathbb{T}]$ except $t^\star$. $\mathcal{B}$ sets $\hat{h}_{t^\star} = \hat{v}$.

(iv) $\mathcal{B}$ sets a group public key, $gpk = (g_1, h, g_2, \hat{h}, w, \{\hat{h}_t\}_{t=1}^{\mathbb{T}}, \{\hat{h}_{\mathrm{att}_j}\}_{\mathrm{att}_j \in Att})$.

(v) For the user $\mathrm{U}_{i^\star}$, $\mathcal{B}$ define $x_{i^\star} = a$ and $A_{i^\star} = (g_1 h^{y_{i^\star}})^{1/(\gamma+a)}$, which are unknown for $\mathcal{B}$. $\mathcal{B}$ sets $B_{i^\star,j} = \psi((\hat{u}^a)r_j) = \psi(g_2^{ar_j}) = \psi(\hat{h}_j^a)$ except for $j = t^\star$. $B_{i^\star,t^\star} = \psi(\hat{v}^a) = \psi(\hat{h}_{t^\star})^a)$ is also unknown to $\mathcal{B}$.

(vi) $\mathcal{B}$ gives gpk to $\mathcal{A}$.

Hash queries. At any time, $\mathcal{A}$ can query the hash function $\mathcal{H}_0$ or $\mathcal{H}$. $\mathcal{B}$ responds with random values with consistency.

Phase 1 $\mathcal{A}$ requests the queries as given in `BU-user anonymity` game. $\mathcal{B}$ answers to these queries as the real settings of VLR-ABGS scheme, since $\mathcal{B}$ knows all the values. If $i = i^\star$, then $\mathcal{B}$ responds the queries as follows.

- **Signing queries:** $\mathcal{B}$ computes a simulated group signature of $i^\star$, depending on t as follows,
  Case of $t \neq t^\star$:

  (i) $\mathcal{B}$ selects $r, \beta, \delta \in_R \mathbb{Z}_p^*$ and sets $\hat{f} = \hat{h}_t^{\delta}, f = \psi(\hat{f})$.

  (ii) $\mathcal{B}$ computes $C_2 = f^{\beta} B_{i^\star t}^{\delta} = f^{\beta} \psi(\hat{h}_t^{x_{i^\star}})^{\delta} = f^{\beta + x_{i^\star}}$, and $C_3 = \psi(\hat{h}_t)^{\beta}$.

  (iii) $\mathcal{B}$ selects $C_1 \in_R \mathbb{G}_1$ and set $C_4 = C_1^{s_T}$. $\mathcal{B}$ gets s_T from the `BuildTree` procedure.

  (iv) $\mathcal{B}$ computes a simulated `SPK` V by selecting a random $c, s_\alpha, s_\beta, s_{x_i}, s_\tau \in_R \mathbb{Z}_p^*$ and computing the corresponding (R_1, R_2, R_3) by following the procedure, *Verification of* `SPK`, given in appendix B, and patching the hash oracle at $\mathcal{H}(gpk, t, M, \{C_i\}_{i=1}^{4}, \{R_i\}_{i=1}^{3})$ to c. If this backpatch fails then $\mathcal{B}$ outputs a random guess and aborts. Furthermore, $\mathcal{B}$ defines $\mathcal{H}_0(gpk, M, r) = \hat{f}$. If this backpatch fails then $\mathcal{B}$ outputs a random guess and aborts.

  Case of $t = t^\star$:

(i) $\mathcal{B}$ selects $r, \beta, \delta \in_R \mathbb{Z}_p^*$ and sets $\hat{f} = \hat{u}^\delta, f = \psi(\hat{f})$.

(ii) $\mathcal{B}$ computes $C_2 = \psi(\hat{u}^\beta \hat{U})^\delta = \psi(\hat{f}^{\beta + x_{i^\star}})$, and $C_3 = \psi(\hat{h}_{t^\star})^\beta$.

(iii) From here it is the same as in case of $t \neq t^\star$.

Then $\mathcal{B}$ responds signature $\sigma = (\{C_i\}_{i=1}^4, V, r)$ to $\mathcal{A}$. Note that it has the same distribution as the real due to the perfect zero-knowledge-ness of `SPK`.

- **Corruption queries:** $\mathcal{B}$ outputs a random guess and aborts.
- **Join queries:** $\mathcal{B}$ outputs a random guess and aborts.
- **Revocation queries:** If $t \neq t^\star$, $\mathcal{B}$ responds $B_{i^\star t}$. Otherwise outputs a random guess and aborts.

Challenge $\mathcal{A}$ outputs a message M^*, the current time interval t^*, a predicate Υ^*, an attribute set ζ and two uncorrupted members $\mathrm{U}_{i_0}, \mathrm{U}_{i_1}$, such that $\Upsilon^*(\zeta) = 1, \zeta \subseteq \mathcal{A}_{i_0}$ and $\zeta \subseteq \mathcal{A}_{i_1}$, to be challenged. $\mathcal{T}_{\Upsilon^*}$ is a public values of the predicate Υ. If $t^* \neq t^\star$ then $\mathcal{B}$ outputs a random guess and aborts, its probability is $1/\mathbb{T}$. $\mathcal{B}$ picks $\kappa \in_R \{0, 1\}$. Then, if $i_\kappa \neq i^\star$, $\mathcal{B}$ outputs a random guess and aborts, its probability is $1/n$. Otherwise $\mathcal{B}$ try to simulate the challenged signature σ^* from A_{i_κ} as follows,

- $\mathcal{B}$ selects $r^* \in_R \mathbb{Z}_p^*$, regards b as β which is unknown, and sets $\hat{f} = \hat{h}, f = \psi(\hat{f})$. $\mathcal{B}$ gets corresponding s_T from the `BuildTree` procedure.
- $\mathcal{B}$ selects $C_1 \in_R \mathbb{G}_1$, and sets $C_2 = \psi(\hat{Z}), C_3 = \psi(\hat{V}) = \psi(\hat{h}_{t^*})^\beta)$ and $C_4 = C_1^{s_T}$. Note that if $\hat{Z} = \hat{h}^{a+b}$ then $C_2 = \psi(\hat{h}^{a+b}) = f^{\beta + x_{i_\kappa}}$.
- $\mathcal{B}$ computes a simulated `SPK` V by selecting a random $c^*, s_\alpha^*, s_\beta^*, s_{x_i}^*, s_\tau^* \in_R \mathbb{Z}_p^*$ and computing the corresponding (R_1, R_2, R_3) by following the procedure, verification of `SPK`, given in appendix B, and patching the hash oracle at $\mathcal{H}(gpk, t^*, M^*, \{C_i^*\}_{i=1}^4, \{R_i\}_{i=1}^3)$ to c^*. If this backpatch fails then $\mathcal{B}$ outputs a random guess and aborts. Furthermore, $\mathcal{B}$ defines $\mathcal{H}_0(gpk||M^*||r^*) = \hat{f}$. If this backpatch fails then $\mathcal{B}$ outputs a random guess and aborts. The simulation failure probability is less than $(q_H + q_S)/p$.

In case of failure, $\mathcal{B}$ outputs a random bit and aborts, otherwise, $(\{C_i^*\}_{i=1}^4, V^*, r^*)$ is the signature on M^* given back to $\mathcal{A}$.

Phase 2. $\mathcal{A}$ requests the queries as given in `BU-user anonymity` game and $\mathcal{B}$ answers it similar to the phase 1.

Output. $\mathcal{A}$ outputs its guess $\kappa' \in \{0,1\}$ with advantage ϵ. Our algorithm $\mathcal{B}$ outputs 0 if $\kappa = \kappa'$ (indicating that $\hat{Z} = \hat{h}^{a+b}$); otherwise $\mathcal{B}$ outputs 1 (indicating that $\hat{Z}$ is random in $\mathbb{G}_2$). Thus it has advantage of $\left(\frac{1}{n^{\mathbb{T}}} - \frac{q_S+q_H}{p}\right)\epsilon$ in distinguishing DLIN tuples. □

Theorem 4.4.5 *The proposed scheme preserves the attribute unforgeability under KEA1 and DL assumptions.*

Proof Lemma 4.4.6 implies the Theorem 4.4.5. □

Lemma 4.4.6 *Under the DL and KEA1 assumptions there exists no PPT adversary $\mathcal{A}$ which passes verification with forged attributes with non negligible probability.*

Proof The input to the simulator $\mathcal{B}$ is an instance of the DL problem, $(g, g') \in \mathbb{G}_2^2$. Let $\xi = log_g g'$.

Setup: According to the VLR-ABGS scheme setup $\mathcal{B}$ generates the system parameters, *params* and sets $g_2 = g'$, $\hat{h} = g$, $g_1 = \psi(g_2)$ and $h = \psi(\hat{h})$, and generate the remaining parameters, $gpk = (g_1, h, g_2, \hat{h}, w, \{\hat{h}_t\}_{t=1}^{\mathbb{T}}, \{\hat{h}_{\text{att}_j}\}_{\text{att}_j \in Att}), ik = (\gamma, S)$ and $\vec{grt}$.

Queries: As $\mathcal{B}$ knows all the keys, it can answer all the queries generated by an adversary $\mathcal{A}$ according to definition of `attribute unforgeability` .

Output: Finally, $\mathcal{A}$ outputs a signature σ^* with forged attribute certificates on message M^*, a predicate Υ^* whose public values are $\mathcal{T}_{\Upsilon^*} = (\{s_{d_j}\}_{d_j \in D_{T_{\Upsilon^*}}}, \hat{h}_T, T^{\text{ext}})$, and signer's secret key sk_{i^*} such that $\Upsilon(\mathcal{A}_{i^*}) \neq 1$. As it is a valid signature which passes verification algorithm and from (4.1) $C_4^* = (C_1^*)^{s_T}$. This can be viewed as $C_4^* = A_{i^*}^{s_T} h^{s_T \alpha} = (g_1 h^{y_{i^*}})^{\frac{s_T}{\gamma + x_{i^*}}} h^{s_T \alpha} = (g_1^{s_T})^{\frac{1}{\gamma + x_{i^*}}} X$ and $g_1^{s_T}$ can be extracted by raising the power $\gamma + x_{i^*}$, where

$$X = (h^{y_{i^*}})^{\frac{s_T}{\gamma + x_{i^*}}} h^{s_T \alpha}$$

It is like $\mathcal{B}$ is giving input $(h = \psi(g), h_T = \psi(g)^{s_T})$ to $\mathcal{A}$ and $\mathcal{A}$ implicitly returns $(\psi(g'), \psi(g')^{s_T})$. Then by KEA1 assumption, $\mathcal{B}$ can utilize the extractor $\bar{\mathcal{A}}$ to extract a value ξ. Under DL assumption it can be done with negligible probability. Thus the signature produced by the forged attribute certificates can pass verification with negligible probability.

To be more particular, we can assume that the $\mathcal{A}$ is missing atleast one attribute certificate, say $T_{i^*,j} = A_{i^*}^{s_j}$, i.e. $(\psi(g')h^{y_{i^*}})^{\frac{s_j}{\gamma+x_{i^*}}}$ is unknown to $\mathcal{A}$, but he knows $\hat{h}^{s_j} = g^{s_j}$. And $\mathcal{A}$ is producing it in forged group signature σ^*. Then similar to above from KEA1 and DL assumptions it is negligible to produce such signatures. □

Theorem 4.4.7 *The proposed scheme preserves the collusion resistance of attribute.*

Proof Lemma 4.4.8 implies the Theorem 4.4.7. □

Lemma 4.4.8 *Even if some malicious participants* $\mathrm{U}_{i_1}, ..., \mathrm{U}_{i_k} (k > 1)$ *with the set of attributes* $\zeta_{i_1}, ..., \zeta_{i_k}$ *collude, they cannot make a valid signature associated with a predicate* Υ, *where* $(\cup_{j=1}^{k} \Upsilon(\zeta_{i_j}) = 1)$ *and* $\Upsilon(\zeta_{i_j}) \neq 1$ *for* $j = 1, ..., k$ *with non-negligible probability.*

Proof Without loss of generality, we assume that U_{i_1} with ζ_{i_1} and U_{i_2} with ζ_{i_2} represent malicious participants. U_{i_1} and U_{i_2} attempt to make a valid signature associated with Υ which satisfies $\Upsilon(\zeta_{i_1} \cup \zeta_{i_2}) = 1, \Upsilon(\zeta_{i_1}) \neq 1$ and $\Upsilon(\zeta_{i_2}) \neq 1$. They can make the SPK of $(\alpha, \beta, x_0, \tau)$ satisfy the SPK relations because they have a valid membership certificate A_{i_1}. We assume that $A_{i_1}^t = A_{i_2}$, where $t \in \mathbb{Z}_p^*$. Note that the probability of $t = 1$ is negligible. Then, from equation (4.1) $C_4^* = C_1^{*s_T}$, $\Sigma_{\mathrm{att}_j \in \zeta_{i_1}} \Delta_j t_j + \Sigma_{\mathrm{att}_j \in \zeta_{i_2}} t \Delta_j s_j + \Sigma_{d_j \in D_{T_\Upsilon}^{\zeta}} \Delta_{d_j} s_{d_j} \neq s_T$ holds since $t \neq 1$. This means that they cannot use $\{T_{i_1,j}\}_{\mathrm{att}_j \in \zeta_{i_1}}$ and $\{T_{i_2,j}\}_{\mathrm{att}_j \in \zeta_{i_2}}$ simultaneously. i.e. they cannot collude the attribute certificates. □

Theorem 4.4.9 *Suppose an adversary* $\mathcal{A}$ *breaks the traceability of the proposed scheme with the advantage* ϵ. *Then, in the random oracle model,an algorithm* $\mathcal{B}$ *can be constructed that breaks the q-SDH assumption with the advantage* $\frac{1}{6}\epsilon$.

Proof Since the membership certificate format is similar to the one proposed in [48], the proof is similar to the proof given in [48; 52]. The input of simulator $\mathcal{B}$ is $(g, g', g'_1, ..., g'_q) \in \mathbb{G}_1 \times \mathbb{G}_2^{q+1}$, where $g = \psi(g'), g'_i = (g')^{\xi^i}$(for $i \in [1, q]$) and let $g'_0 = g'$. Let $q-1$ be the total number of members.
Setup: $\mathcal{B}$ simulates KeyGen as follows:

(i) $\mathcal{B}$ selects $\nu, \{x_i\}_{i=1}^{q-1}, \{y_i\}_{i=1}^{q-1}, \{s_j\}_{(\forall \text{att}_j \in Att)}, \{r_t\}_{t=1}^{\mathbb{T}} \in_R \mathbb{Z}_p^*$, where $\mathbb{T}$ is the number of time intervals.

(ii) $\mathcal{B}$ selects a target user $\mathrm{U}_{i^\star} \in \{\mathrm{U}_1, ..., \mathrm{U}_{q-1}\}$, and sets $\gamma = \xi - x_{i^\star}$. $\mathcal{B}$ computes $g_1, h, g_2, \hat{h}$ and w as follows:

- Let $f(y) = \prod_{i=1}^{q-1}(y + x_i)$. Therefore, $f(\gamma) = f(\xi - x_{i^\star}) = \prod_{i=1}^{q-1}(\xi - x_{i^\star} + x_i) = \sum_{i=0}^{q-1}(\alpha_i \xi^i)$, where $\alpha_0, ..., \alpha_{q-1} \in \mathbb{Z}_p$ are the coefficients of the polynomial $f(\gamma)$, are computable.
- Let $f_i(y) = f(y)/(y + x_i) = \prod_{j=1, j\neq i}^{q-1}(y + x_j)$. Thus, $f_i(\gamma) = f_i(\xi - x_{i^\star}) = \prod_{j=1, j\neq i}^{q-1}(\xi - x_{i^\star} + x_j) = \sum_{j=0}^{q-2}(\beta_j \xi^j)$, where $\beta_0, ..., \beta_{q-2}$ are the coefficients of the polynomial $f_i(\gamma)$, are computable.
- Note that, $g^{f_i(\gamma)} = (g^{f(\gamma)})^{\frac{1}{x_i + \gamma}}$ and $f_{i^\star}(\gamma) = f(\gamma)/(\xi - x_{i^\star} + x_{i^\star}) = f(\gamma)/\xi$.
- Set $g_2 = (g')^{\nu f(\gamma)}/(g')^{y_{i^\star} f_{i^\star}(\gamma)} = \prod_{i=0}^{q-1}(g'_i)^{\nu \alpha_i} / \prod_{j=0}^{q-2}(g'_j)^{y_{i^\star}\beta_j}$
- $\hat{h} = g'^{f_{i^\star}(\gamma)} = \prod_{j=0}^{q-2}(g'_j)^{\beta_j}$
- $h = \psi(\hat{h})$.
- $g_1 = \psi(g_2) = h^{\nu\xi}/h^{y_{i^\star}}$
- $w = \left\{\prod_{i=0}^{q-1}(g'_{i+1})^{\alpha_i \nu} / \prod_{j=0}^{q-2}(g'_{j+1})^{\beta_j y_{i^\star}}\right\} / g_2^{x_{i^\star}}$
  $= \left\{(g')^{\nu\xi f(\gamma)}/(g')^{y_{i^\star}\xi f_{i^\star}(\gamma)}\right\} / g_2^{x_{i^\star}}$
  $= g_2^{\xi - x_{i^\star}} = g_2^{\gamma}$

Thus $\mathcal{B}$ can compute these values by using the $q-$SDH input instances.

(iii) $\mathcal{B}$ computes $\{\hat{h}_t = \hat{h}^{r_t}\}_{t=1}^{\mathbb{T}}, grt[i][t] = \hat{h}_t^{x_i}, \forall i \in \{1, ..., n\}, t \in \{1, ..., \mathbb{T}\}$ and other parameters as the real settings.

(iv) $\mathcal{B}$ makes $params = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \psi, \mathcal{H}_0, \mathcal{H}, Att), \vec{grt}, ik = (\gamma, \{s_j\}_{\text{att}_j \in Att}), gpk = (g_1, h, g_2, \hat{h}, w, \{\hat{h}_t\}_{t=1}^{\mathbb{T}}, \{\hat{h}_{\text{att}_j}\}_{\text{att}_j \in Att})$. $params, gpk$ and $\vec{grt}$ are given to $\mathcal{A}$.

Queries: In the Join queries, $\mathcal{B}$ can get a secret value y_i of a corrupted user by extracting the commitment value. $\mathcal{B}$ computes a group membership certificate as follows:
In the case of $i = i^\star : A_{i^\star} = h^\nu = (h^{\nu\xi})^{\frac{1}{\xi}} = (g_1 h^{y_{i^\star}})^{\frac{1}{\gamma + x_{i^\star}}}$.
In the case of $i \neq i^\star$:Compute A_i as follows:

$$\begin{aligned} A_i &= \left(g^{\prod_{j=1, j\neq i, i^\star}^{q-1} (\xi - x_{i^\star} + x_j)}\right)^{y_i - y_{i^\star}} g^{\nu f_i(\gamma)} \\ &= g^{\frac{y_i}{\xi + x_i - x_{i^\star}} \prod_{j=1, j\neq i^\star}^{q-1} (\xi - x_{i^\star} + x_j)} \times \\ &\quad \left\{ g^{\nu f(\gamma)} / g^{y_{i^\star} \prod_{j=1, j\neq i^\star}^{q-1} (\xi - x_{i^\star} + x_j)} \right\}^{\frac{1}{\xi + x_i - x_{i^\star}}} \\ &= (g_1 h^{y_i})^{\frac{1}{\gamma + x_i}} \end{aligned}$$

$\mathcal{B}$ computes $\{T_{i,j}\}_{\mathrm{att}_j \in \mathcal{A}_i} = \{A_i^{s_j}\}_{\mathrm{att}_j \in \mathcal{A}_i}$. Thus $\mathcal{B}$ can answer all the queries made by an adversary according to `traceability` game.
Output: Finally, $\mathcal{A}$ outputs a forged signature $\sigma^* = (\{C_i^*\}_{i=1}^4, c^*, s_\alpha^*, s_\beta^*, s_\tau^*, s_x^*, r^*)$ with ϵ advantage.
By using the forking lemma, $\mathcal{B}$ can get the two valid signatures $(\{C_i^*\}_{i=1}^4, c^*, s_\alpha^*, s_\beta^*, s_\tau^*, s_x^*, r^*)$ and $(\{C_i^*\}_{i=1}^4, c', s'_\alpha, s'_\beta, s'_\tau, s'_x, r^*)$ with probability $\epsilon' \geq \frac{1}{5} - \frac{8q_h}{\eta 2^k}, \eta > \frac{240 q_H}{2^k}$ [48]. Let $c'' = c^* - c', s''_\alpha = s^*_\alpha - s'_\alpha, s''_\beta = s^*_\beta - s'_\beta, s''_x = s^*_x - s'_x$ and $s''_\tau = s^*_\tau - s'_\tau$. Let $\tilde{x} = s''_x / c'', \tilde{\alpha} = s''_\alpha / c'', \tilde{\tau} = s''_\tau / c'', \tilde{A} = C_1^* / h^{\tilde{\alpha}}$, and $\tilde{y} = \tilde{\tau} - \tilde{\alpha}\tilde{x}$.
Now $(\tilde{A}, \tilde{x}, \tilde{y})$ is a valid member certificate because $\frac{e(C_1^*, w)}{e(g_1, g_2)} = \frac{e(h, g_2)^{\tilde{\tau}} e(h, w)^{\tilde{\alpha}}}{e(C_1^*, g_2)^{\tilde{x}}}$ holds. From the success of the adversary in the attack game, we know that $\tilde{A}$ does not belong to $\{A_i\}_{i=1}^{q-1}$. We assume that $\tilde{x} \neq x_{i^\star}$.
Consider,

$$\begin{aligned} \tilde{A} &= (g_1 h^{\tilde{y}})^{\frac{1}{\tilde{x} + \gamma}} \\ &= (h^{\nu\xi} h^{\tilde{y} - y_{i^\star}})^{\frac{1}{\tilde{x} + \gamma}} \\ &= h^{\frac{\nu\xi + (\tilde{y} - y_{i^\star})}{\tilde{x} + \gamma}} \\ &= \left(g^{(\nu\xi + (\tilde{y} - y_{i^\star})) \prod_{i=1, i\neq i^\star}^{q-1} (\xi + x_i - x_{i^\star})} \right)^{\frac{1}{\tilde{x} + \xi - x_{i^\star}}} \\ &= \left(g^{\sum_{i=0}^{q-1} z_i \xi^i} \right)^{\frac{1}{\tilde{x} + \xi - x_{i^\star}}} \text{ (can be written in this form)} \end{aligned}$$

$$= \quad g^{\frac{\tilde{z}_0}{\tilde{x}+\xi-x_{i^\star}}+\sum_{i=1}^{q-1}\tilde{z}_i\xi^i} \text{(can be written in this form)}$$

The polynomial coefficients $z_0, ..., z_{q-1}, \tilde{z}_0, \tilde{z}_1, ..., \tilde{z}_{q-1}$ are computable. Let $x = \tilde{x} - x_{i^\star}$, then $(\tilde{A}/g^{\sum_{i=1}^{q-1}\tilde{z}_i\xi^i})^{\frac{1}{\tilde{z}_0}} = g^{\frac{1}{x+\xi}}$ holds. Therefore $(x, g^{\frac{1}{x+\xi}})$ is the new SDH tuple. If $\tilde{x} = x_{i^\star}$ then $(0, g^{\frac{1}{\xi}})$ will be the new SDH tuple. The advantage of $\mathcal{B}$ is $(\frac{1}{5} - \frac{8q_H}{\eta 2^k})\epsilon \geq \frac{1}{6}\epsilon$, since $\eta > \frac{240q_H}{2^k}$. □

Theorem 4.4.10 *Suppose an adversary $\mathcal{A}$ breaks the non-frameability of the proposed scheme with the advantage ϵ. Then, an algorithm $\mathcal{B}$ can be constructed that breaks the DL assumption with the advantage $\frac{1}{12}(1 + \frac{1}{n})\epsilon$, where n is the number of honest members.*

Proof Since the membership certificate format is similar to the one proposed in [48], the proof is similar to the proof given in [48; 52]. The input of simulator $\mathcal{B}$ is $(g, g') \in \mathbb{G}_2 \times \mathbb{G}_2$, let $\xi = log_g g'$. We consider the two types of adversaries by the results of the `OpenUser` algorithm. We explain the details of classification of the adversary in the proof. Let q be the number of all members, n be the number of honest members, and $q_1 = q - n$ be the number of corrupt members. We assume that all initial members $\{U_1, ..., U_n\}$ are honest.

Setup: $\mathcal{B}$ simulates KeyGen as follows:

(i) $\mathcal{B}$ selects $d \in_R \{0, 1\}$. If $d = 1$, then $\mathcal{B}$ selects a target user $U_{i^\star} \in \{U_1, ..., U_n\}$. Note that $d = 0$ means $\mathcal{B}$ guesses that $\mathcal{A}$ is Type 1 adversary, and $d = 1$ means $\mathcal{B}$ guesses that $\mathcal{A}$ is Type 2 adversary.

(ii) $\mathcal{B}$ computes the group public key and member certificates as follows:

  (a) $\mathcal{B}$ selects $\gamma, \{s_j\}_{(\forall \text{att}_j \in Att)}, \{x_i, y_i\}_{i=1}^{q} \in_R \mathbb{Z}_p^*$. If $d = 1$, then set $y_{i^\star} = \xi$.

  (b) If $d = 0$, then $\mathcal{B}$ sets $g_1 = \psi(g), g_2 = g, \hat{h} = g'$ and $h = \psi(g')$.

  (c) If $d = 1$, then $\mathcal{B}$ selects $g_2 \in_R \mathbb{G}_2$ and sets $g_1 = \psi(g_2), \hat{h} = g, h = \psi(g)$ and $y_{i^\star} = \xi$.

  (d) $\mathcal{B}$ computes $w = g_2^{\gamma}$.

(e) $\mathcal{B}$ computes member certificates $\{(A_i, x_i, y_i)\}_{i=1}^{q}$ by using γ. If $d = 1$, then $A_{i^\star} = (g_1\psi(g'))^{\frac{1}{x_{i^\star}+\gamma}} = (g_1 h^{y_{i^\star}})^{\frac{1}{x_{i^\star}+\gamma}}$.

(f) $\mathcal{B}$ computes other values as the real settings, and gets $params = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \psi, \mathcal{H}_0, \mathcal{H}, Att), \vec{grt}, ik = (\gamma, S)$, and $gpk = (g_1, h, g_2, \hat{h}, w, \{\hat{h}_t\}_{t=1}^{\mathbb{T}}, \{g_{\mathrm{att}_j}\}_{\mathrm{att}_j \in Att})$.

(iii) $\mathcal{B}$ gives $params, gpk, ik$ and $\vec{grt}$ to $\mathcal{A}$.

Queries: In Join queries, $\mathcal{A}$ knows (A_i, x_i), for $i \in [1, q]$, because $\mathcal{A}$ plays the role of corrupted GM. However, $\mathcal{A}$ cannot know secret key of a target user $y_i^\star$. For Signing queries, $\mathcal{B}$ makes a group signature by using (A_i, x_i, y_i), and return its signature, if $d = 1$ and $i = i^\star$, then $\mathcal{B}$ aborts. For Corruption queries, $\mathcal{B}$ answers y_i, if $d = 1$ and $i = i^\star$, then $\mathcal{B}$ aborts.
Output: Finally, $\mathcal{A}$ outputs the valid group signature for honest user, say U_k. We can get the member certificate $(\tilde{A}, \tilde{x}, \tilde{y})$ by using the same technique as for traceability. We define a Type 1 adversary $\mathcal{A}$, which is the case of $\tilde{A} = A_k \in \{A_i\}_{i=1}^{n}$ and $\tilde{x} \neq x_k$. We define a Type 2 adversary $\mathcal{A}$, which is the case of $(\tilde{A}, \tilde{x}) = (A_k, x_k)$.

- In the case of Type 1 : If $d \neq 0$, then $\mathcal{B}$ aborts. Otherwise $\tilde{A} = (g_1 h^{\tilde{y}})^{\frac{1}{\tilde{x}+\gamma}} = (g_1^{1+\xi\tilde{y}})^{\frac{1}{\tilde{x}+\gamma}}$ holds. As $\tilde{A} = A_k = (g_1 h^{y_k})^{\frac{1}{x_k+\gamma}} = (g_1^{1+\xi y_k})^{\frac{1}{x_k+\gamma}}$ holds. Therefore, $\mathcal{B}$ can compute $\xi = \frac{\tilde{x}-x_k}{\{\tilde{y}(x_k+\gamma)-y_k(\tilde{x}+\gamma))\}}$.
- In the case of Type 2 : If $d \neq 1$, then $\mathcal{B}$ aborts. If $k \neq i^\star$, then $\mathcal{B}$ aborts. Otherwise, $\tilde{A} = (g_1 h^{\tilde{y}})^{\frac{1}{\tilde{x}+\gamma}} = (g_1\psi(g)^{\tilde{y}})^{\frac{1}{\tilde{x}+\gamma}}$ holds. Moreover, $\tilde{A} = A_{i^\star} = (g_1 h^{y_{i^\star}})^{\frac{1}{x_{i^\star}+\gamma}} = (g_1\psi(g)^{y_{i^\star}})^{\frac{1}{x_{i^\star}+\gamma}}$ holds. Therefore $\mathcal{B}$ can get $\xi = \tilde{y}$.

The advantage of $\mathcal{B}$ is $(\frac{1}{2}(\frac{1}{5} - \frac{8q_H}{\eta 2^k})\epsilon + \frac{1}{2}\frac{1}{n}(\frac{1}{5} - \frac{8q_H}{\eta 2^k})\epsilon) > \frac{1}{12}(1 + \frac{1}{n})\epsilon$, since $\eta > \frac{240q_H}{2^k}$. □

4.5 Comparison

In the proposed scheme, a group signature contains 4 elements from $\mathbb{G}_1$ and 6 elements from $\mathbb{Z}_p^*$. The use of MNT family of curves [89] can make the representation of

elements in $\mathbb{G}_1$ short, one can take p to be a 170-bit prime, and the represent of $\mathbb{G}_1$ and $\mathbb{G}_T$ can be expressed in 171 and 1020 bits, respectively [26; 28]. Thus, the total group signature length is 1704 bits(= 213 bytes).

Let $\Phi = |\zeta|$, where ζ be the set of attributes which is associated with a signature and $m = |Att|$. Let $\hat{m}$ be the average number of attributes assigned to any user. RO means Random oracle model, e represents the paring operation and r represents the number of revoked members. In Table 4.1, we compare the efficiency of our scheme with other schemes. Note that the parings values $e(h, g_2), e(h, w)$ and $e(g_1, g_2)$ can be precomputed and use in the algorithms to reduce the number of paring operations in each invocation of algorithm. Also note that the verification cost of the proposed scheme is independent of the number of attributes, where as in other schemes the verification cost is linear in terms of the number of attributes. From the table it can be noticed that the combined scheme of Herranz et al. [70] and Libert et al. [82] in the standard model neither achieves non-frameability nor its verification cost is independent of the number of attributes and also all the key lengths are large.

Table 4.1: Comparison of VLR-ABGS scheme with other schemes

	Khader [72]	Emura et al. [52]	$\binom{\text{Herranz et al. [70]}}{\text{+Libert et al. [82]}}$	Our Scheme
Attribute anonymity	no	no	yes	yes
VLR	yes	no	yes	yes
Backward unlinkability	no	no	yes	yes
Non-frameability	no	yes	no	yes
Attribute revocation	yes	no	no	no
Predicate	monotone	monotone	threshold	monotone
Signature length	$\mathrm{O}(\Phi)$	$\mathrm{O}(\Phi)$	$\binom{(15\lvert\mathbb{G}_p\rvert)+}{(46\lvert\mathbb{G}_p\rvert+1\lvert\mathbb{G}_T\rvert))}=\mathrm{O}(1)$	$4\lvert\mathbb{G}_p\rvert + 6\lvert\mathbb{Z}_p^*\rvert=\mathrm{O}(1)$
User's secret key length	$(\hat{m}+1)\lvert\mathbb{G}_1\rvert + \lvert\mathbb{Z}_p^*\rvert$	$(\hat{m}+1)\lvert\mathbb{G}_1\rvert + 2\lvert\mathbb{Z}_p^*\rvert$	$((\hat{m}+m-1)\lvert\mathbb{G}_p\rvert) + (3\lvert\mathbb{G}_p\rvert)$	$\lvert\mathbb{G}_1\rvert + 2\lvert\mathbb{Z}_p^*\rvert$
Assumption	DLIN, q-SDH	DL,DDH, q-SDH	$\binom{\text{DLin},(\ell,m,t)-\text{aMSE}-\text{CDH},}{\ell-\text{HSDH,DTDH}}$	$\binom{\text{DL},q-\text{SDH, DLIN,}}{\text{KEA1}}$
Model	RO	RO	standard	RO
Signing cost	$\binom{(7+2\Phi)\mathbb{G}_1+(5+\Phi)\mathbb{G}_T}{+(\Phi+1)e}$	$\binom{(9+3\Phi)\mathbb{G}_1+(1+\Phi)\mathbb{G}_2}{+8\mathbb{G}_T+3e}$	$\binom{((6m+6m'+\Phi(\Phi-1)+14)\mathbb{G}_p)}{+((m'+440)\mathbb{G}_p)}$	$\binom{(2\Phi+10)\mathbb{G}_1+5\mathbb{G}_T}{+1e}$
Verification cost	$\binom{(6+2r)\mathbb{G}_1+(8+2\Phi)\mathbb{G}_T}{+(\Phi+2r+1)e}$	$\binom{(11+2\Phi)\mathbb{G}_1+(\Phi+1)\mathbb{G}_2}{+14\mathbb{G}_T+6e}$	$\binom{((4m+6m')\mathbb{G}_p+21\mathbb{G}_T+33e)}{+((m'+3)\mathbb{G}_p+48\mathbb{G}_T+(75+r)e)}$	$\binom{(r+8)\mathbb{G}_1}{+8\mathbb{G}_T+(5+r)e}$

4.6 Summary

We have proposed a VLR-ABGS scheme which achieves attribute anonymity and has backward unlinkability feature with constant signature length, and proven that it is secure under random oracle model with well known assumptions. We note that our scheme is efficient than the other ABGS schemes in terms of signing cost, verification cost and signature length. Furthermore, suspension of group member is possible for a prescribed time interval.

Chapter 5

ABGS Schemes with Attribute Anonymity without Non-frameability in the Standard Model

In the previous chapters, we proposed ABGS schemes with attribute anonymity secure in the random oracle model. In this chapter, we propose an ABGS scheme with attribute anonymity and constant length signature which is secure in the standard model. Further, we propose another scheme having same features but with shorter signature length.

5.1 Introduction

Khader's ABGS schemes are secure in the random oracle model [72; 73]. Emura et al. proposed an ABGS scheme in the random oracle model [51]. All these scheme's signature length is variable with the number of attributes and does not have attribute anonymity. We propose an ABGS scheme with attribute anonymity in the standard

model [4]. Our construction achieves constant length signature which is independent of the number of attributes. We also device another construction for short ABGS scheme with attribute anonymity with the similar features as above but with the shorter signature length [3]. We prove that our schemes, in the standard model, preserve attribute anonymity unconditionally, user anonymity in CPA attacks under Subgroup Decision assumption, preserves traceability and attribute unforgeability under DL and KEA1 assumptions. Our schemes do not preserve non-frameability.

In Section 5.2, we present the proposed ABGS scheme with attribute anonymity and the related security definitions. The construction of the proposed ABGS scheme is described in Section 5.3 along with the security analysis and comparison with previous schemes. In Section 5.4 we give another construction of the proposed ABGS scheme with shorter signature length along with the security analysis and comparison with previous schemes. Finally we summarize in Section 5.5.

5.2 Proposed Scheme

In this section, we give some definitions which are similar to the one given in papers [33; 52; 73] but altered to add attribute anonymity.

Let k be the security parameter, Att be the universal set of attributes, Υ used to denote the predicate, T_Υ be an access tree representing the predicate Υ, $\mathcal{T}_\Upsilon$ the public values associated with T_Υ, gpk the group public key used to verify the validity of the group signature, ik the issuing key used for issuing private keys to users, ok_{User} the opening key used for opening the signer's identity from the given group signature, $\mathrm{id} \in \{0,1\}^k$ represents the user identity, k_{id} be the user's private key, $\mathcal{A}_{\mathrm{id}} \subseteq Att$ the attributes of the user with identity id and $\Upsilon(\zeta) = 1$ denotes that the attribute set ζ satisfies the predicate Υ.

Definition 5.2.1 (ABGS) *An ABGS scheme consists of the following five algorithms. Unless specified all the algorithms are probabilistic.*

- $(params, gpk, ik, ok_{user}) \leftarrow \texttt{Setup}(1^k)$*: It takes the security parameter k as an input and generates system parameters params, a group public key gpk, an issuing key ik for enrolling group members and an opening key ok_{user} for identifying the signers.*

- $k_{\mathrm{id}} \leftarrow \texttt{Join}(gpk, ik, \mathrm{id}, \mathcal{A}_{\mathrm{id}})$*: This algorithm generates the private key for the user with identity* id*. It takes ik, user identity* id *and subset of attributes $\mathcal{A}_{\mathrm{id}} \subseteq Att$, and outputs a user private key k_{id} which is to be given to the user.*

- $\sigma \leftarrow \texttt{Sign}(gpk, k_{\mathrm{id}}, \zeta, M, \Upsilon)$*: It takes gpk, k_{id}, an attribute set $\zeta \subseteq \mathcal{A}_{\mathrm{id}}$, a message M and a predicate Υ, and outputs a group signature σ.*

- $0/1 \leftarrow \texttt{Verify}(gpk, M, \Upsilon, \sigma)$*: This is a deterministic algorithm which outputs a boolean value. If it is 1 it claims that σ is a valid group signature on M with respect to Υ, otherwise invalid.*

- $\mathrm{id}/\perp \leftarrow \texttt{Open}(gpk, ok_{user}, \sigma)$: *This is a deterministic algorithm which takes an opening key ok_{user} and a group signature σ, and outputs either* id *or $\perp$. If* id *then the algorithm claims that the group member with an identity* id *has produced σ, and if $\perp$, it claims that no group member produced σ.*

Following are the entities in ABGS scheme:

- Group manager `GM` has issuing key ik and opening key ok_{user}. Using issuing key `GM` enrolls a user into the group by allotting some privilege (in terms of attributes) say $\mathcal{A}_{\mathrm{id}} \subseteq Att$ and issuing a user's private key k_{id}, by running the `Join` algorithm. `GM` runs `Open` algorithm to reveal the signer's identity from the group signature.

- Group members or signers who are having their private keys k_{id}'s. They run `Sign` algorithm to produce a group signature on a document M with predicate Υ if they possess valid attribute set which satisfies the predicate.

- Outsider or verifier who can seek a group signature for a document M with predicate Υ from group manager GM. He can also verify the group signature using the group public key, gpk.

ABGS scheme is correct if the group signatures produced by an honest group member are verified and reveals the identity of the signer.

Definition 5.2.2 (Correctness) *We call an ABGS scheme is* correct *if for all honestly generated* $(params, gpk, ik, ok_{user}) \leftarrow \mathtt{Setup}(1^k)$*, for all* $k_{\mathrm{id}} \leftarrow \mathtt{Join}(gpk, ik, \mathrm{id}, \mathcal{A}_{\mathrm{id}})$ *the following equations hold.*

$$1 \leftarrow \mathtt{Verify}(gpk, M, \Upsilon, \mathtt{Sign}(gpk, k_{\mathrm{id}}, \zeta, M, \Upsilon))$$

$$: \zeta \subseteq \mathcal{A}_{\mathrm{id}} \text{ and } \Upsilon(\zeta) = 1$$

$$\mathrm{id} \leftarrow \mathtt{Open}(gpk, ok_{user}, \mathtt{Sign}(gpk, k_{\mathrm{id}}, \zeta, M, \Upsilon))$$

We write $\mathsf{Sign}(gpk, ik, id, \zeta, M, \Upsilon)$ (i.e. in place of user private key k_{id}, we use issuing key ik and user identity id) to denote the following task: pick the corresponding k_{id} from the list (we assume that a list $\{(\mathrm{id}, k_{\mathrm{id}})\}$ is maintained) and returns the group signature $\sigma \leftarrow \mathtt{Sign}(gpk, k_{\mathrm{id}}, \zeta, M, \Upsilon)$, and if the related k_{id} is not present in the list (i.e. k_{id} is not generated yet) then choose $\mathcal{A}_{\mathrm{id}} \subseteq Att$ randomly such that $\exists \zeta \in \mathcal{A}_{\mathrm{id}}, \Upsilon(\zeta) = 1$ and get $k_{\mathrm{id}} \leftarrow \mathtt{Join}(gpk, ik, \mathrm{id}, \mathcal{A}_{\mathrm{id}})$, store it in a list and finally return the intended group signature $\sigma \leftarrow \mathtt{Sign}(gpk, k_{\mathrm{id}}, \zeta, M, \Upsilon)$.
For convenience, in the definitions below we denote sign oracle as $\mathsf{Sign}(gpk, ik, ., ., ., .)$ to generate the group signature requested by an adversary with the query that includes user identity id, a message M and a predicate Υ. And we denote join oracle as $\mathsf{Join}(gpk, ik, ., .)$ to generate a user private key $k_{\mathrm{id}} \leftarrow \mathtt{Join}(gpk, ik, \mathrm{id}, \mathcal{A}_{\mathrm{id}})$ upon input id and an attribute set $\mathcal{A}_{\mathrm{id}} \subseteq Att$ queried by the adversary.

In ABGS scheme a group member may have multiple attribute sets to satisfy the predicate and he can produce a group signature using one of them. An ABGS scheme preserves attribute anonymity if it is computationally difficult to identify with what attribute set he produces the signature.

Definition 5.2.3 (`Attribute anonymity`) *We say that the ABGS scheme preserves* attribute anonymity *if, for all honestly generated* $(params, gpk, ik, ok_{user}) \leftarrow \texttt{Setup}(1^k)$*, for all predicates* Υ*, for all attribute sets* $\mathcal{A}_{\text{id}} \subseteq Att$ *such that there exist* $\zeta_1, \zeta_2 \subseteq \mathcal{A}_{\text{id}}$ *and* $\Upsilon(\zeta_1) = \Upsilon(\zeta_2) = 1$*, for all* $k_{\text{id}} \leftarrow \texttt{Join}(gpk, ik, \text{id}, \mathcal{A}_{\text{id}})$ *and all messages* M*, the distributions* $\texttt{Sign}(gpk, k_{\text{id}}, \zeta_1, M, \Upsilon)$ *and* $\texttt{Sign}(gpk, k_{\text{id}}, \zeta_2, M, \Upsilon)$ *are identical.*

In other words, even the computationally unbounded adversary cannot link a signature to a set of attributes used to generate it.

ABGS scheme preserves user anonymity if there are at least two group members possessing valid attribute sets and one of them produces the group signature then it should be computationally hard to identify who produced the group signature among them.

Definition 5.2.4 (`User anonymity (CPA)`) *We say that the ABGS scheme preserves* user anonymity *under* CPA[1] *if for all PPT* $\mathcal{A}$*, the probability that* $\mathcal{A}$ *wins the following game is negligible.*

- `Setup` : *The simulator* $\mathcal{B}$ *generates* $(params, gpk, ik, ok_{user}) \leftarrow \texttt{Setup}(1^k)$*.* $\mathcal{B}$ *gives* gpk *to* $\mathcal{A}$*.*
- `Phase1` : $\mathcal{A}$ *is given access to the oracles* $\mathit{Join}(gpk, ik, ., .)$ *and* $\mathit{Sign}(gpk, ik, ., ., .)$*.*
- `Challenge` : $\mathcal{A}$ *outputs* M^*, Υ^* *and two identities* $ID_1, ID_2 : \exists \zeta_1 \subseteq \mathcal{A}_{ID_1}, \zeta_2$[2]$\subseteq \mathcal{A}_{ID_2}$ *and* $\Upsilon^*(\zeta_1) = \Upsilon^*(\zeta_2) = 1$ *to be challenged. The simulator* $\mathcal{B}$ *randomly selects* $x \in_R \{1, 2\}$ *and responds with a group signature* $\sigma^* \leftarrow \texttt{Sign}(gpk, k_{ID_x}, \zeta_x, M^*, \Upsilon^*)$*. The constraints are the private keys of* ID_1 *and* ID_2 *to the join oracle, and group signatures on* (M^*, Υ^*, ID_1) *and* (M^*, Υ^*, ID_2) *to the sign oracle should not be queried before.*

[1]Note: If we provide Open oracle to the adversary then the user anonymity will be enhanced to CCA - security notion

[2]ζ_2 can be equal to ζ_1. Since we are concerned only about the user anonymity the attribute anonymity is separately considered in attribute anonymity definition.

- `Phase2` : $\mathcal{A}$ *can make all queries similar to* `Phase1` *under the constraints mentioned above.*
- `Output` : $\mathcal{A}$ *outputs a bit* x'*, and wins if* $x = x'$.

The advantage of $\mathcal{A}$ *is defined as* $Adv_{\mathcal{A}} = |Pr(x = x') - \frac{1}{2}|$.

Thus there should not exist any PPT adversary to link a group signature to a user with non-negligible probability.

ABGS scheme preserves traceability if it is possible to trace the valid group signature to its signer with the help of group opening key.

Definition 5.2.5 (`Traceability`) *We say that the ABGS scheme is* traceable *if for all PPT* $\mathcal{A}$*, the probability that* $\mathcal{A}$ *wins the following game is negligible.*

- `Setup` : *The simulator* $\mathcal{B}$ *generates* $(params, gpk, ik, ok_{user}) \leftarrow \texttt{Setup}(1^k)$. $\mathcal{B}$ *gives* (gpk, ok_{user}) *to* $\mathcal{A}$.
- `Queries` : $\mathcal{A}$ *is given access to the oracles* $\mathsf{Join}(gpk, ik, ., .)$ *and* $\mathsf{Sign}(gpk, ik, ., ., ., .)$.
- `Output` : $\mathcal{A}$ *outputs a message* M^**, a predicate* Υ^* *and a group signature* σ^*.

$\mathcal{A}$ *wins if*
(1) $\texttt{Verify}(gpk, M^*, \Upsilon^*, \sigma^*) = 1$ *and (2)* $\texttt{Open}(gpk, ok_{user}, \sigma^*) = \perp$.

Thus it should be impossible to produce an untraceable valid group signature by any PPT adversary.

ABGS scheme preserves attribute unforgeability if it is hard for a group member to forge an attribute certificate in order to produce a valid group signature.

Definition 5.2.6 (`Attribute unforgeability`) *We say that the ABGS scheme preserves* attribute unforgeability *if for all PPT $\mathcal{A}$, the probability that $\mathcal{A}$ wins the following game is negligible.*

- `Setup` : *The simulator $\mathcal{B}$ generates* $(params, gpk, ik, ok_{user}) \leftarrow \mathtt{Setup}(1^k)$. *$\mathcal{B}$ gives gpk to $\mathcal{A}$.*
- `Queries` : *$\mathcal{A}$ is given access to the oracles* $\mathsf{Join}(gpk, ik, ., .)$ *and* $\mathsf{Sign}(gpk, ik, ., ., .)$.
- `Output` : *$\mathcal{A}$ outputs a message M^*, a predicate Υ^* and a group signature σ^*.*

$\mathcal{A}$ wins if
(1) $\mathtt{Verify}(gpk, M^*, \Upsilon^*, \sigma^*) = 1$,
(2) $\mathtt{Open}(gpk, ok_{user}, \sigma^*) = \mathrm{id}$ *and*
(3) $\nexists \zeta \in \mathcal{A}_{\mathrm{id}} : \Upsilon(\zeta) = 1$.

Thus it should be impossible for any PPT adversary to satisfy the predicate with invalid set of attributes.

ABGS scheme preserves collusion resistance of attribute certificates if it is computationally hard for group members to collude by pooling their attribute certificates to satisfy the predicate and to produce a valid group signature.

Definition 5.2.7 (`Collusion resistance of attributes`) *We say that the ABGS scheme preserves* collusion resistance of attributes *if for all PPT $\mathcal{A}$, the probability that $\mathcal{A}$ wins the following game is negligible.*

- `Setup` : *The simulator $\mathcal{B}$ generates* $(params, gpk, ik, ok_{user}) \leftarrow \mathtt{Setup}(1^k)$. *$\mathcal{B}$ gives gpk to $\mathcal{A}$.*
- `Queries` : *$\mathcal{A}$ is given access to the oracles* $\mathsf{Join}(gpk, ik, ., .)$ *and* $\mathsf{Sign}(gpk, ik, ., ., .)$.
- `Output` : *$\mathcal{A}$ outputs a message M^*, a predicate Υ^* and a group signature σ^*.*

$\mathcal{A}$ wins if (1) $\texttt{Verify}(gpk, M^*, \Upsilon^*, \sigma^*) = 1$*, and (2) $\mathcal{A}$ has obtained* $k_{\mathrm{id}_1}, ..., k_{\mathrm{id}_k}$: $\Upsilon^*(\cup_{j=1}^{k} \mathcal{A}_{\mathrm{id}_j}) = 1$ *and* $\Upsilon^*(\mathcal{A}_{\mathrm{id}_j}) \neq 1$ *for* $j = 1, ..., k$.

Thus the users with invalid set of attributes each, cannot collude with each other to pool a valid attribute set for producing a valid group signature.

Definition 5.2.8 (Full anonymity) *We say that the ABGS scheme preserves* full anonymity *if it preserves both attribute anonymity and user anonymity. That is, signature should not reveal the signer's identity and also the attributes he holds.*

Definition 5.2.9 (Full traceability) *We say that the ABGS scheme preserves* full traceability *if it preserves traceability, attribute unforgeability and collusion resistance of attributes. That is, even if there exists a coalition of users, it is impossible to forge signatures and attributes.*

5.3 Construction - 1

In this section, we describe our first construction of ABGS scheme with attribute anonymity [4]. Our construction is based on the Boyen et al.'s two-level signature scheme from [34] and the technique to build the access trees is from [51]. We use non-interactive proof system technique of Groth and Sahai under subgroup decision assumption (see Sec. 2.4.7) to generate the NIWI proofs for the relations in the group signature, which helps to preserve the user anonymity of the proposed scheme. We prove that our scheme, in the standard model, preserves attribute anonymity unconditionally, user anonymity in CPA attacks under Subgroup Decision assumption, traceability under ℓ-HSDH assumption and attribute unforgeability under DL and KEA1 assumptions. In contrast to other existing ABGS schemes [51; 72; 73] our scheme is built in standard model with attribute anonymity and achieves a constant size signature independent of the number of attributes.

- `Setup`(1^k): It takes the security parameter k as an input and outputs the system parameters $params$, group public key gpk, issuing key ik and the opening key ok_{user}.

  (i) Select the primes p and q of size k. Let the groups $\mathbb{G}$ and $\mathbb{G}_T$ be of order $n = pq$ for which there exists a bilinear map e from $\mathbb{G} \times \mathbb{G}$ to $\mathbb{G}_T$. Let $\mathbb{G}_p$ and $\mathbb{G}_q$ be the subgroups of $\mathbb{G}$ of order p and q, respectively.

  (ii) Define the universal set of attributes, $Att = \{\text{att}_1, ..., \text{att}_m\}, m = O(k)$.

  (iii) Select the generators $g, u \in \mathbb{G}$ and $h \in \mathbb{G}_q$.

  (iv) For each attribute $att_i \in Att$ select the attribute secrets $s_i \in \mathbb{Z}_q^*$. Let $S = \{s_i\}_{att_i \in Att}$.

  (v) Compute the public values of the attributes $\{h_{att_i} = h^{s_i}\}_{att_i \in Att}$.

  (vi) Select the exponents $\alpha, z \in_R \mathbb{Z}_n^*$ and compute $Z = g^z$.

  (vii) Select the generators $v', v_1, ..., v_{m'} \in \mathbb{G}$ and define the Waters function, $\mathcal{F} : \{0,1\}^{m'} \rightarrow \mathbb{G}$, for $M = (\mu_1, ..., \mu_{m'}) \in \{0,1\}^{m'}$ such that $\mathcal{F}(M) = v' \Pi_{j=1}^{m'} v_j^{\mu_j}$, where $m' = O(k)$.

  (viii) Output the system parameters,

  $$params = (n, \mathbb{G}, \mathbb{G}_T, e, Att)$$

  The group public key,

  $$gpk = (g, u, Z, h, \mathcal{F}, \{h_{att_i}\}_{att_i \in Att}) \in \mathbb{G}^3 \times \mathbb{G}_q \times \mathbb{G}^{m'+1} \times \mathbb{G}_q^m$$

  The issuing key ik and the opening key ok_{user},

  $$ik = (g^\alpha, z, S) \in \mathbb{G} \times \mathbb{Z}_n^* \times \mathbb{Z}_q^{*m}, ok_{user} = q \in \mathbb{Z}$$

  The description of $\mathcal{F}$ includes the generators $v', v_1, ..., v_{m'}$.

- `Join`$(gpk, ik, \text{id}, \mathcal{A}_{\text{id}})$: It takes group public key, issuing key, user identity id and subset of attributes $\mathcal{A}_{\text{id}} \subseteq Att$, and outputs user private key k_{id}.

(i) Select the unique identifier $s_{\text{id}} \in \mathbb{Z}_n^*$.

(ii) Compute the membership certificate $g_{\text{id}} = (g^{\alpha})^{\frac{1}{z+s_{\text{id}}}}$.

(iii) Compute the attribute certificates $\{g_{\text{id},i} = g_{\text{id}}^{s_i}\}_{\text{att}_i \in \mathcal{A}_{\text{id}}}$.

(iv) Output the user private key,

$$k_{\text{id}} = (k_{\text{id},1}, k_{\text{id},2}, k_{\text{id},3}, k_{\text{id},4}) = (g_{\text{id}}, g^{s_{\text{id}}}, u^{s_{\text{id}}}, \{g_{\text{id},i}\}_{\text{att}_i \in \mathcal{A}_{\text{id}}}) \in \mathbb{G}^{3+|\mathcal{A}_{\text{id}}|}$$

- `BuildTree`(gpk, ik, Υ): It generates a public values for the predicate Υ.

  (i) Let T_{Υ} be the tree that represents the predicate Υ.

  (ii) Get extension tree $T^{\text{ext}} \leftarrow$ `AddDummyNode`(T_{Υ}).

  (iii) Get secret values for each dummy node and the secret value of root of T^{ext} using $(\{s_{d_j}\}_{d_j \in D_T}, s_T) \leftarrow$ `AssignedVaule`(S, T^{ext}).

  (iv) Output the public values of tree T^{ext},

$$\mathcal{T}_{\Upsilon} = (\{s_{d_j}\}_{d_j \in D_T}, A_T = e(g^{\alpha}, g^{s_T}), T^{\text{ext}})$$

- `Sign`$(gpk, k_{\text{id}}, \zeta, M, \Upsilon)$: It generates a group signature σ on message $M \in \{0,1\}^{m'}$ with user private key k_{id} who satisfy the predicate Υ with his subset of attributes $\zeta \subseteq \mathcal{A}_{\text{id}} : \Upsilon(\zeta) = 1$.

  (i) Get the public values of Υ from the public repository[1].

  (ii) Select a random $s \in \mathbb{Z}_n^*$. Compute $\rho = (\rho_1, \rho_2, \rho_3, \rho_4) = (k_{\text{id},1}^{s_T}, k_{\text{id},2}, k_{\text{id},3}.\mathcal{F}(M)^s, g^{-s})$ where $\rho_1 = k_{\text{id},1}^{s_T}$ is computed as follows:

  - Select $\zeta \subseteq \mathcal{A}_{\text{id}} : \Upsilon(\zeta) = 1$.
  - Get $(\{\Delta_{\text{att}_j}\}_{(\forall \text{att}_j \in \zeta)}, \{\Delta_{d_j}\}_{(\forall d_j \in D_T^{\zeta})}) \leftarrow$ `MakeSimplifiedTree`(ζ, T^{ext}).
  - Compute $\rho_1 = \Pi_{\text{att}_i \in \zeta} g_{\text{id},i}^{\Delta_{\text{att}_i}} g_{\text{id}}^{(\Sigma_{d_j \in D_T^{\zeta}} \Delta_{d_j} s_{d_j})} = g_{\text{id}}^{\Sigma_{\text{att}_i \in \zeta} \Delta_{\text{att}_i} s_j} g_{\text{id}}^{\Sigma_{d_j \in D_T^{\zeta}} \Delta_{d_j} s_{d_j}} = g_{\text{id}}^{s_T}$.

[1] `GM` runs `BuildTree` algorithm to generate the public values of the predicate Υ and stores it in a public repository. Note that if the public values of the required predicate is present in the public repository then the user will not approach `GM`.

(iii) Commit the group elements ρ_i, for $i \in \{1, ..., 4\}$. Choose $t_1, t_2, t_3, t_4 \in \mathbb{Z}_n$, and compute $\sigma_1 = \rho_1 h^{t_1}, \sigma_2 = \rho_2 h^{t_2}, \sigma_3 = \rho_3 h^{t_3}$ and $\sigma_4 = \rho_4 h^{t_4}$.

(iv) Compute the NIWI proofs for the committed variables $\rho_1, ..., \rho_4$ satisfying the following pairing-product equations

$$e(Z, \rho_1)e(\rho_2, \rho_1) = A_T \tag{5.1}$$

$$e(g, \rho_3)e(\mathcal{F}(M), \rho_4)e(u^{-1}, \rho_2) = 1 \tag{5.2}$$

From (2.1) for the relation (5.1), according to Groth-Sahai NIWI proof system under subgroup decision assumption we have $\mathcal{A}_1 = Z, \mathcal{A}_2 = 1, \mathcal{X}_1 = \rho_1, \mathcal{X}_2 = \rho_2, t_T = A_T, \gamma_{11} = 0, \gamma_{12} = 0, \gamma_{21} = 1$ and $\gamma_{22} = 0$. Then the proof

$$\begin{aligned} \pi_1 &= \Pi_{i=1}^2 \mathcal{A}_i^{t_i} . \Pi_{i=1}^2 (\Pi_{j=1}^2 (\mathcal{X}_j^{t_i(\gamma_{ij}+\gamma_{ji})})) . \Pi_{i=1}^2 (\Pi_{j=1}^2 (h^{t_i t_j \gamma_{ij}})) \\ &= Z^{t_1} \rho_2^{t_1} \rho_1^{t_2} h^{t_2 t_1}. \end{aligned}$$

And for the relation (5.2), we have $\mathcal{A}_1 = g, \mathcal{A}_2 = \mathcal{F}(M), \mathcal{A}_3 = u^{-1}\mathcal{X}_1 = \rho_3, \mathcal{X}_2 = \rho_4, \mathcal{X}_3 = \rho_2, t_T = 1$ and $\gamma_{ij} = 0$ for $i, j \in \{1, 2, 3\}$. Then the proof

$$\begin{aligned} \pi_2 &= \Pi_{i=1}^3 \mathcal{A}_i^{t_i} . \Pi_{i=1}^3 (\Pi_{j=1}^3 (\mathcal{X}_j^{t_i(\gamma_{ij}+\gamma_{ji})})) . \Pi_{i=1}^3 (\Pi_{j=1}^3 (h^{t_i t_j \gamma_{ij}})) \\ &= g^{t_3} \mathcal{F}(M)^{t_4} (u^{-1})^{t_2}. \end{aligned}$$

(v) Output a group signature:

$$\sigma = (\{\sigma_i\}_{i=1}^4, \pi_1, \pi_2) \in \mathbb{G}^6$$

Notice that the attribute certificates are only used in computing ρ_1 value. Also notice that the signature size is independent of number of attributes $|\zeta|$.

- `Verify`$(gpk, M, \Upsilon, \sigma)$: It verifies the group signature's validity as follows,

$$e(Z, \sigma_1)e(\sigma_2, \sigma_1) = A_T e(h, \pi_1) \tag{5.3}$$

$$e(g, \sigma_3)e(\mathcal{F}(M), \sigma_4)e(u^{-1}, \sigma_2) = e(h, \pi_2) \tag{5.4}$$

Returns 1 if the above equations holds, else return 0.

Equation (5.3) establishes that the signer is a valid member holding required attributes that satisfies the predicate Υ and equation (5.4) establishes that the group signature is on message M.

- $\texttt{Open}(gpk, ok_{user}, \sigma)$: Parse the signature and get σ_2. Calculate $(\sigma_2)^q$ and tests:

$$(\sigma_2)^q = (g^{s_{\text{id}}} h^{t_1})^q \stackrel{?}{=} (g^{s_{\text{id}}})^q$$

All the $(g^{s_{\text{id}}})^q$ can be pre-computed and stored as a list by the opener. It returns the corresponding id if it matches any such value from the list, else returns 0. Time to find the identity id is linearly dependent on the number of initial users.

5.3.1 Security Analysis

Theorem 5.3.1 *The proposed ABGS scheme is correct.*

Proof The correctness follows from the Groth-Sahai proof system correctness.

Theorem 5.3.2 *The proposed ABGS scheme preserves attribute anonymity.*

Proof According to the definition of attribute anonymity it is sufficient to show that for any predicate Υ and for any subset of attributes $\mathcal{A}_{\text{id}} : \exists \zeta_1, \zeta_2 \subseteq \mathcal{A}_{\text{id}}$ that satisfies predicate i.e., $\Upsilon(\zeta_1) = \Upsilon(\zeta_2) = 1$, the output of $\texttt{Sign}(gpk, k_{\text{id}}, \zeta_1, M, \Upsilon)$ is indistinguishable from the output of $\texttt{Sign}(gpk, k_{\text{id}}, \zeta_2, M, \Upsilon)$, subject to the constraint that they pass the verification algorithm.
For any group signature $\sigma = (\{\sigma_i\}_{i=1}^4, \pi_1, \pi_2)$, the attribute certificates are hidden

and used in σ_1 computation and it is easy to observe that for any two given group signatures by the same user the value $\sigma_1 = (g_{\mathrm{id}} h^{t_1})^{s_T}$ will not distinguish among themselves, since both are identical because of same s_T value. Thus it will not reveal the underlying subset of attributes, but only it proves that it satisfies the predicate. Thus the proposed scheme preserves attribute anonymity.

Theorem 5.3.3 *The proposed ABGS scheme preserves user anonymity under subgroup decision assumption.*

Proof The user anonymity definition says that any PPT adversary should not be able to link a group signature to the user and the witness-indistinguishable feature of Groth-Sahai proof system ensures that the proof will not reveal which of the witnesses the prover has used. Thus the proof follows from the proof of composable witness-indistinguishability of Groth-Sahai proof system [68].

Thus the Construction - 1 preserves full anonymity.

Theorem 5.3.4 *The proposed ABGS scheme satisfies traceability under the chosen message existential unforgeability of the two-level signature scheme.*

Proof We note that our ABGS scheme is an extension form of a two-level signature scheme and the proof is similar to that of [34]. Intuitively, we prove that our ABGS is secure against chosen message attack by using two-level signature's unforgeability. Suppose there exists a simulator $\mathcal{B}$ who interacts with the adversary $\mathcal{A}$ and wants to break the two-level signature scheme. Then, $\mathcal{B}$ executes the following algorithms and plays a game with $\mathcal{A}$.

Setup: The simulator $\mathcal{B}$ is given the factorization $n = pq$ of the group order $|\mathbb{G}| = n$. As usual, $\mathbb{G}_p$ and $\mathbb{G}_q$ denotes the subgroups of $\mathbb{G}$ of order p and q, respectively, and by analogy let $\mathbb{G}_{T_p}$ and $\mathbb{G}_{T_q}$ denotes the subgroups of $\mathbb{G}_T$ of order p and q, respectively. $\mathcal{B}$ is given the two-level signatures scheme public parameters,

$$\tilde{PP} = \{\tilde{g}, \tilde{Z} = \tilde{g}^z, \tilde{u} = \tilde{g}^y, \tilde{v}' = \tilde{g}^{z'}, \tilde{v}_1 = \tilde{g}^{z_1}, ..., \tilde{v}_{m'} = \tilde{g}^{z_{m'}}, \tilde{A} = e(\tilde{g}, \tilde{g})^{\alpha}\} \in \mathbb{G}_p^{m'+4} \times \mathbb{G}_{T_p}$$

Using this $\mathcal{B}$ simulates ABGS scheme. The parameters of ABGS scheme is generated as follows,

- Select the generators $(h, f, \gamma, \nu', \nu_1, ..., \nu_{m'}) \in \mathbb{G}_q^{m'+4}$ and two random exponents $\beta, \psi \in \mathbb{Z}_q$.
- Define the attribute set $Att = \{att_1, ..., att_m\}$.
- Choose secret value for attributes $S = \{s_j\}_{j=1}^m \in \mathbb{Z}_q^*$. Compute the public values as $\{h_{att_j} = h^{s_j}\}_{j=1}^m$.

$\mathcal{B}$ publishes the group public key, $gpk = (g = \tilde{g}f, u = \tilde{u}\gamma, Z = \tilde{Z}.f^{\psi}, h, \mathcal{F} = \{v' = \tilde{v}'\nu', v_1 = \tilde{v}_1\nu_1, ..., v_{m'} = \tilde{v}_{m'}\nu_{m'}\}, \{h_{att_j}\}_{j=1}^m)$.
$\mathcal{B}$ knows all the values except z and $\mathcal{B}$ has access to the two-level signature scheme's oracles namely, *key extraction* oracle and *signing* oracle. The distribution of the public key is the same as in the real scheme. $\mathcal{A}$ is given $ok_{user} = q$.

Queries: $\mathcal{A}$ is given access to the oracles $\mathsf{Join}(gpk, ik, ., .)$ and $\mathsf{Sign}(gpk, ik, ., ., .)$. The implementation of oracles is as follows,

- $\mathsf{Join}(gpk, ik, ., .)$: To answer the query to this oracle upon receiving $\mathrm{id}, \mathcal{A}_{\mathrm{id}} \subseteq Att$. $\mathcal{B}$ queries the *key extraction* oracle of two-level signature scheme and obtain the user's private key $\tilde{k}_{\mathrm{id}} = (\tilde{k}_{\mathrm{id},1}, \tilde{k}_{\mathrm{id},2}, \tilde{k}_{\mathrm{id},3}) \in \mathbb{G}_p^3$. Next, the simulator $\mathcal{B}$ internally associates a persistent random $r_{\mathrm{id}} \in \mathbb{Z}_q$ to id, recalling the value previously associated to id from storage as needed. Then $\mathcal{B}$ computes the requested key as, $k_{\mathrm{id}} = (k_{\mathrm{id},1} = \tilde{k}_{\mathrm{id},1}.(f^{\beta})^{\frac{1}{\psi+r_{\mathrm{id}}}}, k_{\mathrm{id},2} = \tilde{k}_{\mathrm{id},2}.f^{r_{\mathrm{id}}}, k_{\mathrm{id},3} = \tilde{k}_{\mathrm{id},3}.\gamma^{r_{\mathrm{id}}}, k_{\mathrm{id},4} = \{k_{\mathrm{id},1}^{s_j}\}_{\forall att_j \in \mathcal{A}_{\mathrm{id}}})$.
  Notice that this is a well formed key in our scheme. Indeed, since $\tilde{g} \in \mathbb{G}_p$ and $h \in \mathbb{G}_q$, it follows that $e(\tilde{g}, h) = 1$ in $\mathbb{G}_T$, and hence,

$$e(Zk_{\mathrm{id},2}, k_{\mathrm{id},1}) = e(\tilde{g}^{z+s_{id}} f^{\psi+r_{id}}, \tilde{g}^{\frac{\alpha}{z+s_{id}}} f^{\frac{\beta}{\psi+r_{id}}}) = e(\tilde{g}, \tilde{g})^{\alpha} e(f, f)^{\beta} = e(g, g)^{\alpha},$$

  and similarly, $e(k_{id,2}, u) = e(k_{id,3}, g)$.

- $\mathsf{Sign}(gpk, ik, ., ., .)$: To answer the query to this oracle upon receiving user identity id, a message $M = (\mu_1, ..., \mu_{m'}) \in \{0,1\}^{m'}$ and a predicate Υ. $\mathcal{B}$ queries to the *signing* oracle of two-level signature scheme and obtains $\sigma = (\sigma_1^\star, \sigma_2^\star, \sigma_3^\star)$ corresponding with (id, M). Next, the simulator $\mathcal{B}$ creates or recalls the persistent random value $r_{id} \in \mathbb{Z}_q$ associated to id, as described above. $\mathcal{B}$ chooses a random exponent $r_0 \in \mathbb{Z}_q$ and let $\mathcal{T}_\Upsilon = (\{s_{d_j}\}_{d_j \in D_T}, A_T = A^{s_T}, T^{\text{ext}})$ be the public values of the queried predicate Υ and note that $\mathcal{B}$ knows the value s_T. $\mathcal{B}$ then creates an unblined signature,

$$\rho = \left(\rho_1 = (\sigma_1^\star)^{s_T}.(f^\beta)^{\frac{s_T}{\psi + r_{id}}}, \rho_2 = \sigma_2^\star.f^{r_{id}}, \rho_3 = \sigma_3^\star.\gamma^{r_{id}}.(\nu' \Pi_{i=1}^{m'} \nu_i^{\mu_i})^{r_0}, \rho_4 = \sigma_4^\star.f^{-r_0}\right).$$

  Notice that this is a valid unblind signature in our scheme. The simulator $\mathcal{B}$ can next simply commit the variables and constructs the NIWI proof of it as given in the signature procedure, and then give the resulting signature. We could see that this is a valid group signature. $\mathcal{A}$ could check its validity by using gpk and opens its identity by using $ok_{user} = q$.

Output: At some point, $\mathcal{A}$ outputs its forged signature $\sigma^* = (\sigma_1^*, \sigma_2^*, \sigma_3^*, \sigma_4^*, \pi_1^*, \pi_2^*)$ with $(\mathrm{id}^*, M^*, \Upsilon^*)$. According to the traceability game constraint, id^* should be excluded from key extraction queries and $(\mathrm{id}^*, M^*, \Upsilon^*)$ should not be queried from signing oracle before.
Then $\mathcal{B}$ generates $\lambda : \lambda \equiv 1(mod\ p)$ and $\lambda \equiv 0(mod\ q)$. Then, from π_1^*, π_2^* and the two verification equations, we obtain:

$$e(Z, \sigma_1^*)e(\sigma_2^*, \sigma_1^*) = A_T e(h, \pi_1^*)$$

$$e(g, \sigma_3^*)e(\mathcal{F}(M), \sigma_4^*)e(u^{-1}, \sigma_2^*) = e(h, \pi_2^*)$$

And we use λ and s_T to obtain:

$$e(Z\sigma_2^{*\lambda}, {\sigma_1^*}^{\frac{\lambda}{s_T}}) = A$$

$$e(\sigma_2^{*\lambda}, u) \stackrel{?}{=} e(\sigma_3^{*\lambda}, g).e(\sigma_4^{*\lambda}, \mathcal{F}(M))$$

Since $({\sigma_1^*}^{\frac{\lambda}{s_T}}, \sigma_2^{*\lambda}, \sigma_3^{*\lambda}, \sigma_4^{*\lambda})$ pass the verification equation of two-level signature scheme, it is a forged two-level signature, which means $\mathcal{B}$ successfully breaks the unforgeability of two-level signature scheme. Hence the theorem is proved.

Theorem 5.3.5 *The proposed scheme preserves attribute unforgeability.*

Proof Lemma 5.3.6 implies this theorem.

Lemma 5.3.6 *Under the DL and KEA1 assumptions there exists no PPT adversary $\mathcal{A}$ which passes verification with forged attributes with non negligible probability.*

Proof The input to the simulator $\mathcal{B}$ is an instance of the DL problem, $(v, v') \in \mathbb{G}_q^2$, where q is prime. Let $\xi = log_v v'$.
Setup: According to scheme setup define the groups $\mathbb{G}_p, \mathbb{G}$ and $\mathbb{G}_T$, of order p, n and n, respectively, where $n = pq$. Define the universal set of attribute Att. Simulate the ABGS scheme and generate the parameters by setting $h = v$ and $g = v'g'$, where $g' \in_R \mathbb{G}_p$

$$gpk = (g, u, Z, h, \mathcal{F}, \{h_{att_i}\}_{att_i \in Att}), ik = (g^\alpha, z, S) \text{ and } ok_{user} = q \in \mathbb{Z}$$

Queries: As $\mathcal{B}$ knows all the keys, it can answer all the queries generated by an adversary $\mathcal{A}$. That is, $\mathcal{A}$ is given access to the oracles $\mathsf{Join}(gpk, ik, ., .)$ and $\mathsf{Sign}(gpk, ik, ., ., .)$.

Output: Finally, $\mathcal{A}$ outputs a signature σ^* with forged attributes on message M^*, a predicate Υ^* whose public values are $\mathcal{T}_{\Upsilon^*} = (\{s_{d_j}\}, A_T, T^{\text{ext}})$, and user's private key k_{ID^*} such that $\Upsilon(\mathcal{A}_{\text{ID}^*}) \neq 1$. As it is a valid signature and from (5.3) $\sigma_1^* = (g_{\text{ID}^*}^{s_T} h^{t_1})$. This can be viewed as $\sigma_1^* = (v'g')^{\frac{s_T}{z+s_{\text{ID}^*}}} h^{t_1} = (v')^{\frac{s_T}{z+s_{\text{ID}^*}}} X$ and ${v'}^{s_T}$ can be extracted by raising the power $z + s_{\text{ID}^*}$, where

$$X = (h^{t_1})(g')^{\frac{s_T}{s_{\text{ID}^*}+z}}$$

It is like $\mathcal{B}$ is giving input $(h = v, h_T = v^{s_T})$ to $\mathcal{A}$ and $\mathcal{A}$ implicitly returns $(v', (v')^{s_T})$. Then by KEA1 assumption, $\mathcal{B}$ can utilize the extractor $\bar{\mathcal{A}}$ to extract a value ξ. Under DL assumption it can be done with negligible probability. Thus the signature produced by the forged attributes can pass verification with negligible probability.
To be more particular, we can assume that the $\mathcal{A}$ is missing one attribute certificate to satisfy the predicate, say $g_{\mathrm{ID}^*}^{s_j}$, i.e. $(v')^{s_j}$ is unknown to $\mathcal{A}$, but he knows $h^{s_j} = v^{s_j}$. And $\mathcal{A}$ is producing it in forged group signature σ^*. Then similar to above from KEA1 and DL assumption it is negligible to produce such signatures.

Theorem 5.3.7 *The proposed scheme preserves collusion resistance of attributes.*

Proof Lemma 5.3.8 implies this theorem.

Lemma 5.3.8 *Even if some malicious participants* $\mathrm{U}_{\mathrm{id}_1}, ..., \mathrm{U}_{\mathrm{id}_k} (k > 1)$ *with the set of attributes* $\mathcal{A}_{\mathrm{id}_1}, ..., \mathcal{A}_{\mathrm{id}_k}$ *collude, they cannot make a valid signature associated with predicate* Υ, *where* $\Upsilon(\cup_{j=1}^{k} \mathcal{A}_{\mathrm{id}_j}) = 1$ *and* $\Upsilon(\mathcal{A}_{\mathrm{id}_j}) \neq 1$ *for* $j = 1, ..., k$ *with non negligible probability.*

Proof Without loss of generality we assume that $\mathrm{U}_{\mathrm{id}_0}$ with $\mathcal{A}_{\mathrm{id}_0}$ and $\mathrm{U}_{\mathrm{id}_1}$ with $\mathcal{A}_{\mathrm{id}_1}$ represent malicious participants. $\mathrm{U}_{\mathrm{id}_0}$ and $\mathrm{U}_{\mathrm{id}_1}$ attempt to make a valid signature associated with predicate Υ such that $\Upsilon(\mathcal{A}_{\mathrm{id}_0} \cup \mathcal{A}_{\mathrm{id}_1}) = 1$ and $\Upsilon(\mathcal{A}_{\mathrm{id}_0}) \neq 1, \Upsilon(\mathcal{A}_{\mathrm{id}_1}) \neq 1$. They can satisfy verification equation (5.4) since they have valid membership certificates. We assume that $g_{\mathrm{id}_0}^{t} = g_{\mathrm{id}_1}$, where $t \in \mathbb{Z}_n^*$. Note that the probability of $t = 1$ is negligible. And they try to compute

$$\rho_1 = g_{id_0}^{\Sigma_{\mathrm{att}_j \in \mathcal{A}_{\mathrm{id}_0}} \Delta_{\mathrm{att}_j} s_j} + g_{id_1}^{\Sigma_{\mathrm{att}_j \in \mathcal{A}_{\mathrm{id}_1}} \Delta_{\mathrm{att}_j} s_j} + g_{id_0}^{\Sigma_{d_j \in D_T^{\zeta}} \Delta_{d_j} s_{d_j}}$$

$$= g_{id_0}^{\Sigma_{\mathrm{att}_j \in \mathcal{A}_{\mathrm{id}_0}} \Delta_{\mathrm{att}_j} s_j + t\Sigma_{\mathrm{att}_j \in \mathcal{A}_{\mathrm{id}_1}} \Delta_{\mathrm{att}_j} s_j + \Sigma_{d_j \in D_T^{\zeta}} \Delta_{d_j} s_{d_j}}$$

Then from (2.6)

$$\Sigma_{\mathrm{att}_j \in \mathcal{A}_{\mathrm{id}_0}} \Delta_{\mathrm{att}_j} s_j + t\Sigma_{\mathrm{att}_j \in \mathcal{A}_{\mathrm{id}_1}} \Delta_{\mathrm{att}_j} s_j + \Sigma_{d_j \in D_T^{\zeta}} \Delta_{d_j} s_{d_j} \neq s_T$$

holds. Since $t \neq 1$ this means that they cannot collude.

Table 5.1: Comparison of ABGS scheme without non-frameability in standard model with other schemes

	Khader [72]	Emura et al. [51]	Herranz et al.[70]+ Boyen et al.[34]	Our Scheme
User anonymity	CPA	CCA2	CPA	CPA
Attribute anonymity	no	no	yes	yes
Non-frameability	no	yes	no	no
Signature length	O(Φ)	O(Φ)	$15\|\mathbb{G}_p\| + 6\|\mathbb{G}_n\| = O(1)$	$6\|\mathbb{G}_p\|$=O(1)
User′s Private Key Length	$(m'+1)\|\mathbb{G}_p\| + \|\mathbb{Z}_p^*\|$	$(m'+1)\|\mathbb{G}_p\| + 2\|\mathbb{Z}_p^*\|$	$(m+\hat{m}+3)\|\mathbb{G}_p\|$	$(3+\hat{m})\|\mathbb{G}\|$
Assumptions	DLDH, ℓ-SDH	DDH, ℓ-SDH, DL	DLin, (ℓ, m, t)-aMSE-CDH	$\begin{pmatrix}\text{SGD, } \ell-\text{HSDH,}\\ \text{DL, KEA1}\end{pmatrix}$
Model	RO	RO	Standard	Standard
Signing	$\begin{pmatrix}(7+2\Phi)\mathbb{G}_p+(5+\Phi)\mathbb{G}_T\\ +(\Phi+1)e\end{pmatrix}$	$\begin{pmatrix}(9+3\Phi)\mathbb{G}_p+(1+\Phi)\mathbb{G}_p\\ +8\mathbb{G}_T+3e\end{pmatrix}$	$\begin{pmatrix}(6m+6m'+\Phi(\Phi-1)\\ +14)\mathbb{G}_p\end{pmatrix}$	$(18+m'+2\Phi)\mathbb{G}$
Verification	$\begin{pmatrix}(6+2r)\mathbb{G}_p+(8+2\Phi)\mathbb{G}_T\\ +(\Phi+2r+1)e\end{pmatrix}$	$\begin{pmatrix}(11+2\Phi)\mathbb{G}_p+(\Phi+1)\mathbb{G}_2\\ +14\mathbb{G}_3+6e\end{pmatrix}$	$\begin{pmatrix}(4m+6m')\mathbb{G}_p\\ +33e+21\mathbb{G}_T\end{pmatrix}$	$(2+m')\mathbb{G}+3\mathbb{G}_T+6e$

Thus the Construction - 1 preserves full traceability.

5.3.2 Comparison

In the construction - 1, the group signature contains 6 elements from $\mathbb{G}$. In Table 5.1 we compare our proposed construction - 1 with the existing ABGS schemes. Let $\Phi = |\zeta|$, where ζ be the set of attributes associated with a signature and $m = |Att|$. Let $\hat{m} \leq m$ be the maximum number of attributes assigned to a user, r be the number of revoked members and m' is the message length. Let RO denotes the Random oracle model, SGD denotes the Subgroup Decision assumption, (ℓ, m, t)-aMSE-CDH denotes the (ℓ, m, t)- augmented multi-sequence of exponents computational Diffie-Hellman and let e represents the bilinear operation. We note that the verification cost of the proposed scheme is independent of the number of attributes, where as in other schemes the verification cost is linear in terms of the number of attributes. Also the scheme is in standard model and has comparatively short signature length.

5.4 Construction - 2

In this section, we describe our second construction for ABGS scheme with attribute anonymity having shorter signature length [3]. But here our construction is based on the two-level signature scheme by Liang et al. [79]. We prove that our scheme, in the standard model, preserves attribute anonymity unconditionally, user anonymity in

CPA attacks under Subgroup Decision assumption, traceability under ℓ-MOMSDH assumption and attribute unforgeability under DL and KEA1 assumptions. When compared to the Construction - 1 the signature length in this construction is shorter.

- `Setup`(1^k): It takes the security parameter k as an input and outputs the system parameters $params$, the group public key gpk, issuing key ik and the opening key ok_{user}.

  (i) Select the primes p and q of size k. Let the groups $\mathbb{G}$ and $\mathbb{G}_T$ be of order $n = pq$ for which there exists a bilinear map e from $\mathbb{G} \times \mathbb{G}$ to $\mathbb{G}_T$. Let $\mathbb{G}_p$ and $\mathbb{G}_q$ be the subgroups of $\mathbb{G}$ of order p and q, respectively.

  (ii) Define the universal set of attributes, $Att = \{\text{att}_1, ..., \text{att}_m\}, m = O(k)$.

  (iii) Define the collision resistant hash function, $\mathcal{H} : \{0,1\}^{m'} \rightarrow Z_n$, $m' = O(k)$.

  (iv) Select the generators $g, u \in \mathbb{G}$ and $h \in \mathbb{G}_q$.

  (v) For each attribute att_i select the attribute secret $s_i \in \mathbb{Z}_q^*$. Let $S = \{s_i\}_{att_i \in Att}$.

  (vi) Compute the public values of the attributes $\{h_{att_i} = h^{s_i}\}_{att_i \in Att}$.

  (vii) Select the secret $z \in_R \mathbb{Z}_n^*$ and compute $Z = g^z$.

  (viii) Output the system parameters,

  $$params = (n, \mathbb{G}, \mathbb{G}_T, e, Att, \mathcal{H})$$

  The group public key,

  $$gpk = (g, u, h, Z, \{h_{att_i}\}_{att_i \in Att})$$

  The issuing key $ik = (z, S)$, the opening key $ok_{user} = q$ and the size $= \ell$.

  The gpk also include the description of $(n, \mathbb{G}, \mathbb{G}_T, e), Att, \mathcal{H}$.

- `Join`$(gpk, ik, \text{id}, A_{\text{id}})$: It takes group public key, issuing key, user identity id and subset of attributes $\mathcal{A}_{\text{id}} \subseteq Att$, and outputs user private key k_{id}.

(i) Select the secret unique identifier $s_{\mathrm{id}} \in \mathbb{Z}_n^*$.

(ii) Compute $g_{\mathrm{id}} = g^{\frac{1}{s_{\mathrm{id}}+z}}$.

(iii) Compute the attribute certificates $\{g_{\mathrm{id},i} = g_{\mathrm{id}}^{s_i}\}_{\mathrm{att}_i \in \mathcal{A}_{\mathrm{id}}}$.

(iv) Output the user private key,

$$k_{\mathrm{id}} = (k_{\mathrm{id}}^{(1)}, k_{\mathrm{id}}^{(2)}, k_{\mathrm{id}}^{(3)}) = (s_{\mathrm{id}}, g_{\mathrm{id}}, \{g_{\mathrm{id},i}\}_{\mathrm{att}_i \in \mathcal{A}_{\mathrm{id}}})$$

- `BuildTree`(gpk, ik, Υ): It generates a public values for the predicate Υ.

(i) Let T_Υ be the tree that represents the predicate Υ.

(ii) Get extension tree $T^{\mathrm{ext}} \leftarrow$ `AddDummyNode`$(\mathtt{T}_\Upsilon)$.

(iii) Get secret values for each dummy node and the secret value of root of T^{ext} using $(\{s_{d_j}\}_{d_j \in D_T}, s_T) \leftarrow$ `AssignedVaule`(S, T^{ext}).

(iv) Output the public values of tree T_Υ.

$$\mathcal{T}_\Upsilon = (\{s_{d_j}\}_{d_j \in D_T}, h_T = h^{s_T}, g_T = g^{s_T}, T^{\mathrm{ext}})$$

- `Sign`$(gpk, k_{\mathrm{id}}, \zeta, M, \Upsilon)$: It generates a group signature σ on message $M \in \{0,1\}^{m'}$ with user private key k_{id} who satisfy the predicate Υ with his subset of attributes $\zeta \subseteq \mathcal{A}_{\mathrm{id}} : \Upsilon(\zeta) = 1$.

(i) Get the public values of Υ from the public repository[1].

(ii) Compute $\rho = (\rho_1, \rho_2, \rho_3) = (g^{s_{\mathrm{id}}}, g_{\mathrm{id}}^{s_T}, u^{\frac{1}{s_{\mathrm{id}}+\mathcal{H}(M)}})$
where $\rho_2 = g_{\mathrm{id}}^{s_T}$ is computed as follows,

- Select $\zeta \subseteq \mathcal{A}_{\mathrm{id}} : \Upsilon(\zeta) = 1$.
- Get $(\{\Delta_{\mathrm{att}_j}\}_{(\forall \mathrm{att}_j \in \zeta)}, \{\Delta_{d_j}\}_{(\forall d_j \in D_T^\zeta)}) \leftarrow$ `MakeSimplifiedTree`(ζ, T^{ext}).
- Compute $\rho_2 = \Pi_{\mathrm{att}_i \in \zeta} g_{\mathrm{id},i}^{\Delta_{\mathrm{att}_i}} g_{\mathrm{id}}^{(\Sigma_{d_j \in D_T^\zeta} \Delta_{d_j} s_{d_j})} = g_{\mathrm{id}}^{\Sigma_{\mathrm{att}_i \in \zeta} \Delta_{\mathrm{att}_i} s_i} g_{\mathrm{id}}^{\Sigma_{d_j \in D_T^\zeta} \Delta_{d_j} s_{d_j}} = g_{\mathrm{id}}^{s_T}$.

[1] `GM` runs `BuildTree` algorithm to generate the public values of the predicate Υ and stores it in a public repository. Note that if the public values of the required predicate is present in the public repository then the user will not approach `GM`.

We hide the ρ values as follows:

(iii) Choose $t_1, t_2, t_3 \in_R \mathbb{Z}_n$, and compute $\sigma_1 = \rho_1 h^{t_1}, \sigma_2 = \rho_2 h^{t_2}$ and $\sigma_3 = \rho_3 h^{t_3}$.

(iv) Compute $\pi_1 = \rho_2^{t_1}(Z\rho_1)^{t_2} h^{t_1 t_2}, \pi_2 = \rho_3^{t_1}(g^{\mathcal{H}(M)}\rho_1)^{t_3} h^{t_1 t_3}$.

(v) Output a group signature:

$$\sigma = (\{\sigma_i\}_{i=1}^{3}, \pi_1, \pi_2) \in \mathbb{G}^5$$

Notice that the attribute certificates are used only in computing ρ_2. Also notice that the signature size is independent of number of attributes $|\zeta|$.

- `Verify`$(gpk, M, \Upsilon, \sigma)$:

  (i) Compute $T_1 = e(\sigma_1 Z, \sigma_2) e(g, g_T)^{-1}$ and
  $T_2 = e(\sigma_1 g^{\mathcal{H}(M)}, \sigma_3) e(g, u)^{-1}$

  (ii) Verify the following equations:

$$T_1 \stackrel{?}{=} e(\pi_1, h) \tag{5.5}$$

$$T_2 \stackrel{?}{=} e(\pi_2, h) \tag{5.6}$$

  Returns 1 if the above equations holds, else return 0.
  Equation (5.5) establishes that the signer is a valid member holding the required attributes that satisfies the predicate Υ and equation (5.6) establishes that the group signature is on the message M

- `Open`(gpk, ok_{user}, σ): Parse the signature and get σ_1. Calculate $(\sigma_1)^q$ and test:

$$(\sigma_1)^q = (g^{s_{\mathrm{id}}} h^{t_1})^q \stackrel{?}{=} (g^{s_{\mathrm{id}}})^q$$

  All the $(g^{s_{\mathrm{id}}})^q$ can be pre-computed and stored as a list by the opener. It returns the corresponding id if it matches any such value from the list, else returns 0. Time to find the identity id is linearly dependent on the number of initial users.

5.4.1 Security Analysis

Theorem 5.4.1 *The proposed ABGS scheme is correct.*

Proof The correctness is followed from the scheme.

Theorem 5.4.2 *The proposed ABGS scheme satisfies traceability under the chosen message existential unforgeability of the Liang et al.'s two-level signature scheme.*

Proof We note that our ABGS scheme is an extension form of a two-level signature scheme and the proof is similar to that of [79]. Intuitively, our ABGS is secure against chosen message attack by using two-level signature's unforgeability.
Suppose there exists a simulator $\mathcal{B}$ who interacts with the adversary $\mathcal{A}$ and wants to break the two-level signature scheme. Then, $\mathcal{B}$ executes the following algorithms and plays a game with $\mathcal{A}$.

Setup: $\mathcal{B}$ is given the public parameters of two-level signature scheme:

$$params = \{p, q, n, \mathbb{G}, \mathbb{G}_T, \mathbb{G}_p, e, \mathcal{H}\}, PP = \{g, u, Z = g^z\}$$

Using this $\mathcal{B}$ simulates ABGS scheme. The parameters of ABGS scheme is generated as follows,

- Select the generator $h \in \mathbb{G}_q$, where $\mathbb{G}_q$ is a subgroup of $\mathbb{G}$ and of order q.
- Define the attribute set $Att = \{att_1, ..., att_m\}$.
- Choose secret value for attributes $S = \{s_j\}_{j=1}^m \in \mathbb{Z}_q^*$. Compute the public values as $\{h_{att_j} = h^{s_j}\}_{j=1}^m$.

$\mathcal{B}$ publishes the group public key, $gpk = (g, u, h, Z, \{h_{att_j}\}_{j=1}^m)$.
$\mathcal{B}$ knows all the values except z, for which it queries the two-level signature scheme's oracles namely, key extraction oracle and signing oracle. $\mathcal{A}$ is given $ok_{user} = q$.

Queries: $\mathcal{A}$ is given access to the oracles $\mathsf{Join}(gpk, ik, ., .)$ and $\mathsf{Sign}(gpk, ik, ., ., .)$. The implementation of oracles is as follows,

- $\mathsf{Join}(gpk, ik, ., .)$: To answer the query to this oracle upon receiving $\mathrm{id}, \mathcal{A}_{\mathrm{id}} \subseteq Att$. $\mathcal{B}$ queries the key extraction oracle of two-level signature scheme and obtain the user's private key $k_{\mathrm{id}} = (k_{\mathrm{id},1}, k_{\mathrm{id},2}) = (s_{\mathrm{id}}, g_{\mathrm{id}} = g^{\frac{1}{z+s_{\mathrm{id}}}})$. Then $\mathcal{B}$ computes $k_{\mathrm{id}}^{(3)} = \{g_{\mathrm{id},j} = g_{\mathrm{id}}^{s_j}\}_{\forall att_j \in \mathcal{A}_{\mathrm{id}}}$ and replies with $k_{\mathrm{id}} = (k_{\mathrm{id}}^{(1)}, k_{\mathrm{id}}^{(2)}, k_{\mathrm{id}}^{(3)}) = (s_{\mathrm{id}}, g_{\mathrm{id}}, \{g_{\mathrm{id},i}\}_{\mathrm{att}_i \in \mathcal{A}_{\mathrm{id}}})$. Notice that $\mathcal{A}$ can make valid group signatures with this key.

- $\mathsf{Sign}(gpk, ik, ., ., .)$: To answer the query to this oracle upon receiving user identity id, a message M and a predicate Υ. $\mathcal{B}$ queries to the signing oracle of two-level signature scheme and obtains $\sigma = (\sigma_1^\star, \sigma_2^\star, \sigma_3^\star)$ corresponding with (id, M). Let $\mathcal{T}_\Upsilon = (\{s_{d_j}\}_{d_j \in D_T}, h_T = h^{s_T}, g_T = g^{s_T}, T^{\mathrm{ext}})$ be the public values of the queried predicate Υ and note that $\mathcal{B}$ knows the value s_T. Then, $\mathcal{B}$ randomly choose t_1, t_2, t_3, and generates the group signature, $\sigma = (\sigma_1^\star h^{t_1}, (\sigma_2^\star)^{s_T} h^{t_2}, \sigma_3^\star h^{t_3}, (\sigma_2^\star)^{s_T t_1} (Z\sigma_1^\star)^{t_2} h^{t_1 t_2}, (\sigma_3^\star)^{t_1} (g^{\mathcal{H}(M)} \sigma_1^\star)^{t_3} h^{t_1 t_3})$.
  We could see that this a valid group signature. $\mathcal{A}$ could check its validity by using gpk and opens its identity by using $ok_{user} = q$.

Output: At some point, $\mathcal{A}$ outputs its forged signature $\sigma^* = (\sigma_1^*, \sigma_2^*, \sigma_3^*, \pi_1^*, \pi_2^*)$ with $(\mathrm{id}^*, M^*, \Upsilon^*)$. According to the traceability game constraint, id^* should be excluded from key extraction queries and $(\mathrm{id}^*, M^*, \Upsilon^*)$ should not be queried from signing oracle before.
Then $\mathcal{B}$ generates $\lambda : \lambda \equiv 1 (mod\ p)$ and $\lambda \equiv 0 (mod\ q)$. Then, from π_1^*, π_2^* and the first two verification equations, we obtain:

$$e(\sigma_1^* Z, \sigma_2^*) e(g, g_T)^{-1} = e(\pi_1^*, h_T)$$

$$e(\sigma_1^* g^{\mathcal{H}(M^*)}, \sigma_3^*) e(g, g)^{-1} = e(\pi_2^*, h)$$

And we use λ to obtain:

$$e(\sigma_1^{*\lambda} Z, \sigma_2^{*\frac{\lambda}{s_T}}) = e(g, g)$$

$$e(\sigma_1^{*\lambda} g^{\mathcal{H}(M^*)}, \sigma_3^{*\lambda}) = e(g, g)$$

Since $(\sigma_1^{*\lambda}, \sigma_2^{*\frac{\lambda}{s_T}}, \sigma_3^{*\lambda})$ pass the verification equation of two-level signature scheme, it is a forged two-level signature, which means $\mathcal{B}$ successfully breaks the unforgeability of two-level signature scheme. Hence the theorem is proved.

Theorem 5.4.3 *The proposed scheme preserves attribute unforgeability.*

Proof Lemma 5.4.4 implies this theorem.

Lemma 5.4.4 *Under the DL and KEA1 assumptions there exists no PPT adversary $\mathcal{A}$ which passes verification with forged attributes with non negligible probability.*

Proof The input to the simulator $\mathcal{B}$ is an instance of the DL problem, $(v, v') \in \mathbb{G}_q^2$, where q is prime. Let $\xi = log_v v'$.
Setup: According to scheme setup define the groups $\mathbb{G}_p, \mathbb{G}$ and $\mathbb{G}_T$, of order p, n and n, respectively, where $n = pq$. Define the universal set of attribute Att. Simulate the ABGS scheme and generate the parameters by setting $h = v$ and $g = v'g'$, where $g' \in_R \mathbb{G}_p$

$$gpk = (g, u, h, Z, \{h_{att_i}\}_{att_i \in Att}), ik = (z, S) \text{ and } ok_{user} = q$$

Queries: As $\mathcal{B}$ knows all the keys, it can answer all the queries generated by an adversary $\mathcal{A}$. That is, $\mathcal{A}$ is given access to the oracles $\mathsf{Join}(gpk, ik, ., .)$ and $\mathsf{Sign}(gpk, ik, ., ., .)$.

Output: Finally, $\mathcal{A}$ outputs a signature σ^* with forged attributes on message M^*, a predicate Υ^* whose public values are $\mathcal{T}_{\Upsilon^*} = (\{s_{d_j}\}, h_T, g_T, T^{\text{ext}})$, and user's private key k_{ID^*} such that $\Upsilon(\mathcal{A}_{\text{ID}^*}) \neq 1$. As it is a valid signature and from (5.5) $\sigma_2^* = (g_{\text{ID}^*} h^{t_2})^{s_T}$. This can be viewed as $\sigma_2^* = (v'g')^{\frac{s_T}{z+s_{\text{ID}^*}}} h^{t_2} = (v')^{\frac{s_T}{z+s_{\text{ID}^*}}} X$ and v'^{s_T} can be extracted by raising the power $z + s_{\text{ID}^*}$, where

$$X = h^{t_2} (g')^{\frac{s_T}{s_{\text{ID}^*}+z}}, h^{t_2} = \frac{\sigma_2^*}{g_{\text{ID}^*}}$$

Note that $\mathcal{B}$ knows g_{ID^*}. It is like $\mathcal{B}$ is giving input $(h = v, h_T = v^{s_T})$ to $\mathcal{A}$ and $\mathcal{A}$ implicitly returns $(v', (v')^{s_T})$. Then by KEA1 assumption, $\mathcal{B}$ can utilize the extractor $\bar{\mathcal{A}}$ to extract a value ξ. Under DL assumption it can be done with negligible probability. Thus the signature produced by the forged attributes can pass verification with negligible probability.
To be more particular, we can assume that the $\mathcal{A}$ is missing one attribute certificate to satisfy the predicate, say $g_{\mathrm{ID}^*}^{s_j}$, i.e. $(v')^{s_j}$ is unknown to $\mathcal{A}$, but he knows $h^{s_j} = v^{s_j}$. And $\mathcal{A}$ is producing it in forged group signature σ^*. Then similar to above from KEA1 and DL assumption it is negligible to produce such signatures.

Theorem 5.4.5 *The proposed scheme preserves collusion resistance of attributes.*

Proof Lemma 5.4.6 implies this theorem.

Lemma 5.4.6 *Even if some malicious participants* $\mathrm{U}_{\mathrm{id}_1}, ..., \mathrm{U}_{\mathrm{id}_k} (k > 1)$ *with the set of attributes* $\mathcal{A}_{\mathrm{id}_1}, ..., \mathcal{A}_{\mathrm{id}_k}$ *collude, they cannot make a valid signature associated with predicate* Υ, *where* $\Upsilon(\cup_{j=1}^{k} \mathcal{A}_{\mathrm{id}_j}) = 1$ *and* $\Upsilon(\mathcal{A}_{\mathrm{id}_j}) \neq 1$ *for* $j = 1, ..., k$ *with non negligible probability.*

Proof Without loss of generality we assume that $\mathrm{U}_{\mathrm{id}_0}$ with $\mathcal{A}_{\mathrm{id}_0}$ and $\mathrm{U}_{\mathrm{id}_1}$ with $\mathcal{A}_{\mathrm{id}_1}$ represent malicious participants. $\mathrm{U}_{\mathrm{id}_0}$ and $\mathrm{U}_{\mathrm{id}_1}$ attempt to make a valid signature associated with predicate Υ such that $\Upsilon(\mathcal{A}_{\mathrm{id}_0} \cup \mathcal{A}_{\mathrm{id}_1}) = 1$ and $\Upsilon(\mathcal{A}_{\mathrm{id}_0}) \neq 1, \Upsilon(\mathcal{A}_{\mathrm{id}_1}) \neq 1$. They can satisfy verification equation (5.6) since they have valid s_{id}. We assume that $g_{\mathrm{id}_0}^{t} = g_{\mathrm{id}_1}$, where $t \in \mathbb{Z}_n^*$. Note that the probability of $t = 1$ is negligible. And they tries to compute

$$\rho_2 = g_{id_0}^{\Sigma_{\mathrm{att}_j \in \mathcal{A}_{\mathrm{id}_0}} \Delta_{\mathrm{att}_j} s_j} + g_{id_1}^{\Sigma_{\mathrm{att}_j \in \mathcal{A}_{\mathrm{id}_1}} \Delta_{\mathrm{att}_j} s_j} + g_{id_0}^{\Sigma_{d_j \in D_T^{\zeta}} \Delta_{d_j} s_{d_j}}$$

$$= g_{id_0}^{\Sigma_{\mathrm{att}_j \in \mathcal{A}_{\mathrm{id}_0}} \Delta_{\mathrm{att}_j} s_j + t\Sigma_{\mathrm{att}_j \in \mathcal{A}_{\mathrm{id}_1}} \Delta_{\mathrm{att}_j} s_j + \Sigma_{d_j \in D_T^{\zeta}} \Delta_{d_j} s_{d_j}}$$

Then from (2.6)

$$\Sigma_{\mathrm{att}_j \in \mathcal{A}_{\mathrm{id}_0}} \Delta_{\mathrm{att}_j} s_j + t\Sigma_{\mathrm{att}_j \in \mathcal{A}_{\mathrm{id}_1}} \Delta_{\mathrm{att}_j} s_j + \Sigma_{d_j \in D_T^{\zeta}} \Delta_{d_j} s_{d_j} \neq s_T$$

holds. Since $t \neq 1$ this means that they cannot collude.

Thus the Construction - 2 preserves full traceability.

Theorem 5.4.7 *The proposed ABGS scheme preserves attribute anonymity.*

Proof According to the definition of attribute anonymity it is sufficient to show that for any predicate Υ and for any subset of attributes $\mathcal{A}_{\text{id}} : \exists \zeta_1, \zeta_2 \subseteq \mathcal{A}_{\text{id}}$ that satisfies predicate i.e., $\Upsilon(\zeta_1) = \Upsilon(\zeta_2) = 1$, the output of $\texttt{Sign}(gpk, k_{\text{id}}, \zeta_1, M, \Upsilon)$ is indistinguishable from the output of $\texttt{Sign}(gpk, k_{\text{id}}, \zeta_2, M, \Upsilon)$, subject to the constraint that they pass the verification algorithm.

For any group signature $\sigma = (\{\sigma_i\}_{i=1}^{3}, \pi_1, \pi_2)$, the attribute certificates are hidden and used in σ_2 computation and it is easy to observe that for any two given group signatures by the same user the value $\sigma_2 = (g_{\text{id}} h^{t_2})^{s_T}$ will not distinguish among themselves, since both are identical because of same s_T value. Thus it will not reveal the underlying subset of attributes, but only it proves that it satisfies the predicate. Thus the proposed scheme preserves attribute anonymity.

Theorem 5.4.8 *Suppose no t-time adversary can solve the subgroup decision problem with advantage at least ϵ_{sub}. Then every t'-time adversary $\mathcal{A}$ which breaks the user anonymity, we have that $Adv_{\mathcal{A}} < 2\epsilon_{sub}$, where $t \approx t'$.*

Proof We use a game technique where G_0 is the real ABGS user anonymity game, and G_1 is a game in which the public parameters are the same as in the original game except that h is chosen randomly from $\mathbb{G}$ instead of $\mathbb{G}_q$. Let the adversary's advantage in the original game be $Adv_{\mathcal{A}}$, and in the modified game be $Adv_{\mathcal{A},G_1}$.
First, in Lemma 5.4.9, we show that the two games are indistinguishable, unless the subgroup decision assumption is easy. Second, in Lemma 5.4.10, we use an information-theoretic argument to prove that in the game G_1 the adversary's advantage is zero. Then the theorem follows from these results. The proofs of these two lemmas are similar to that of [67; 79] and [34].

Lemma 5.4.9 *For all t'-time adversaries as above, $Adv_{\mathcal{A}} - Adv_{\mathcal{A},G_1} < 2\epsilon_{sub}$.*

Proof Suppose there is a simulator $\mathcal{B}$ trying to solve subgroup decision problem. Upon receiving a subgroup decision challenge $(n, \mathbb{G}, \mathbb{G}_T, e, w)$ the simulator $\mathcal{B}$ first creates public parameters for the ABGS scheme by setting $h = w$ and then choosing the remaining public parameters exactly as in the ABGS scheme. It then sends the public information to $\mathcal{A}$ and plays the user anonymity game with it. If $w \in_R \mathbb{G}_q$ then the game being played is the normal user anonymity game; otherwise, if $w \in_R \mathbb{G}$, then the game played is a different game we call G_1. In either case, the algorithm $\mathcal{B}$ will be able to answer all queries, since it knows the issuing key.
At some point the adversary will choose a message M and two identities ID_1 and ID_2 it wishes to be challenged upon, under the usual constraints that it had not previously made a signing key query on ID_x or a signature query on (ID_x, M). The simulator $\mathcal{B}$ will create the requisite challenge signature on M and $\mathcal{A}$ will guess the identity of the signer. If $\mathcal{A}$ answers correctly, then $\mathcal{B}$ outputs $b = 1$, to signify that $w \in \mathbb{G}_q$; otherwise it outputs $b = 0$, to signify that $w \in \mathbb{G}$.
Let $Adv_{\mathcal{B}}$ be the advantage of the simulator $\mathcal{B}$ in the subgroup decision game. As we know that,

$$Pr[w \in \mathbb{G}] = Pr[w \in \mathbb{G}_q] = \frac{1}{2}$$

we deduce that,

$$\begin{aligned} Adv_{\mathcal{A}} - Adv_{\mathcal{A},G_1} &= Pr[b = 1 | w \in \mathbb{G}_q] - Pr[b = 1 | w \in \mathbb{G}] \\ &= 2Pr[b = 1, w \in \mathbb{G}_q] - 2Pr[b = 1, w \in \mathbb{G}] \\ &= 2Adv_{\mathcal{B}} < 2\epsilon_{sub} \end{aligned}$$

Under the hardness of subgroup decision assumption $Adv_{\mathcal{B}}$ must be lesser than ϵ_{sub}, given that $\mathcal{B}$ runs in time $t \approx t'$.

Lemma 5.4.10 *For any algorithm $\mathcal{A}$, we have $Adv_{\mathcal{A},G_1} = 0$.*

Proof We prove that when h is chosen uniformly from $\mathbb{G}$ at random, instead of $\mathbb{G}_q$, the adversary $\mathcal{A}$ can not sense the identity from the challenge signature. Although

the tracing value s_{ID_x} may have been used to answer previous signing queries on (ID_x, M), the challenge signature is statistically independent of the real identity.
Let the challenge group signature is $\sigma = (\sigma_1, \sigma_2, \sigma_3, \pi_1, \pi_2)$. Since the signature values $\sigma_1, \sigma_2, \sigma_3$ are blinded with random number $h^{t_1}, h^{t_2}, h^{t_3} \in \mathbb{G}$, they reveal nothing about the identity. Then, we give two signatures: σ with (ID_1, M, Υ) and σ' with (ID_2, M, Υ) and analyze two tuples $\pi = (\pi_1, \pi_2), \pi' = (\pi'_1, \pi'_2)$.
If $\sigma_1 = \sigma'_1, \sigma_2 = \sigma'_2$ and $\sigma_3 = \sigma'_3$, we show that π and π' do not reveal the identity either.

$$\begin{aligned} g^{s_{ID_1}} h^{t_1} &= g^{s_{ID_2}} h^{t'_1} \\ g^{\frac{s_T}{z+s_{ID_1}}} h^{t_2} &= g^{\frac{s_T}{z+s_{ID_2}}} h^{t'_2} \\ u^{\frac{1}{s_{ID_1}+\mathcal{H}(M)}} h^{t_3} &= u^{\frac{1}{s_{ID_2}+\mathcal{H}(M)}} h^{t'_3} \end{aligned}$$

Suppose $h = g^{\eta}, h = u^{\xi}, \varepsilon = \frac{z+s_{ID_1}}{z+s_{ID_2}}, \tau = \frac{s_{ID_1}+\mathcal{H}(M)}{s_{ID_2}+\mathcal{H}(M)}$, we obtain that

$$\begin{aligned} t'_1 &= t_1 + \frac{s_{ID_1} - s_{ID_2}}{\eta} \\ t'_2 &= t_2 + \frac{s_T}{\eta}\left(\frac{1}{z+s_{ID_1}} - \frac{1}{z+s_{ID_2}}\right) = t_2 + \frac{s_T(1-\varepsilon)}{\eta(z+s_{ID_1})} \\ t'_3 &= t_3 + \frac{1}{\xi}\left(\frac{1}{s_{ID_1}+\mathcal{H}(M)} - \frac{1}{s_{ID_2}+\mathcal{H}(M)}\right) = t_3 + \frac{1-\tau}{\xi(s_{ID_1}+\mathcal{H}(M))} \end{aligned}$$

Now, we need to show that π_1, π_2 do not reveal any information about the user's identity. From the adversary's point of view, we see that $\pi_1, \pi_2, \pi'_1, \pi'_2$ satisfy,

$$\begin{aligned} \pi'_1 &= g^{\frac{s_T t'_1}{z+s_{ID_2}}} g^{(z+s_{ID_2})t'_2} h^{t'_1 t'_2} \\ \log_g \pi'_1 &= \frac{s_T t_1 + s_T \frac{s_{ID_1} - s_{ID_2}}{\eta}}{z+s_{ID_2}} + (z+s_{ID_2})(t_2 + \frac{s_T(1-\varepsilon)}{\eta(z+s_{ID_1})}) + \\ &\quad \eta(t_1 + \frac{s_{ID_1} - s_{ID_2}}{\eta})(t_2 + \frac{s_T(1-\varepsilon)}{\eta(z+s_{ID_1})}) \\ &= \frac{s_T t_1}{z+s_{ID_2}} + \frac{s_T(s_{ID_1} - s_{ID_2})}{\eta(z+s_{ID_2})} + zt_2 + s_{ID_2}t_2 + \\ &\quad \frac{s_T(1-\varepsilon)(z+s_{ID_2})}{\eta(z+s_{ID_1})} + \eta t_1 t_2 + s_{ID_1}t_2 - s_{ID_2}t_2 + \frac{s_T t_1(1-\varepsilon)}{z+s_{ID_1}} + \\ &\quad \frac{s_T(1-\varepsilon)(s_{ID_1} - s_{ID_2})}{\eta(z+s_{ID_1})} \\ &= \frac{s_T t_1}{z+s_{ID_1}} + (z+s_{ID_1})t_2 + \eta t_1 t_2 \\ \pi'_1 &= g^{\frac{s_T t_1}{z+s_{ID_1}} + (z+s_{ID_1})t_2 + \eta t_1 t_2} \\ &= g^{\frac{s_T t_1}{z+s_{ID_1}}} g^{(z+s_{ID_1})t_2} h^{t_1 t_2} \end{aligned}$$

$$= \pi_1$$

$$\begin{aligned}
\pi_2' &= u^{\frac{t_1'}{s_{ID_2}+\mathcal{H}(M)}} g^{(s_{ID_2}+\mathcal{H}(M))t_3'} h^{t_1' t_3'} \\
\log_g \pi_2' &= \frac{t_1 + \frac{s_{ID_1}-s_{ID_2}}{\eta}}{s_{ID_2}+\mathcal{H}(M)} \cdot \frac{\eta}{\xi} + (s_{ID_2}+\mathcal{H}(M))(t_3 + \frac{1-\tau}{\xi(s_{ID_1}+\mathcal{H}(M))}) + \\
&\quad \eta(t_1 + \frac{s_{ID_1}-s_{ID_2}}{\eta})(t_3 + \frac{1-\tau}{\xi(s_{ID_1}+\mathcal{H}(M))}) \\
&= \frac{t_1}{s_{ID_2}+\mathcal{H}(M)} \cdot \frac{\eta}{\xi} + \frac{s_{ID_1}-s_{ID_2}}{\xi(s_{ID_2}+\mathcal{H}(M))} + \mathcal{H}(M)t_3 + s_{ID_2}t_3 + \\
&\quad \frac{(1-\tau)(s_{ID_2}+\mathcal{H}(M))}{\xi(s_{ID_1}+\mathcal{H}(M))} + \eta t_1 t_3 + s_{ID_1}t_3 - s_{ID_2}t_3 + \\
&\quad \frac{t_1(1-\tau)}{s_{ID_1}+\mathcal{H}(M)} \cdot \frac{\eta}{\xi} + \frac{(1-\tau)(s_{ID_1}-s_{ID_2})}{\xi(s_{ID_1}+\mathcal{H}(M))} \\
&= \frac{t_1}{s_{ID_1}+\mathcal{H}(M)} \cdot \frac{\eta}{\xi} + (s_{ID_1}+\mathcal{H}(M))t_3 + \xi t_1 t_3 \\
\pi_2' &= g^{\frac{t_1}{s_{ID_1}+\mathcal{H}(M)} \cdot \frac{\eta}{\xi} + (s_{ID_1}+\mathcal{H}(M))t_3 + \eta t_1 t_3} \\
&= u^{\frac{t_1}{s_{ID_1}\mathcal{H}(M)}} g^{(s_{ID_1}+\mathcal{H}(M))t_3} h^{t_1 t_3} \\
&= \pi_2
\end{aligned}$$

Therefore, (π_1, π_2) is identical to (π_1', π_2'). The challenge signature σ does not reveal the identity id, though the simulator uses s_{id} to generate it. Hence the advantage of any adversary in the anonymity game G_1 is zero.

Thus the Construction - 2 preserves full anonymity.

5.4.2 Comparison

In the Construction - 2, the group signature contains 5 elements from $\mathbb{G}_n$. In Table 5.2 we compare our constructions with the existing ABGS schemes [72], [51] and [70]+[79]. Let $\Phi = |\zeta|$, where ζ be the set of attributes associated with a signature and $m = |Att|$. Let $m' \leq m$ be the number of attributes assigned to any user and r be the number of revoked members. Let RO denotes the Random oracle model, SGD denotes the Subgroup Decision assumption, (ℓ, m, t)-aMSE-CDH denotes the (ℓ, m, t)- augmented multi-sequence of exponents computational Diffie-Hellman and

let e represents the bilinear operation. We note that the verification cost of the proposed scheme is constant, where as other schemes verification cost is linear in terms of the number of attributes. The signature length of Construction - 2 is shorter than the Construction - 1 by one group element.

Table 5.2: Comparison of short ABGS scheme without non-frameability in standard model with other schemes

	Khader [72]	Emura et al. [51]	Herranz et al. [70]+ Liang et al. [79]	Our Scheme
User anonymity	CPA	CCA2	CPA	CPA
Attribute anonymity	no	no	yes	yes
Non-frameability	no	yes	no	no
Signature length	$O(\Phi)$	$O(\Phi)$	$15\|\mathbb{G}_p\| + 5\|\mathbb{G}_n\| = O(1)$	$5\|\mathbb{G}_n\| = O(1)$
User's Signing Key Length	$(m'+1)\|\mathbb{G}_p\| + \|\mathbb{Z}_p^*\|$	$(m'+1)\|\mathbb{G}_p\| + 2\|\mathbb{Z}_p^*\|$	$(m+m')\|\mathbb{G}_p\| + \|\mathbb{G}_n\| + \|\mathbb{Z}_p\|$	$(m'+1)\|\mathbb{G}_n\| + \|\mathbb{Z}_n^*\|$
Assumptions	DLDH, ℓ-SDH	DDH, ℓ-SDH, DL	$\begin{pmatrix}\text{DLin}_{,(\ell,m,t)}-\text{aMSE}-\text{CDH}\\ \text{SGD, } \ell'-\text{MOMSDH}\end{pmatrix}$	$\begin{pmatrix}\text{SGD, } \ell-\text{SDH, } \ell'-\text{MOMSDH,}\\ \text{DL, KEA1}\end{pmatrix}$
Model	RO	RO	Standard	Standard
Signing	$\begin{pmatrix}(7+2\Phi)\mathbb{G}_p+(5+\Phi)\mathbb{G}_T\\ +(\Phi+1)e\end{pmatrix}$	$\begin{pmatrix}(9+3\Phi)\mathbb{G}_p+(1+\Phi)\mathbb{G}_p\\ +8\mathbb{G}_T+3e\end{pmatrix}$	$\begin{pmatrix}(6m+6m'+\Phi(\Phi-1)+14)\mathbb{G}_p\\ +(m'+22)\mathbb{G}_n\end{pmatrix}$	$(22+2\Phi)\mathbb{G}_n$
Verification	$\begin{pmatrix}(6+2r)\mathbb{G}_p+(8+2\Phi)\mathbb{G}_T\\ +(\Phi+2r+1)e\end{pmatrix}$	$\begin{pmatrix}(11+2\Phi)\mathbb{G}_p+(\Phi+1)\mathbb{G}_2\\ +14\mathbb{G}_3+6e\end{pmatrix}$	$\begin{pmatrix}((4m+6m')\mathbb{G}_p+33e+21\mathbb{G}_T)\\ +((3)\mathbb{G}_n+6e+2\mathbb{G}_T)\end{pmatrix}$	$3\mathbb{G}+2\mathbb{G}_T+6e$

5.5 Summary

We have proposed two ABGS schemes having attribute anonymity with the constant size signature, and proven that they are secure under the standard model. Also the schemes achieve the constant computational cost at the verifier side. We observed that our schemes in the standard model are better than the existing ABGS schemes in terms of efficiency along with additional features viz. attribute anonymity and constant size signature.

Chapter 6

An ABGS Scheme with Attribute Anonymity and Attribute Tracing in the Standard Model

In the last chapter, we presented an ABGS scheme with attribute anonymity in the standard model which does not preserves non-frameability. In this chapter, we present an ABGS scheme not only having attribute anonymity feature but also with *attribute tracing* feature in the standard model. Moreover the scheme preserves non-frameability, i.e., even colluding group manager cannot forge the signature.

6.1 Introduction

The ABGS schemes proposed by Khader [72; 73] and by Emura et al. [51] do not have attribute anonymity and are secure under non-standard model. Moreover the Khader schemes are not non-frameable. We address attribute anonymity issue in the standard model which preserve non-frameability. We add a new feature called *attribute tracing* feature, which allows a user to know with what privilege (an attribute set) the signer has signed the document regardless of who did it. Notice that to build an ABGS

scheme with attribute anonymity in the standard model one can also combine an ABS scheme [70] with a group signature scheme [34], but it incurs combined cost of both the schemes.

In Section 6.2, a model of the proposed scheme and security definitions are given. The construction of the proposed ABGS scheme is described in Section 6.3. Its security analysis is given in Section 6.4. The comparison with the previous schemes is given in the Section 6.5. Finally we summarize in Section 6.6.

6.2 Proposed Scheme

In this section, we present the model and security definitions of ABGS scheme which is similar to the one given in [20; 51; 72] but with the added attribute anonymity and tracing features. Let k be the security parameter, *params* the system parameters, *Att* the universal set of attributes, Υ used to denote a predicate, $\Upsilon(\zeta) = 1$ denotes that the attribute set $\zeta \subseteq Att$ satisfies the predicate Υ, *gpk* the group public key, *ik* the issuing key used for issuing private keys to the users, ok_{user} the user opening key used to open the user identity of the group signature, tk_{att} the attribute tracing key used to trace the attributes of the group signature, $\mathcal{A}_i \subseteq Att$ the set of attributes assigned to the user U_i, sk_i denotes the private key for the member U_i and $\vec{reg}$ be the registration table with the group manager where the current group members information are stored.

A user U_i can make a group signature on a document M with the predicate Υ if there exists a set of attributes $\zeta \subseteq \mathcal{A}_i$ with the user such that $\Upsilon(\zeta) = 1$.

Definition 6.2.1 (ABGS) *An ABGS scheme consists of following algorithms. Unless otherwise indicated, algorithms are randomized.*

- $params \leftarrow \mathtt{Setup}(1^k)$: *This algorithm takes the security parameter k as an input and returns the system parameter params.*

- $(gpk, ik, ok_{user}, tk_{att}) \leftarrow \texttt{KeyGen}(params)$: *This algorithm takes the system parameter params, and returns a group public key gpk, an issuing key ik, a user opening key* ok_{user} *and an attribute tracing key* tk_{att}.

- $sk_i \leftarrow \texttt{Join}(\langle params, gpk, ik, upk_i, \mathcal{A}_i\rangle, \langle params, gpk, upk_i, usk_i\rangle)$: *This is an interactive group joining protocol between a user* U_i *(using his secret key* usk_i*) and the* ***GM*** *(using the issuing key ik and the attributes* $\mathcal{A}_i \subseteq Att$ *for* U_i*). In the protocol* U_i *ends with a member private key* sk_i *and* ***GM*** *ends with an updated registration table* $\vec{reg}$.

- $\sigma \leftarrow \texttt{Sign}(params, gpk, sk_i, \zeta, M, \Upsilon)$: *This algorithm takes* $params, gpk, sk_i$, *an attribute set* $\zeta \subseteq \mathcal{A}_i$, *message* M, *and the predicate* Υ *as an input and returns a group signature* σ *on* M.

- $0/1 \leftarrow \texttt{Verify}(params, gpk, M, \Upsilon, \sigma)$: *This is a deterministic algorithm verifies the validity of the group signature* σ *against gpk and returns* $1/0$. *If* 1 *then the algorithm claims that the* σ *is a valid group signature, otherwise,* σ *is invalid.*

- $i/\perp \leftarrow \texttt{OpenUser}(params, gpk, ok_{user}, \sigma, M, \Upsilon, \vec{reg})$: *This is a deterministic algorithm which takes as input* $params, gpk, ok_{user}, \sigma, \Upsilon, M$ *and* $\vec{reg}$, *and returns either* $i \geq 1$ *or* $\perp$. *If* i, *the algorithm claims that the group member with identity* i *has produced* σ, *and if* $\perp$, *then no group member produced* σ.

- $\zeta/\perp \leftarrow \texttt{TraceAtt}(params, gpk, tk_{att}, \sigma, M, \Upsilon)$: *This is a deterministic algorithm which takes as input* $params, gpk, tk_{att}, \sigma, M$ *and* Υ, *and outputs either the attribute set* $\zeta \subseteq Att$ *or* $\perp$. *Here it claims that* ζ *is the attribute set that is used to satisfy* Υ *in producing* σ. *If* $\perp$, *then the algorithm claims that no attribute set is used to produce* σ.

Entities: Following are the entities in ABGS scheme:

- The group manager GM, also known as *issuer*, has issuing key *ik* using which he enrolls a user into the group by allotting some privileges (in terms of attributes)

say $\mathcal{A}_i \subseteq Att$ and issuing a user's private key sk_i, by running interactive `Join` algorithm with the user.

- The *opener* has user opening key ok_{user} by which he is able to open the signature and reveal the user identity through `OpenUser` algorithm.
- The *attribute tracer* has the attribute tracing key tk_{att} by which he can trace the attribute set ζ from the group signature, which is used to satisfy the predicate Υ, by running the `TraceAtt` algorithm.
- Group members or signers who are having their private keys sk_i. They run `Sign` algorithm to produce a group signature on a document M with predicate Υ if they possess valid attribute set $\mathcal{A}_i$ which satisfies the predicate.
- Outsider or verifier who can seek a group signature for a document M with predicate Υ from group manager `GM`. He can also verify the group signature using the group public key, gpk.

Note Normally the `Setup` and `KeyGen` algorithms are run by some trusted party and he will distribute the appropriate keys to the concerned entities.

ABGS scheme is correct if the group signatures produced by an honest group member are verified, and reveals the identity of the signer and the attribute set used.

Definition 6.2.2 (`Correctness`) Correctness *requires that for all* $params \leftarrow \texttt{Setup}(1^k)$, *all* $(gpk, ik, ok_{user}, tk_{att}) \leftarrow \texttt{KeyGen}(params)$, $sk_i \leftarrow \texttt{Join}(\langle params, gpk, ik, upk_i, \mathcal{A}_i\rangle, \langle params, gpk, upk_i, usk_i\rangle)$, *all* Υ, *all* $\zeta \subseteq Att$ *and all* $M \in \{0,1\}^*$, *if* $\mathrm{U}_i \in r\vec{e}g$, $\zeta \subseteq \mathcal{A}_i, \Upsilon(\zeta) = 1$ *and* $\sigma = \texttt{Sign}(params, gpk, sk_i, \zeta, M, \Upsilon)$ *then*

$$
\begin{aligned}
1 &\leftarrow \texttt{Verify}(params, gpk, M, \Upsilon, \sigma) \\
\bigwedge i &\leftarrow \texttt{OpenUser}(params, gpk, ok_{user}, \sigma, M, \Upsilon, r\vec{e}g) \\
\bigwedge \zeta &\leftarrow \texttt{TraceAtt}(params, gpk, tk_{att}, \sigma, M, \Upsilon)
\end{aligned}
$$

holds.

In the following definitions the adversary can run the Join protocol (similar to [24]):

- either through the joinP-oracle (passive join), which means that it creates an honest user for whom it does not know the private keys: the index i is added to the HU (Honest Users) list;
- or through the joinA-oracle (active join), which means that it interacts with the group manager to create a user whom it will control: the index i is added to the CU (Corrupted Users) list.

Note that when the adversary is given the issuing key (the group manager is corrupted) then the adversary does not need access to the joinA oracle since it can simulate it by itself, to create corrupted users (that are not necessarily in CU). After a user is created, the adversary plays the role of corrupted users, and can interact with honest users, granted some oracles:

- corrupt(i), if $i \in$ HU, provides the specific private key of this user. The adversary can now control it during the whole simulation. Therefore i is moved from HU to CU;
- sign(i, M, Υ), if $i \in$ HU, plays as the honest user i would do in the signature process to generate a signature on message M with predicate Υ;
- openusr(M, σ, Υ), if (M, Υ, σ) is valid, returns the identity i of the signer;
- tratt(M, σ, Υ), if (M, Υ, σ) is valid, returns the attribute set ζ which used to satisfy Υ in producing σ.

In ABGS scheme a group member may have multiple attribute sets to satisfy the predicate and he can produce a group signature using one of them. An ABGS scheme preserves attribute anonymity if it is computationally difficult to identify with what attribute set he produces the signature.

Definition 6.2.3 (`Attribute Anonymity`) *We say that the ABGS scheme preserves* attribute anonymity *if for all PPT $\mathcal{A}$, the probability that $\mathcal{A}$ wins the following game is negligible.*

- ***Setup:*** *The challenger $\mathcal{C}$ runs $(gpk, ik, ok_{user}, tk_{att}) \leftarrow \mathtt{KeyGen}(params)$. $\mathcal{C}$ gives gpk, ik, ok_{user} to $\mathcal{A}$.*
- `Phase1` : *$\mathcal{A}$ is given access to the oracles:* $\mathsf{joinP}, \mathsf{corrupt}, \mathsf{sign}$ *and* tratt.
- `Challenge` : *$\mathcal{A}$ outputs M^*, Υ^*, and an uncorrupted users U_i (i.e. $i \notin \mathsf{CU}$) such that $\exists \zeta_{i_0}, \zeta_{i_1} \subseteq \mathcal{A}_i$ and $\Upsilon(\zeta_{i_0}) = 1, \Upsilon(\zeta_{i_1}) = 1$ holds. $\mathcal{C}$ randomly selects $\kappa \in_R \{0, 1\}$ and responds with a group signature $\sigma^* \leftarrow \mathtt{Sign}(params, gpk, sk_i, \zeta_{i_\kappa}, M, \Upsilon)$.*
- `Phase 2` : *$\mathcal{A}$ can make queries similar to Phase 1. However $\mathcal{A}$ cannot make query to* $\mathsf{corrupt}$ *on i.*

Output: *Finally, $\mathcal{A}$ outputs a bit κ', and wins if $\kappa' = \kappa$.*

The advantage of $\mathcal{A}$ is defined as $Adv^{att-anon}(\mathcal{A}) = |Pr(\kappa = \kappa') - \frac{1}{2}|$.

Thus there should not exists any PPT adversary to link a group signature to a set of attributes used to generate it.

ABGS scheme preserves user anonymity if there are at least two group members possessing valid attribute sets and one of them produces the group signature then it should be computationally hard to identify who produced the group signature among them.

Definition 6.2.4 (`User Anonymity`) *We say that the ABGS scheme preserves* user anonymity *if for all PPT $\mathcal{A}$, the probability that $\mathcal{A}$ wins the following game is negligible.*

- ***Setup:*** *The challenger $\mathcal{C}$ runs $(gpk, ik, ok_{user}, tk_{att}) \leftarrow \mathtt{KeyGen}(params)$. $\mathcal{C}$ gives gpk, ik, tk_{att} to $\mathcal{A}$.*

- `Phase1` : $\mathcal{A}$ *is given access to the oracles:* joinP, corrupt, sign *and* openusr.
- `Challenge` : $\mathcal{A}$ *outputs* M^*, Υ^*, *and an uncorrupted users* $\mathrm{U}_{i_0}, \mathrm{U}_{i_1}$ *(i.e.* $i_0, i_1 \notin$ CU*) and,* $\zeta : \zeta \subseteq \mathcal{A}_{i_0}, \zeta \subseteq \mathcal{A}_{i_1}$ *and* $\Upsilon(\zeta^1) = 1$. $\mathcal{C}$ *randomly selects* $\kappa \in_R \{0, 1\}$ *and responds with a group signature* $\sigma^* \leftarrow \mathtt{Sign}(params, gpk, sk_{i_\kappa}, \zeta, M, \Upsilon)$.
- `Phase 2` : $\mathcal{A}$ *can make queries similar to Phase* 1. *However* $\mathcal{A}$ *cannot make query to* corrupt *on* i_0 *and* i_1 *at any time.*

Output: *Finally,* $\mathcal{A}$ *outputs a bit* κ', *and wins if* $\kappa' = \kappa$.

The advantage of $\mathcal{A}$ *is defined as* $Adv^{user-anon}(\mathcal{A}) = |Pr(\kappa = \kappa') - \frac{1}{2}|$.

Thus there should not exist any PPT adversary to link a group signature to a signer with non-negligible probability.

ABGS scheme preserves traceability if it is possible to trace the valid group signature to its signer with the help of group opening key.

Definition 6.2.5 (`Traceability`) *We say that the ABGS scheme preserve* traceability *if for all PPT* $\mathcal{A}$, *the probability that* $\mathcal{A}$ *wins the following game is negligible.*

- ***Setup:*** *The challenger* $\mathcal{C}$ *runs* $(gpk, ik, ok_{user}, tk_{att}) \leftarrow \mathtt{KeyGen}(params)$. $\mathcal{C}$ *gives* gpk, ok_{user} and tk_{att} *to* $\mathcal{A}$.
- ***Queries:*** $\mathcal{A}$ *is given access to the oracles:* joinP, joinA, corrupt *and* sign.
- ***Output:*** $\mathcal{A}$ *outputs a message* M^*, *a predicate* Υ^* *and a group signature* σ^*.

$\mathcal{A}$ *wins if*
(1) $\mathtt{Verify}(params, gpk, M^*, \Upsilon^*, \sigma^*) = 1$ *and*
(2) $\mathtt{OpenUser}(params, gpk, ok_{user}, \sigma^*, M^*, \Upsilon^*, \vec{reg}) = \perp$.
The advantage of $\mathcal{A}$ *is defined as the probability that* $\mathcal{A}$ *wins.*

[1]Here ζ can be different for $\mathrm{U}_{i_0}, \mathrm{U}_{i_1}$ but we are concerned about user anonymity rather than attribute anonymity

Thus it should be impossible to produce an untraceable valid group signature by any PPT adversary.

ABGS scheme preserves non-frameability if it is difficult to produce a valid group signature which trace back to a group member who does not produce it, even with the help of group manager's secret key.

Definition 6.2.6 (`Non-frameability`) *We say that the ABGS scheme preserves* non-frameability *if for all PPT $\mathcal{A}$, the probability that $\mathcal{A}$ wins the following game is negligible.*

- ***Setup:*** *The challenger $\mathcal{C}$ runs $(gpk, ik, ok_{user}, tk_{att}) \leftarrow \mathtt{KeyGen}(params)$. $\mathcal{C}$ gives gpk, ik, ok_{user}* and *tk_{att} to $\mathcal{A}$.*
- ***Queries:*** *$\mathcal{A}$ is given access to the oracles:* joinP, corrupt *and* sign.
- ***Output:*** *Finally, $\mathcal{A}$ outputs a message M^*, a predicate Υ^* and a group signature σ^*.*

$\mathcal{A}$ wins if
(1) $\mathtt{Verify}(params, gpk, M^*, \Upsilon^*, \sigma^*) = 1$,
(2) $\mathtt{OpenUser}(params, gpk, ok_{user}, \sigma^*, M^*, \Upsilon^*, r\vec{e}g) = i^*$,
(3) $i \in \mathsf{HU}$.
The advantage of $\mathcal{A}$ is defined as the probability that $\mathcal{A}$ wins.

Thus even the group manager should not be able to forge a group signature which trace back to a honest member.

ABGS scheme preserves attribute unforgeability if it is hard for a group member to forge an attribute certificate in order to produce a valid group signature.

Definition 6.2.7 (`Attribute Unforgeability`) *We say that the ABGS scheme preserves* attribute unforgeability *if for all PPT $\mathcal{A}$, the probability that $\mathcal{A}$ wins the following game is negligible.*

- ***Setup:*** *The challenger $\mathcal{C}$ runs $(gpk, ik, ok_{user}, tk_{att}) \leftarrow \mathtt{KeyGen}(params)$. $\mathcal{C}$ gives gpk, ok_{user} and tk_{att} to $\mathcal{A}$.*
- ***Queries:*** *$\mathcal{A}$ is given access to the oracles:* $\mathsf{joinP}, \mathsf{joinA}, \mathsf{corrupt}$ *and* sign.
- ***Output:*** *$\mathcal{A}$ outputs a message M^*, a predicate Υ^* and a group signature σ^*.*

$\mathcal{A}$ wins if
(1) $\mathtt{Verify}(params, gpk, M^*, \Upsilon^*, \sigma^*) = 1$,
(2) $\mathtt{OpenUser}(params, gpk, ok_{user}, \sigma^*, M^*, \Upsilon^*, r\vec{e}g) = i^*$ *and*
(3) $\nexists \zeta \in \mathcal{A}_{i^*} : \Upsilon(\zeta) = 1$.
The advantage of $\mathcal{A}$ is defined as the probability that $\mathcal{A}$ wins.

Thus it should be impossible for any PPT adversary to satisfy the predicate with invalid set of attributes.

ABGS scheme preserves collusion resistance of attribute certificates if it is computationally hard for group members to collude by pooling their attribute certificates to satisfy the predicate and to produce a valid group signature.

Definition 6.2.8 (`Collusion resistance of Attributes`) *We say that the ABGS scheme preserves* collusion resistance of attributes *if for all PPT $\mathcal{A}$, the probability that $\mathcal{A}$ wins the following game is negligible.*

- ***Setup:*** *The challenger $\mathcal{C}$ runs $(gpk, ik, ok_{user}, tk_{att}) \leftarrow \mathtt{KeyGen}(params)$. $\mathcal{C}$ gives gpk, ok_{user} and tk_{att} to $\mathcal{A}$.*
- ***Queries:*** *$\mathcal{A}$ is given access to the oracles:* $\mathsf{joinP}, \mathsf{joinA}, \mathsf{corrupt}$ *and* sign.
- ***Output:*** *$\mathcal{A}$ outputs a message M^*, a predicate Υ^* and a group signature σ^*.*

$\mathcal{A}$ wins if
(1) $\mathtt{Verify}(params, gpk, M^*, \Upsilon^*, \sigma^*) = 1$, *and*
(2) $\mathcal{A}$ has obtained $sk_{i_1}, ..., sk_{i_k} : \Upsilon^*(\cup_{j=1}^{k} \mathcal{A}_{i_j}) = 1$ and $\Upsilon^*(\mathcal{A}_{i_j}) \neq 1$ for $j = 1, ..., k$.
The advantage of $\mathcal{A}$ is defined as the probability that $\mathcal{A}$ wins.

Thus the users with valid set of attributes each, cannot collude with each other to pool a valid attribute set for producing a valid group signature.

6.3 Construction

A construction of an ABGS scheme with attribute anonymity and attribute tracing features is presented in this section. For our construction we use the membership certificate format of [24; 48] to achieve non-frameability and the technique to build the access trees from [51]. We use Groth-Sahai non-interactive proof system under SXDH assumption (see Sec. 2.4.7.1) to generate the NIWI proofs for the relations in the group signature. We use existing constructions [24; 51] as a base to build our scheme which addresses the said issues and we prove that the construction is secure under standard model. In contrast to other existing ABGS schemes [51; 72; 73], our scheme is built in the standard model with attribute anonymity and achieves a constant size signature, independent of the number of attributes.

Let T_Υ be an access tree representing the predicate Υ, $\mathcal{T}_\Upsilon$ the public values associated with T_Υ, (upk_i, usk_i) the verification/signing key of a signature scheme $DSig$ for user U_i, A_i the membership certificate for U_i, $\{T_{i,j}\}_{\mathrm{att}_j \in \mathcal{A}_i}$ denotes the attribute certificates of U_i and the $T_{i,j}$ is the attribute certificate of the attribute $att_j \in Att$ of user U_i.

For a polynomial number of scalars $z_i \in \mathbb{Z}_p^*$, and a pair $(g, g^y) \in \mathbb{G}_1^2$, the values $g^{1/(y+z_i)}$ looks to be random and independent [50]. This is used to build our identifier, $\mathrm{ID}(y, z_i) = g^{1/(y+z_i)}$, in the group signature. In the proof of user anonymity, the simulator will be able to choose a z_i prior to any interaction with the adversary and we depend on q-DDHI assumption [50].

- `Setup`(1^k): It generates the system parameters, $params = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, Att)$; where

  (i) $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are the cyclic groups of prime order p, where $2^{k-1} < p < 2^k$.

  (ii) $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear map.

(iii) g_1 and g_2 are the generators of the groups $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively.

(iv) $Att = \{\text{att}_1, ..., \text{att}_m\}$, for $m = O(k)$ is the universal set of attributes[1].

- `KeyGen`($params$): It takes an input system parameters $params$ and outputs a group public key gpk, an issuing key ik, a user opening key ok_{user} and an attribute tracing key tk_{att}.

  (i) Select the generators $h, v', v_1, ..., v_{m'} \in \mathbb{G}_1$ and define the Waters function, $\mathcal{F} : \{0,1\}^{m'} \rightarrow \mathbb{G}_1$, for $M = (\mu_1, ..., \mu_{m'}) \in \{0,1\}^{m'} \mathcal{F}(M) = v' \Pi_{j=1}^{m'} v_j^{\mu_j}$, where $m' = O(k)$.

  (ii) Select $\gamma \in_R \mathbb{Z}_p^*$, and computes $\omega = g_2^{\gamma}$.

  (iii) For each $\text{att}_j \in Att$, choose a secret $s_j \in_R \mathbb{Z}_p^*$, sets $S = \{s_j\}_{\text{att}_j \in Att}$, and computes $g_{\text{att}_j} = g_1^{s_j}, \forall \text{att}_j \in Att$.

  (iv) For the Groth-Sahai proof under the instantiation based on SXDH assumption, choose a vectors $\vec{u} = (\vec{u_1}, \vec{u_2} = \vec{u_1}^{t_1}), \vec{u'} = (\vec{u'_1}, \vec{u'_2} = \vec{u'_1}^{t_1})$ and $\vec{v} = (\vec{v_1}, \vec{v_2} = \vec{v_1}^{t_2})$, where $\vec{u}_1 = (g_1, g_1^{\alpha_1}) \in \mathbb{G}_1^2$, $\vec{u'}_1 = (g_1, g_1^{\alpha'_1}) \in \mathbb{G}_1^2$ and $\vec{v}_1 = (g_2, g_2^{\alpha_2}) \in \mathbb{G}_2^2$ for $t_1, t_2, \alpha_1, \alpha'_1, \alpha_2 \in_R \mathbb{Z}_p^*$. α_1 and α_2 are commitment keys.

  (v) Outputs

$$gpk = (h, \omega, \mathcal{F}, \{g_{\text{att}_j}\}_{\text{att}_j \in Att}, \vec{u}, \vec{u'}, \vec{v}), ik = (\gamma, S), ok_{user} = \alpha_1, tk_{att} = \alpha'_1.$$

  The description of $\mathcal{F}$ includes the generators $v', v_1, ..., v_{m'}$.
  Here $\vec{u'}$ is needed apart from $\vec{u}$ to make independent commitment in the signature algorithm. With this we make attribute tracing key independent of user opening key. Otherwise if both the authorities are same we can remove $\vec{u'}$ from the gpk and has common key for ok_{user} and tk_{att}.

- `Join`($< params, gpk, ik, upk_i, \mathcal{A}_i >, < params, gpk, upk_i, usk_i >$): A user U_i with the pair of keys (upk_i, usk_i) in the PKI, interacts with the group manager and the set of attributes $\mathcal{A}_i$, to join the group. This is similar to the `Join` protocol in [24; 48]. As a result of this protocol, U_i gets private key $sk_i =$

[1]For $m = 1$ it becomes a group signature scheme.

$((A_i, X_i, y_i), \{T_{i,j}\}_{\text{att}_j \in \mathcal{A}_i})$, where (A_i, X_i, y_i) is a *membership certificate*, $\{T_{i,j}\}_{\text{att}_j \in \mathcal{A}_i}$ is the set of *attribute certificates* and $\mathcal{A}_i \subseteq Att$ is the set of U_i's attributes. And GM ends with the updated $r\vec{e}g$. The protocol begins as follows,

(i) U_i picks $y_i' \in_R \mathbb{Z}_p^*$, computes and sends $Y_i' = g_1^{y_i'}$, an extractable commitment of y_i'. Note that the trapdoor of the commitment will not be known to anybody except to the simulator in the security proof to be able to extract y_i'.

(ii) GM selects new $x_i \in \mathbb{Z}_p^*$ and a random $y_i'' \in_R \mathbb{Z}_p^*$, computes $A_i = (hY_i'Y_i'')^{1/(\gamma+x_i)}$, $X_{i,2} = g_2^{x_i}$ and $T_{i,j} = h^{\frac{s_j}{\gamma+x_i}} (\forall \text{att}_j \in \mathcal{A}_i)$, where $Y_i'' = g_1^{y_i''}$ and sends $y_i'', A_i, X_{i,2}, \{T_{i,j}\}_{\forall \text{att}_j \in \mathcal{A}_i}$.

(iii) U_i checks whether $e(A_i, \omega g_2^{x_i}) = e(h, g_2)e(g_1, g_2)^{y_i'+y_i''}$. Then U_i computes $y_i = y_i' + y_i''$ and makes a signature $\sigma_i = DSig_{usk_i}(A_i, X_{i,2}, Y_i = g_1^{y_i})$.

(iv) GM verifies σ_i under upk_i and appends the tuple $(i, upk_i, A_i, X_i = (X_{i,1} = g_1^{x_i}, X_{i,2}), Y_i, s_i)$ to $r\vec{e}g$. Then GM sends $X_{i,1}$.

(v) U_i checks the relation $e(X_{i,1}, g_2) = e(g_1, X_{i,2})$. U_i owns an valid membership certificate (A_i, X_i, y_i) and attribute certificates $\{T_{i,j}\}_{\forall \text{att}_j \in \mathcal{A}_i}$, where y_i is known to him only. Thus, $sk_i = (A_i, X_i, y_i, \{T_{i,j}\}_{\text{att}_j \in \mathcal{A}_i}) \in \mathbb{G}_1^2 \times \mathbb{G}_2 \times \mathbb{Z}_p^* \times \mathbb{G}_1^{|\mathcal{A}_i|}$.

GM chooses $s_{m+1} \in \mathbb{Z}_p^*$, and computes $g_{\text{att}_{m+1}} = g_1^{s_{m+1}}$ when a new attribute att_{m+1} is added. Let U_i be issued $T_{i,m+1}$. Then GM computes and sends $T_{i,m+1} = h^{\frac{s_{m+1}}{\gamma+x_i}}$ to U_i and also publish $g_{\text{att}_{m+1}}$ into gpk.

- `BuildTree`(p, S, Υ):

(i) Let T_Υ be the tree that represents the predicate Υ.

(ii) Get extension tree $T^{\text{ext}} \leftarrow$ `AddDummyNode`(T_Υ).

(iii) Get secret value for each dummy node and the secret value of the root of T^{ext} using $(\{s_{d_j}\}_{d_j \in D_T}, s_T) \leftarrow$ `AssignedValue`(S, T^{ext}).

(iv) Output the public values of Υ,

$$\mathcal{T}_\Upsilon = (\{s_{d_j}\}_{d_j \in D_{T_\Upsilon}}, v_T = g_2^{s_T}, T^{ext}).$$

Normally the verifier with his predicate approaches the `GM` for a group signature and `GM` runs `BuildTree` algorithm to generate the public values of the predicate Υ and stores it in a public repository. Then anyone among the group members who are eligible will generate a group signature by using the predicate public value. And in order to verify whether the published values of the predicate are correct one (specially the verifier) can use the `BuildTree-Validity` algorithm. By this the verifiers (outsiders) need *not* trust the `GM` as far as the predicate public values are concerned.

- `BuildTree-Validity`$(params, gpk, \mathcal{T}_\Upsilon)$:

  (i) Randomly choose an attribute set, $Leaves \subseteq Att : \Upsilon(Leaves) = 1$ And gets the corresponding $\Delta_{\text{att}_j}(\forall \text{att}_j \in Leaves)$, and $\Delta_{d_j}(\forall d_j \in D_{T_\Upsilon}^{Leaves})$ by running `MakeSimplifiedTree`$(Leaves,$ $T^{ext})$.

  (ii) Compute $g_{root} = \prod_{\text{att}_j \in Leaves} g_{\text{att}_j}^{\Delta_{\text{att}_j}} \times \prod_{d_j \in D_{T_\Upsilon}^{Leaves}} g_1^{s_{d_j}\Delta_{d_j}}$
  $= \prod_{\text{att}_j \in Leaves} g_1^{s_j \Delta_{\text{att}_j}} \times \prod_{d_j \in D_{T_\Upsilon}^{Leaves}} g_2^{s_{d_j}\Delta_{d_j}}$
  $= g_1^{\sum_{\text{att}_j \in Leaves} \Delta_{\text{att}_j} s_j + \sum_{d_j \in D_{T_\Upsilon}^{Leaves}} \Delta_{d_j} s_{d_j}} = g_1^{s_T}$ from (2.6).

  (iii) Verify whether $e(g_{root}, g_2) \stackrel{?}{=} e(g_1, v_T)$. If not then $\mathcal{T}_\Upsilon$ is the invalid public values of the predicate Υ.

- `Sign`$(params, gpk, sk_i, \zeta, M, \Upsilon)$: It generates a group signature σ on message $M \in \{0,1\}^{m'}$ with the user private key sk_i who satisfy the predicate Υ with his subset of attributes $\zeta \subseteq \mathcal{A}_i : \Upsilon(\zeta) = 1$.

  (i) Get the public values of Υ, $\mathcal{T}_\Upsilon = (\{s_{d_j}\}_{d_j \in D_{T_\Upsilon}}, v_T, T^{ext})$, from the public repository.

  (ii) Let $s_{T_1} = \Sigma_{\text{att}_j \in \zeta} \Delta_{\text{att}_j} s_j$ and $s_{T_2} = \Sigma_{d_j \in D_T^\zeta} \Delta_{d_j} s_{d_j}$. Then from (2.6), $s_{T_1} + s_{T_2} = s_T$.

  (iii) Get $(\{\Delta_{\text{att}_j}\}_{(\forall \text{att}_j \in \zeta)}, \{\Delta_{d_j}\}_{(\forall d_j \in D_{T_\Upsilon}^\zeta)}) \leftarrow$ `MakeSimplifiedTree`(ζ, T^{ext}) and compute s_{T_2}.

  (iv) Creates an ephemeral ID, $\text{ID}(y_i, z) = g_1^{1/(z+y_i)}$, with a random $z \in \mathbb{Z}_p^*$.

(v) Select $r \in_R \mathbb{Z}_p^*$ and set $\rho_1 = A_i, \rho_2 = y_i, \rho_3 = X_i = (g_1^{x_i}, g_2^{x_i}), \rho_4 = \Pi_{\text{att}_j \in \zeta} T_{i,j}^{\Delta_{\text{att}_j}} = h^{\frac{s_{T_1}}{\gamma + x_i}}, \rho_5 = h^{s_{T_2}}, \rho_6 = \text{ID}(y_i, z), \rho_7 = (g_1^z, g_2^z), \rho_8 = h^z \mathcal{F}(M)^r$ and $\rho_9 = g_2^r$,.

(vi) Commit the group elements $\sigma_i = \mathcal{C}(\rho_i)$, for $i = \{1, 3, 4, 5\}$ and $\sigma_2 = (\mathcal{C}^{(1)}(\rho_2), \mathcal{C}^{(2)}(\rho_2))$. Note that for committing the ρ_4 and ρ_5 one has to use $\vec{u}'$. This is to separate the opening of user with the tracing of attributes.

(vii) Compute the NIWI Groth-Sahai proofs for the committed variables $\rho_1, \rho_2, \rho_3, \rho_4, \rho_5$ satisfy the following equations

$$
\begin{aligned}
e(\rho_1, \omega\rho_{3,2}) &= e(h, g_2) \times e(g_1, g_2^{\rho_2}) && (6.1)\\
e(\rho_4, \omega\rho_{3,2}) \times e(\rho_5, g_2) &= e(h, v_T) && (6.2)\\
e(\rho_6, g_2^{\rho_2}\rho_{7,2}) &= e(g_1, g_2) && (6.3)\\
e(\rho_8, g_2) &= e(h, \rho_{7,2}) \times e(\mathcal{F}(M), \rho_9) && (6.4)\\
e(g_1^{\rho_2}, g_2) &= e(g_1, g_2^{\rho_2}) && (6.5)\\
e(\rho_{3,1}, g_2) &= e(g_1, \rho_{3,2}) && (6.6)\\
e(\rho_{7,1}, g_2) &= e(g_1, \rho_{7,2}) && (6.7)
\end{aligned}
$$

(viii) Output the signature

$$\sigma = (\{\sigma_i\}_{i=1}^5, \{\rho\}_{i=6}^9) \in \mathbb{G}_1^{13} \times \mathbb{G}_2^6$$

We add the corresponding Groth-Sahai proofs to the signature to prove the validity of the above pairing equations. Equation (6.1) is a pairing product equation, establishes that the signer has a valid membership certificate issued through the `Join` algorithm (i.e. A_i is well-formed), and the Groth-Sahai proof requires 4 elements in each group. Equation (6.2) is a paring product equation, establishes that the signer possess the required attributes (attribute certificates) that satisfy the predicate Υ and also proves the association of the membership certificates with the attribute certificates, and the proof requires 4 elements in each group. Equation (6.3) is a linear pairing product, establishes that ρ_6 is a well formed ID, and the proof requires only 2 extra elements in $\mathbb{G}_1$.

Equation (6.4) does not use any committed data so it can be directly checked, it establishes that (ρ_8, ρ_9) is a Waters signature of M under the key ρ_7. Equation (6.5) is a quadratic equation, establishes that same y_i is committed in both groups which is needed for traceability adversary modeling, and it can prove with 2 elements in each group. Equation (6.6) is a pairing product equation, establishes that X_i is well-formed which is needed for traceability adversary modeling, and the proof requires 4 elements in each group. Equation (6.7) does not use any committed data so it can be checked directly and this equation is needed for non-frameability adversary modeling. Overall we will need 29 group elements in $\mathbb{G}_1$ and 20 in $\mathbb{G}_2$. Note that the signature is independent of the number of attributes $|\zeta|$.

- `Verify`$(params, gpk, M, \sigma, \Upsilon)$:It verifies to see whether all the paring equations hold according to Groth-Sahai proof system.

- `OpenUser`$(params, gpk, ok_{user}, \sigma, M, \Upsilon, r\vec{e}g)$: For the valid group signature the Opener just opens the commitment of A_i in σ_1, and outputs the corresponding identity i from the $r\vec{e}g$ with respect to A_i, if it is present, otherwise outputs $\perp$.

- `TraceAtt`$(params, gpk, tk_{att}, \sigma, M, \Upsilon)$: For the valid group signature the Attribute Tracer opens the commitment of $\rho_5 = h^{s_{T_2}}$ from σ_5. Then for all $\zeta_k : \Upsilon(\zeta_k) = 1$, it checks $\rho_5 \stackrel{?}{=} h^{s^k_{T_2}}$, where $s^k_{T_2} = \Sigma_{d_j \in D^{\zeta_k}_{T_\Upsilon}} \Delta_{d_j} s_{d_j}$. If any such ζ_k exists then outputs it else outputs $\perp$. We note that for each unique ζ there is unique s_{T_2} value, it is from Lagrange interpolation.

6.4 Security Analysis

Theorem 6.4.1 *The proposed ABGS scheme is correct.*

Proof The correctness follows from the Groth-Sahai proof system. □

Theorem 6.4.2 *The proposed ABGS scheme preserves attribute anonymity under SXDH assumption.*

Proof The proof follows from the Groth-Sahai proof system. Namely the attribute details are hidden in the components ρ_4 and ρ_5 which are committed with Groth-Sahai proof technique to σ_4 and σ_5, therefore under SXDH assumption it is perfectly hiding. □

Theorem 6.4.3 *If there exists an adversary $\mathcal{A}$ that can break the user anonymity of the scheme, then there exists an adversary $\mathcal{B}$ that can break the ℓ-DDHI problem in $\mathbb{G}_1$ or the SXDH assumption, where ℓ is the maximal number of signing queries for a user. And we have*

$$Adv^{\text{user}-\text{anon}} \leq 1/n.(2.Adv^{\text{SXDH}} + Adv^{\ell-\text{DDHI}}) \tag{6.8}$$

where n is the maximal number of join queries and ℓ is the maximal number of signing queries for a user.

Proof The proof follows the approach of anonymity adversary in [24]. The proof is organized as a sequence of games such that adversary has no advantage in final game where as the first game is the real attack game as given in definition (6.2.4). Let S_i denote the event that the adversary wins in the game G_i with advantage $Adv_i = |Pr[S_i] - 1/2|$.

G_1: This is the real game as define in the definition (6.2.4). Challenger $\mathcal{B}$ sets up the scheme and defines the parameters as in the real scheme,

$$\begin{aligned} params &= (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, Att) \\ gpk &= (h, \omega, \mathcal{F}, \{g_{\text{att}_j}\}_{\text{att}_j \in Att}, \vec{u}, \vec{u}', \vec{v}) \\ ik &= (\gamma, S) \\ ok_{user} &= \alpha_1 \\ tk_{att} &= \alpha_1' \end{aligned}$$

$\mathcal{B}$ gives gpk, ik and tr_{att} to $\mathcal{A}$. With this $\mathcal{B}$ answers all the queries made by adversary. At challenge phase $\mathcal{A}$ chooses 2 unrevoked and uncorrupted users U_{i_0}, U_{i_1} and is given

a challenge signature σ^* on behalf of U_{i_κ}, $\kappa \in_R \{0,1\}$. In the output phase, adversary outputs her guess $\kappa' \in \{0,1\}$ and the advantage is $Adv_1 = |Pr[S_1] - 1/2|$.

G_2: Let n be the total number of passive join queries, joinP queries. In this game we modify the simulation G_1, $\mathcal{B}$ picks a challenge user id $i^\star$. In the challenge phase, $\mathcal{B}$ aborts if $i_\kappa \neq i^\star$. $\mathcal{B}$ also aborts if $i^\star$ is queried to corrupt or revoke oracle before or during challenge phase period. The probability that $\mathcal{B}$ succeed in picking correct $i^\star$ is $1/n$. Therefore, $Adv_2 = Adv_1/n$.

G_3: We modify the simulation of G_2. $\mathcal{B}$ chooses $y \in_R \mathbb{Z}_p^*$ and define the $\ell-$DDHI like tuple $A = (g, g^y, ..., g^{y^\ell}) \in \mathbb{G}_1^{\ell+1}, D = g^{1/y} \in \mathbb{G}_1$. $\mathcal{B}$ chooses different random values $z^*, z_1, ..., z_{\ell-1} \in \mathbb{Z}_p^\ell$, and define the polynomial $f(X) = \Pi_{i=1}^{\ell-1}(X - z_i)$, of degree $\ell - 1$. From the above tuple, $\mathcal{B}$ can compute $g_1 = g^{f(X)}$. The future challenge user $i^\star$ will virtually have $y_{i^\star} = y - z^*$ and $x_{i^\star} \in_R \mathbb{Z}_p^*$. $\mathcal{B}$ compute $g^{y_{i^\star}} = g_1^y / g_1^{z^*}$ from the above tuple. The membership certificate for the challenge user is $(g_1^{x_{i^\star}}, g_2^{x_{i^\star}}, g_1^y/g_1^{z^*}, A_{i^\star} = (hg_1^y/g_1^{z^*})^{\beta/(\gamma+x_{i^\star})}, \{T_{i^\star,j} = h^{\frac{s_j}{\gamma+x_{i^\star}}}\}_{att_j \in \mathcal{A}_{i^\star}})$. The setup is indistinguishable from G_2, since all keys are having same distribution.
$\mathcal{B}$ answers all queries according to definition (6.2.4) and for the challenge user $U_{i^\star}$, the j-th signing queries, he computes $\rho_6 = g_1^{1/(y_{i^\star}+z_j)} = g^{\Pi_{i\neq j}(y+z_i)}$, that can be done from the defined tuple, the rest is done as in the real scheme using z_j as random. $\mathcal{B}$ can also answer any corruption query, that should not happen for the challenge user, even if we know y in this game.
For the challenge signing query, he does the same as above with the ephemeral value z^*, and the expected ID, $\rho_6 = g_1^{1/(y_{i^\star}+z^*)} = g^{f(y)/y} = g^{f'(y)} g^{\Pi(z_i)/y}$, where $f'(X) = (\Pi_{i=1}^{\ell}(X + z_i) - \Pi(z_i))/X$ is a polynomial of degree $\ell - 1$ and thus $g^{f'(y)}$ can be computable from the tuple. $\mathcal{B}$ thus compute $\rho_6 = g^{f'(y)}.D^{\Pi(z_i)}$ and returns the challenge signature σ^*. Therefore, $Adv_3 = Adv_2$.

G_4: We modify the game G_3. Here we initialize Groth-Sahai commitment keys in a perfectly hiding setting with the trapdoor, to allow the simulator to cheat in the proofs. Then all the proofs are simulated. This game is indistinguishable from the

previous one under the SXDH. Thus $|Pr[S_4] - Pr[S_3]| = 2.Adv^{\text{SXDH}}$.

$\mathsf{G_5}$: In this game, we do not know anymore y, that we did not use anymore anyway, and thus this game is perfectly indistinguishable from previous one. Thus $Pr[S_5] = Pr[S_4]$.

$\mathsf{G_6}$: In this game, we replaces the defined $\ell-$DDHI tuple with the actual $\ell-$DDHI challenge instance, where y is unknown to $\mathcal{B}$ and D is a random value. Thus this game is indistinguishable from the previous one under the $\ell-$DDHI assumption. Therefore, $|Pr[S_6] - Pr[S_5]| \leq Adv^{\ell-\text{DDHI}}$.
Note that the challenger signature does not depend anymore on the challenge user. When we combine all the probabilities we obtain the upper bound (6.8) on $\mathcal{A}$'s advantage in game $\mathsf{G_1}$. □

Theorem 6.4.4 *The proposed scheme preserves the attribute unforgeability under KEA1 and DL assumptions.*

Proof Lemma 6.4.5 implies the Theorem 6.4.4. □

Lemma 6.4.5 *Under the DL and KEA1 assumptions there exists no PPT adversary $\mathcal{A}$ which passes verification with forged attributes with non negligible probability.*

Proof The input to the simulator $\mathcal{B}$ is an instance of the DL problem, $(g, g') \in \mathbb{G}_1^2$. Let $\xi = log_g g'$.
Setup: According to the ABGS scheme setup $\mathcal{B}$ generates the system parameters, *params*. $\mathcal{B}$ sets $g_1 = g$ and $h = g'$, and generate the remaining parameters, $gpk, ik, ok_{user}, tk_{att}$. $\mathcal{B}$ gives gpk, ok_{user} and tk_{att} to $\mathcal{A}$.
Queries: As $\mathcal{B}$ knows all the keys, it can answer all the queries generated by an adversary $\mathcal{A}$ according to the definition of *Attribute Unforgeability* .
Output: Finally, $\mathcal{A}$ outputs a signature σ^* with forged attribute certificates on message M^*, a predicate Υ^* whose public values are $\mathcal{T}_{\Upsilon^*} = (\{s_{d_j}\}_{d_j \in D_{T_{\Upsilon^*}}}, v_T = g_2^{s_T}, T^{\text{ext}})$,

and signer's secret key sk_{i^*} such that $\Upsilon(\mathcal{A}_{i^*}) \neq 1$ and $\Upsilon(\mathcal{A}_{i^*} \cup att_j) = 1$. As it is a valid signature which passes verification algorithm and from (6.2) $\rho_4^* = h^{\frac{s_{T_1}}{\gamma + x_{i^*}}}$ and $\rho_5^* = h^{s_{T_2}}$ such that $s_{T_1} + s_{T_2} = s_T$. This can be viewed as $\rho_4^* = h^{\frac{s'_{T_1}}{\gamma + x_{i^*}}} h^{\frac{s_j}{\gamma + x_{i^*}}}$, where $s_{T_1} = s'_{T_1} + s_j$ and $h^{\frac{s_j}{\gamma + x_{i^*}}}$ is unknown to $\mathcal{A}$ but she is producing it in signature.
It is like $\mathcal{B}$ is giving input $(g_1 = g, g_1^{s_j} = g_{att_j} = g^{s_j})$ to $\mathcal{A}$ and $\mathcal{A}$ implicitly returns $(h = g', h^{s_j} = g'^{s_j})$. Then by KEA1 assumption, $\mathcal{B}$ can utilize the extractor $\bar{\mathcal{A}}$ to extract a value ξ. Under DL assumption it can be done with negligible probability. Thus the signature produced by the forged attribute certificates can pass verification with negligible probability. Note that the $\mathcal{A}$ can also produce the missing attribute in the value ρ_5^* to satisfy the relation (6.2) but similarly its probability is negligible under KEA1 and DL assumption. □

Theorem 6.4.6 *The proposed scheme preserves the collusion resistance of attribute.*

Proof Lemma 6.4.7 implies the Theorem 6.4.6.

Lemma 6.4.7 *Even if some malicious participants* $\mathrm{U}_{i_1}, ..., \mathrm{U}_{i_k} (k > 1)$ *with the set of attributes* $\zeta_{i_1}, ..., \zeta_{i_k}$ *collude, they cannot make a valid signature associated with a predicate* Υ, *where* $(\cup_{j=1}^{k} \Upsilon(\zeta_{i_j}) = 1)$ *and* $\Upsilon(\zeta_{i_j}) \neq 1$ *for* $j = 1, ..., k$ *with non-negligible probability.*

Proof Without loss of generality, we assume that U_{i_1} with ζ_{i_1} and U_{i_2} with ζ_{i_2} represent malicious participants. U_{i_1} and U_{i_2} attempt to make a valid signature associated with Υ which satisfies $\Upsilon(\zeta_{i_1} \cup \zeta_{i_2}) = 1, \Upsilon(\zeta_{i_1}) \neq 1$ and $\Upsilon(\zeta_{i_2}) \neq 1$. They can satisfy the relations (6.1) and (6.2) because they have a valid membership certificate (A_{i_1}, X, y_{i_1}). We assume that $T_{i_1,j}^t = T_{i_2,j}$, where $t \in \mathbb{Z}_p^*$. Note that the probability of $t = 1$ is negligible. And they tries to compute

$$\rho_6 = h^{\frac{1}{\gamma + x_{i_1}} (\Sigma_{\mathrm{att}_j \in \mathcal{A}_{\mathrm{i}_1}} \Delta_{\mathrm{att}_j} s_j)} \times h^{\frac{1}{\gamma + x_{i_2}} (\Sigma_{\mathrm{att}_j \in \mathcal{A}_{\mathrm{i}_2}} \Delta_{\mathrm{att}_j} s_j)}$$

$$= h^{\frac{1}{\gamma + x_{i_1}} (\Sigma_{\mathrm{att}_j \in \mathcal{A}_{\mathrm{i}_1}} \Delta_{\mathrm{att}_j} s_j + t \Sigma_{\mathrm{att}_j \in \mathcal{A}_{\mathrm{i}_2}} \Delta_{\mathrm{att}_j} s_j)}$$

Then from (2.6)

$$\Sigma_{\mathrm{att}_j \in \mathcal{A}_{\mathrm{i}_1}} \Delta_{\mathrm{att}_j} s_j + t\Sigma_{\mathrm{att}_j \in \mathcal{A}_{\mathrm{i}_2}} \Delta_{\mathrm{att}_j} s_j + \Sigma_{d_j \in D_T^{\zeta}} \Delta_{d_j} s_{d_j} \neq s_T$$

holds. Since $t \neq 1$ this means that they cannot collude. □

Theorem 6.4.8 *If there exists an adversary $\mathcal{A}$ that breaks the traceability of the scheme, then we can build an adversary $\mathcal{B}$ that can break the q-HHSDH assumption, where q is the maximal number of users.*

Proof Since the membership certificate format is similar to the one proposed in [24; 48], the proof directly reduces to the q-HHSDH assumption. The simulator $\mathcal{B}$ receives q-HHSDH challenge $(g_1, h, g_2, \omega = g_2^{\gamma}, (g_1^{x_i}, g_2^{x_i}, y_i, A_i = (hg_1^{y_i})^{1/(\gamma+x_i)})_{i\in[1,q]})$ and tries to solve it, from $\mathcal{A}$ that breaks the traceability of our scheme.
Setup: $\mathcal{B}$ generates the commitment keys, attribute secret and public values, and other parameters as in the ABGS scheme by using the q-HHSDH challenge values. $\mathcal{B}$ gives gpk, ok_{user} and tk_{att} to $\mathcal{A}$.
Queries: To answer the i-th join queries, if this is an active join, $\mathcal{B}$ extracts y_i' chooses his y_i'' so that $y_i' + y_i'' = y_i$, if it is a passive join, $\mathcal{B}$ directly chooses y_i. Thus $\mathcal{B}$ can answer all the queries according to traceability definition.
Output: After atmost q join quires, $\mathcal{A}$ outputs a new signature with a new certificate tuple with non-negligible probability. As $\mathcal{B}$ knows the trapdoor of the commitment scheme, he can obtain $(g_1^x, g_2^x, g_1^y, g_2^y, A = (hg_1^y)^{1/(\gamma+x)})$. Thus $\mathcal{B}$ answers the challenge q-HHSDH instance with the same advantage of $\mathcal{A}$. □

Theorem 6.4.9 *If there exists an adversary $\mathcal{A}$ that breaks the non-frameability of the scheme, then we can build an adversary $\mathcal{B}$ that can either break the q-HSDH or the CDH$^+$ computational problems, or the 1-DDHI or the SXDH decisional problems, where q is the maximal number of signing queries for a user.*

Proof The proof is similar to the proof of non-frameability in the Blazy and Pointcheval [24] traceable signature. There exist two types of adversary, one breaks the non-frameability by forging the new ID, ρ_6, on an uncorrupted user and another breaks

the non-frameability by reusing an existing ID with the corresponding certificate but on a new message. With 1/2 probability $\mathcal{B}$ decides which type of adversary it is.
Type 1: The simulator $\mathcal{B}$ receives q-HSDH challenge $((g_1, g_2, g_1^y, g_2^y), (g_1^{t_i}, g_2^{t_i}, g_1^{1/(y+t_i)})_{i\in[1,q]})$ and tries to solve it, from an adversary $\mathcal{A}$ that breaks the non-frameability of our scheme by forging a new ID, ρ_6, on an uncorrupted user.

Setup: $\mathcal{B}$ generates the gpk, ik, ok_{user} and tk_{att} as the real settings and gives it to $\mathcal{A}$. $\mathcal{B}$ selects the target user on which he expects the attack and sets his membership certificate corresponding to one with y as a secret key.

Queries: $\mathcal{B}$ can answer any joinP query as he knows ik and can answer $\mathsf{corrupt}$ query on any user except the target user, otherwise the simulation fails. $\mathcal{B}$ can answer the sign queries and can answer to atmost q sign queries for the target user with the help of challenge q-HSDH tuple.

Output: After all the queries and the atmost q signing queries for target user, $\mathcal{A}$ succeeds in breaking the non-frameability with non-negligible probability by generating a new tuple $(\rho_6 = g_1^{1/(y+t)}, \rho_7 = (g_1^t, g_2^t))$, on an uncorrupted user. Thus $\mathcal{B}$ solves the q-HSDH challenge with non-negligible probability.

Type 2: The simulator $\mathcal{B}$ is given an asymmetric Waters public key $(pk = (g_1^t, g_2^t)$ for the global parameters $(g_1, g_2, h, \mathcal{F}))$. $\mathcal{B}$ tries to break this signature, and thus the CDH$^+$ problem, from an adversary $\mathcal{A}$ breaking the non-frameability of our scheme by reusing an existing tuple ρ_6, ρ_7 on a new message.
In the first game, G_1, $\mathcal{B}$ knows the discrete logarithm value t, generates a new ik, ok_{user}, tk_{att} and gives ik, ok_{user}, tk_{att} to $\mathcal{A}$ together with the public key $gpk = (h, \omega, \mathcal{F}, \{g_{\mathrm{att}_j}\}_{\mathrm{att}_j\in Att})$. $\mathcal{B}$ can answer any joinP query as he knows ik and extract the secret keys from the extraction key of the commitment scheme, one of those uncorrupted user is expected to be a challenge user, with the secret key y, the one $\mathcal{A}$ has to frame.
$\mathcal{B}$ can answer any signing queries. On one of them for the challenge user, say on M,

he will use the above ξ as ephemeral Waters public key (for the z), and thus computes a $\rho_6 = g_1^{1/(y+t)}$ with the corresponding Groth-Sahai proof. This way $\mathcal{A}$ possesses a valid signature on M, with $\rho_7 = (g_1^t, g_2^t), \rho_8 = h^t \mathcal{F}(\mathcal{M})^s, \rho_9 = g_2^s$. With non-negligible probability $\mathcal{A}$ breaks the non-frameability of our scheme, by hypothesis $\mathcal{A}$ does it by reusing an existing $\rho_1, ..., \rho_7$, as uncorrupted users are indistinguishable, $\mathcal{A}$ frames our challenge user with non-negligible probability, and as the signing queries are finite, he will use $\rho_7 = (g_1^t, g_2^t)$ with non-negligible probability. Therefore, with non-negligible probability $\mathcal{A}$ outputs a new valid signature on M' with $\rho_7 = (g_1^t, g_2^t)$, this means we have (ρ_7, ρ_8, ρ_9) such that $e(\rho_{7,1}, g_2) = e(g_1, \rho_{7,2}), e(\rho_8, g_2) = e(h, \rho_{7,2}) \times e(\mathcal{F}(\mathcal{M}'), \rho_9)$, and thus $\mathcal{B}$ can output a valid forgery on the Waters challenge for the public key (g_1^t, g_2^t). But in this game, we know t.
In a second game, $\mathsf{G_2}$, the Groth-Sahai setup is used as hiding one, so that the proofs can be simulated, and namely without using t. This is indistinguishable from the previous game under the SXDH assumption.
In the third game, $\mathsf{G_3}$, replace ρ_6 by a random value, still simulating the proofs. A random ρ_6 is indistinguishable from the real one under the DDHI problem as seen in user anonymity proof. Furthermore, here there is only one elements, hence the $1-$DDHI assumption. In the last game, one does not need to know t anymore, and thus the signature forgery reduces to breaking the asymmetric CDH$^+$. □

6.5 Comparison

Let $\Phi = |\zeta|$, where ζ be the set of attributes associated with a signature and $m = |Att|$. Let $\hat{m}$ be the average number of attributes assigned to any user and m' is the length of the message, a constant. e represents the paring operation and r represents the number of revoked members. In Table 6.1, we compare the efficiency of our scheme with other schemes. Note that the verification cost of the proposed scheme is independent of the number of attributes, where as in other schemes the verification cost is linear in terms of the number of attributes. From the table it can be noticed that non-frameability is not achieved by combined scheme of Herranz et al. [70] and Boyen et al. [34]. Further, the combined scheme has verification cost that is not independent of the number of attributes and also the key lengths are large.

Table 6.1: Comparison of ABGS scheme in standard model with other schemes

	Khader [72]	Emura et al. [51]	Herranz et al. [70] +Boyen et al. [34]	Our Scheme
CCA2−User Anonymity	no	yes	no	yes
Attribute anonymity	no	no	yes	yes
Non-frameability	no	yes	no	yes
Attribute revocation	yes	no	no	no
Predicate	monotone	monotone	threshold	monotone
Signature length	$O(\Phi)$	$O(\Phi)$	$15\|\mathbb{G}_p\| + 6\|\mathbb{G}_n\| = O(1)$	$29\|\mathbb{G}_1\| + 20\|\mathbb{G}_2\| = O(1)$
User's Secret Key Length	$(\hat{m}+1)\|\mathbb{G}_1\| + \|\mathbb{Z}_p^*\|$	$(\hat{m}+1)\|\mathbb{G}_1\| + 2\|\mathbb{Z}_p^*\|$	$(m+\hat{m})\|\mathbb{G}_p\| + 3\|\mathbb{G}_n\|$	$(2+\hat{m})\|\mathbb{G}_1\| + \|\mathbb{G}_2\| + \|\mathbb{Z}_p^*\|$
Assumption	DLIN, q-SDH	DL,DDH, q-SDH	DLin,(ℓ,m,t)−aMSE−CDH, q−HSDH,DTDH	DL,q−HSDH, SXDH,q−HHSDH, ℓ−DDHI,,CDH$^+$, KEA1
Model	Random Oracle	Random Oracle	Standard	Standard
Verification cost	$(6+2r)\mathbb{G}_1+(8+2\Phi)\mathbb{G}_T+(\Phi+2r+1)e = O(\Phi+r)$	$(11+2\Phi)\mathbb{G}_1+(\Phi+1)\mathbb{G}_2+14\mathbb{G}_T+6e = O(\Phi)$	$(4m+6m')\mathbb{G}_p+21\mathbb{G}_T+33e+(m'+1)\mathbb{G}_p+5\mathbb{G}_T+6e = O(m)$	$O(1)$

6.6 Summary

In this chapter, we proposed an ABGS scheme which achieves attribute anonymity with constant signature size. We proved that it is secure under the standard model. In this scheme, the user opening (or the signer tracing) and the attribute tracing methods are independent. Our scheme is dynamic with respect to both user and attribute i.e. anytime a user can join or attributes can be added without changing the keys. We note that our scheme is efficient than the other ABGS schemes in terms of verification cost and signature length. Moreover, the proposed scheme is non-frameable. In the next chapter, we propose an ABGS scheme with VRL and backward unlinkability in the standard model.

Chapter 7

A VLR-ABGS Scheme with Backward Unlinkability and Attribute Anonymity in the Standard Model

In the previous chapters, we proposed an ABGS schemes with attribute anonymity secure under random oracle model and standard model. We also proposed a VLR-ABGS scheme with attribute anonymity in the random oracle model. In this chapter, we propose an enhanced ABGS scheme with verifier-local revocation (VLR) in the standard model.

7.1 Introduction

Khader proposed an ABGS scheme with VLR feature but does not address attribute anonymity [72]. Afterwards Emura et al. in [52] have proposed an ABGS scheme, but this scheme neither addresses the attribute anonymity issue nor provides the revocation feature. To the best of our knowledge there is only one ABGS scheme

with VLR feature proposed by Khader in [72] but the scheme does not have *backward unlinkability* feature nor addressed attribute anonymity nor it is in standard model. Moreover, in this scheme the signature length is linear in terms of the number of attributes.

We propose an ABGS scheme with VLR feature (VLR-ABGS) with attribute anonymity having backward unlinkability in the standard model [5]. Further, our scheme has constant signature length and is non-frameable. We note that to build a VLR-ABGS scheme with attribute anonymity in the standard model one can also combine an ABS scheme [70] with a VLR-GS scheme [82], but it incurs combined cost of both the schemes.

In Section 7.2, we present the proposed VLR-ABGS scheme with attribute anonymity and the related security definitions. The construction of the proposed VLR-ABGS scheme is given in Section 7.3. Its security analysis is given in Section 7.4. followed by comparison with other schemes in Section 7.5. Finally we summarize in Section 7.6.

7.2 Proposed Scheme

In this section, we present our proposed VLR-ABGS scheme with the security definitions which are similar to [20; 51; 72; 82] but with the added attribute anonymity and backward unlinkability feature. In verifier-local revocation group signatures (VLR-GS)[31], the group manager maintains a periodically updated revocation list (RL) which is used by all verifiers to perform the revocation test and it makes sure that the signatures are not produced by a revoked member.

Let $\texttt{GM}$ be the group manager, k the security parameter, $params$ the system parameters, $\mathbb{T}$ denotes the number of time intervals (it is bounded with k), Att the universal set of attributes, Υ used to denote a predicate, $\Upsilon(\zeta) = 1$ denotes that the attribute set $\zeta \subseteq Att$ satisfies the predicate Υ, gpk the group public key, ik the issuing key used for issuing private keys to the members, ok_{user} the user opening key used to open the user identity of the group signature, $\mathcal{A}_i \subseteq Att$ the set of attributes assigned

to the user U_i, sk_i denotes the private key for the member U_i, $\vec{reg}$ be the registration table with the GM where the current group members information are stored, RL_t be the set of revocation tokens of interval t, which contains the revocation tokens of the revoked users at the interval t and $\vec{grt}$ be the $(n \times \mathbb{T})$-vector of revocation tokens, $\vec{grt} = \{grt[1][1], ..., grt[n][\mathbb{T}]\}$, where $grt[i][t]$ denotes the token of member i at interval t and $n = O(k)$ is the maximal number of users. Note that ik includes $\vec{grt}$ and is private with GM.
A user U_i can make a group signature on a document M with the predicate Υ during the interval t if there exists a set of attributes $\zeta \subseteq \mathcal{A}_i$ with the user such that $\Upsilon(\zeta) = 1$ and $grt[i][t] \notin RL_t$.

Definition 7.2.1 (VLR-ABGS) *An VLR-ABGS scheme consists of following algorithms. Unless otherwise indicated, algorithms are randomized.*

- $params \leftarrow \texttt{Setup}(1^k)$: *This algorithm takes the security parameter k as an input and returns the system parameter params.*
- $(gpk, ik, ok_{user}) \leftarrow \texttt{KeyGen}(params)$: *This algorithm takes the system parameter params, and returns a group public key gpk, an issuing key ik and a user opening key ok_{user}.*
- $sk_i \leftarrow \texttt{Join}(\langle params, gpk, ik, t, upk_i, \mathcal{A}_i\rangle, \langle params, gpk, t, upk_i, usk_i\rangle)$: *This is an interactive group joining protocol between a user U_i (using his secret key usk_i and the current interval t) and the GM (using the issuing key ik and the attributes $\mathcal{A}_i \subseteq Att$ for U_i). In this protocol U_i ends with a member private key sk_i and GM ends with an updated, registration table $\vec{reg}$ and vector of revocation tokens $\vec{grt}$.*
- $\sigma \leftarrow \texttt{Sign}(params, gpk, sk_i, t, \zeta, M, \Upsilon)$: *This algorithm takes params, gpk, t, sk_i, an attribute set $\zeta \subseteq \mathcal{A}_i$, message M, and the predicate Υ as an input and returns a group signature σ on M at the interval t.*

- $0/1 \leftarrow \texttt{Verify}(params, gpk, t, M, \Upsilon, \sigma, RL_t)$: *This is a deterministic algorithm verifies the validity of the group signature σ against gpk and returns* $1/0$. *If* 1 *then the algorithm claims that the σ is a valid group signature, otherwise, σ is invalid.*

- $i/\perp \leftarrow \texttt{Open}(params, gpk, ok_{user}, t, \sigma, M, \Upsilon, \vec{reg})$*: This is a deterministic algorithm which takes as input* $params, gpk, ok_{user}, \sigma, \Upsilon, M$ *and* $\vec{reg}$*, and returns either $i \geq 1$ or $\perp$. If i, the algorithm claims that the group member with identity i has produced σ during the time interval t, and if $\perp$, then no group member produced σ.*

Revoke: When the member U_i is need to be revoke in the interval t, the `GM` publishes (or adds) the secret tokens $grt[i][t], ..., grt[i][\mathbb{T}]$ into the public lists $RL_t, RL_{t+1}, ..., RL_{\mathbb{T}}$, respectively.

Entities: Following are the entities in ABGS scheme:

- The group manager `GM`, also known as *issuer*, has issuing key ik using which he enrolls a user into the group by allotting some privileges (in terms of attributes) say $\mathcal{A}_i \subseteq Att$ and issuing a user's private key sk_i, by running interactive `Join` algorithm with the user. Issuer revoke a group member by publishing the revocation token of the member and also can reveal the signer's identity from the group signature by using $\vec{grt}$.

- The *opener* has user opening key ok_{user} by which he is able to opens the signature and reveals the user identity through `Open` algorithm.

- Group members, or signers, who are having their private keys sk_i. They run `Sign` algorithm to produce a group signature on a document M with predicate Υ;if they possess valid attribute set $\mathcal{A}_i$which satisfies the predicate.

- Outsider or verifier who can seek a group signature for a document M with predicate Υ from group manager `GM`. He can also verify the group signature using the group public key, gpk.

ABGS scheme is correct if the group signatures produced by an honest group member are verified and reveals the identity of the signer.

Definition 7.2.2 (Correctness) Correctness *requires that for all* $params \leftarrow \mathtt{Setup}(1^k)$, *all* $(gpk, ik, ok_{user}) \leftarrow \mathtt{KeyGen}(params), sk_i \leftarrow \mathtt{Join}(\langle params, gpk, ik, t, upk_i, \mathcal{A}_i\rangle, \langle params, gpk, t, upk_i, usk_i\rangle)$, , *all* $t \in \{1, ..., \mathbb{T}\}$, *all* Υ, *all* $\zeta \subseteq Att$ *and all* $M \in \{0,1\}^*$, *if* $\mathrm{U}_j \in r\vec{e}g$, $\zeta \subseteq \mathcal{A}_j, \Upsilon(\zeta) = 1, grt[j][\tilde{t}] \notin RL_{\tilde{t}}$ *and* $\sigma = \mathtt{Sign}(params, gpk, sk_j, \tilde{t}, \zeta, M, \Upsilon)$ *then*

$$\begin{aligned} 1 &\leftarrow \mathtt{Verify}(params, gpk, \tilde{t}, M, \Upsilon, \sigma, RL_t) \\ \bigwedge j &\leftarrow \mathtt{Open}(params, gpk, ok_{user}, \tilde{t}, \sigma, M, \Upsilon, r\vec{e}g) \end{aligned}$$

holds.

In the following definitions the adversary can run the Join protocol (similarly to [24]):

- either through the joinP-oracle (passive join), which means that it creates an honest user for whom it does not know the private keys: the index i is added to the HU (Honest Users) list;
- or through the joinA-oracle (active join), which means that it interacts with the group manager to create a user it will control: the index i is added to the CU (Corrupted Users) list.

Note that when the adversary is given the issuing key (the group manager is corrupted) then the adversary does not need access to the joinA oracle since it can simulate it by itself, to create corrupted users (that are not necessarily in CU). After a user is created, the adversary plays the role of corrupted users, and can interact with honest users, granted some oracles:

- sign(i, M, Υ), if $i \in$ HU, plays as the honest user i would do in the signature process to generate a signature on message M with predicate Υ;

- $\mathsf{open}(M, \sigma, \Upsilon)$, if (M, Υ, σ) is valid, returns the identity i of the signer;
- $\mathsf{corrupt}(i)$, if $i \in \mathrm{HU}$, provides the specific private key of this user. The adversary can now control it during the whole simulation. Therefore i is moved from HU to CU;
- $\mathsf{revoke}(i, t)$, if $i \in \mathrm{HU}$, returns the member i's revocation token for the current period t.

In ABGS scheme a group member may have multiple attribute sets to satisfy the predicate and he can produce a group signature using one of them. An ABGS scheme preserves attribute anonymity if it is computationally difficult to identify with what attribute set he produces the signature.

Definition 7.2.3 (`Attribute anonymity`) *We say that the VLR-ABGS scheme preserves* attribute anonymity *if, for all honestly generated* $(gpk, ik, ok_{user}) \leftarrow \mathtt{KeyGen}($ *params*$)$*, for all predicates* Υ*, for all attribute sets* $\mathcal{A}_i \subseteq Att$ *such that there exist* $\zeta_1, \zeta_2 \subseteq \mathcal{A}_i$ *and* $\Upsilon(\zeta_1) = \Upsilon(\zeta_2) = 1$*, for all* $sk_i \leftarrow \mathtt{Join}(\langle params, gpk, ik, t, upk_i, \mathcal{A}_i \rangle, \langle params, gpk, t, upk_i, usk_i \rangle)$ *and all messages* M*, the distributions* $\mathtt{Sign}(params, gpk, sk_i, t, \zeta_1, M, \Upsilon)$ *and* $\mathtt{Sign}(params, gpk, sk_i, t, \zeta_2, M, \Upsilon)$ *are identical.*

In other words, even the computationally unbounded adversary cannot link a signature to a set of attributes used to generate it (similar to [85]).

ABGS scheme preserves backward unlinkability - user anonymity if there are at least two unrevoked group members possessing valid attribute sets and one of them produces the group signature then it should be computationally hard to identify who produced the group signature among them, even if they are revoked afterwards.

Definition 7.2.4 (`BU-user anonymity`) *We say that the VLR-ABGS scheme preserves* BU-user anonymity *if for all PPT* $\mathcal{A}$*, the probability that* $\mathcal{A}$ *wins the following game is negligible.*

- ***Setup:*** *The challenger $\mathcal{C}$ runs $(gpk, ik, ok_{user}) \leftarrow \texttt{KeyGen}(params)$. $\mathcal{C}$ gives gpk to $\mathcal{A}$.*
- ***Queries:***
  - `Phase1` : *$\mathcal{A}$ is given access to the oracles:* joinP, joinA, corrupt, sign, revoke *and* open.
  - `Challenge` : *At some period $t^* \in \{1, ..., \mathbb{T}\}$, $\mathcal{A}$ outputs M^*, Υ^* and an uncorrupted unrevoked users $\mathrm{U}_{i_0}, \mathrm{U}_{i_1}$ (i.e. $i_0, i_1 \notin$ CU and not queried to* revoke *before or during t^*) and, $\zeta : \zeta \subseteq \mathcal{A}_{i_0}, \zeta \subseteq \mathcal{A}_{i_1}$ and $\Upsilon(\zeta^1) = 1$. $\mathcal{C}$ randomly selects $\kappa \in_R \{0, 1\}$ and responds with a group signature $\sigma^* \leftarrow \texttt{Sign}(params, gpk, t^*, sk_{i_\kappa}, \zeta, M, \Upsilon)$.*
  - `Phase 2` : *$\mathcal{A}$ can make queries similar to Phase 1. However $\mathcal{A}$ cannot make query to* corrupt *on i_0 and i_1 at any time but can query to* revoke *for the intervals after t^*.*
- ***Output:*** *Finally, $\mathcal{A}$ outputs a bit κ', and wins if $\kappa' = \kappa$.*

The advantage of $\mathcal{A}$ is defined as $Adv^{BU-user-anon}(\mathcal{A}) = |Pr(\kappa = \kappa') - \frac{1}{2}|$.

Thus there should not exist any PPT adversary to link a group signature to a signer with non-negligible probability.

ABGS scheme preserves traceability if it is possible to trace the valid group signature to its signer with the help of group opening key.

Definition 7.2.5 (`Traceability`) *We say that the VLR-ABGS scheme preserves* traceability *if for all PPT $\mathcal{A}$, the probability that $\mathcal{A}$ wins the following game is negligible.*

[1]Here ζ can be different for $\mathrm{U}_{i_0}, \mathrm{U}_{i_1}$ but we are concerned about user anonymity rather than attribute anonymity

- ***Setup:*** *The challenger $\mathcal{C}$ runs $(gpk, ik, ok_{user}) \leftarrow \mathtt{KeyGen}(params)$. $\mathcal{C}$ gives gpk, ok_{user} to $\mathcal{A}$.*
- ***Queries:*** *$\mathcal{A}$ is given access to the oracles:* $\mathsf{joinP}, \mathsf{joinA}, \mathsf{corrupt}, \mathsf{revoke}$ *and* sign.
- ***Output:*** *At some period $t^* \in \{1, ..., \mathbb{T}\}$, $\mathcal{A}$ outputs a group signature σ^*, a message M^*, a predicate Υ^* and a set of revocation tokens RL_{t^*}.*

$\mathcal{A}$ wins if
(1) $\mathtt{Verify}(params, gpk, t^*, M^*, \Upsilon^*, RL_t, \sigma^*) = 1$ *and*
(2) $\mathtt{Open}(params, gpk, ok_{user}, t^*, \sigma^*, M^*, \Upsilon^*, r\vec{e}g) = \perp$.
The advantage of $\mathcal{A}$ is defined as the probability that $\mathcal{A}$ wins.

Thus it should be impossible to produce an untraceable valid group signature by any PPT adversary.

ABGS scheme preserves non-frameability if it is difficult to produce a valid group signature which traces back to a group member who has not produce it, even with the help of group manager's secret key.

Definition 7.2.6 (`Non-frameability`) *We say that the VLR-ABGS scheme preserves* non-frameability *if for all PPT $\mathcal{A}$, the probability that $\mathcal{A}$ wins the following game is negligible.*

- ***Setup:*** *The challenger $\mathcal{C}$ runs $(gpk, ik, ok_{user}) \leftarrow \mathtt{KeyGen}(params)$. $\mathcal{C}$ gives gpk, ik* and *ok_{user} to $\mathcal{A}$.*
- ***Queries:*** *$\mathcal{A}$ is given access to the oracles:* $\mathsf{joinP}, \mathsf{corrupt}$ *and* sign.
- ***Output:*** *Finally, at some period $t^* \in \{1, ..., \mathbb{T}\}$, $\mathcal{A}$ outputs a group signature σ^*, a message M^*, a predicate Υ^* and a set of revocation tokens RL_{t^*}.*

$\mathcal{A}$ wins if
(1) $\texttt{Verify}(params, gpk, t^*, M^*, \Upsilon^*, \sigma^*, RL_t) = 1$,
(2) $\texttt{Open}(params, gpk, ok_{user}, t^*, \sigma^*, M^*, \Upsilon^*, r\vec{e}g) = i^*$,
(3) $i^* \in \mathsf{HU}$.
The advantage of $\mathcal{A}$ is defined as the probability that $\mathcal{A}$ wins.

Thus even the group manager should not be able to forge a group signature which trace back to a honest member.

ABGS scheme preserves attribute unforgeability if it is hard for a group member to forge an attribute certificate in order to produce a valid group signature.

Definition 7.2.7 (`Attribute unforgeability`) *We say that the VLR-ABGS scheme preserves* attribute unforgeability *if for all PPT $\mathcal{A}$, the probability that $\mathcal{A}$ wins the following game is negligible.*

- ***Setup:*** *The challenger $\mathcal{C}$ runs* $(gpk, ik, ok_{user}) \leftarrow \texttt{KeyGen}(params)$. *$\mathcal{C}$ gives* gpk, ok_{user} *to $\mathcal{A}$.*
- ***Queries:*** *$\mathcal{A}$ is given access to the oracles:* $\mathsf{joinP}, \mathsf{joinA}, \mathsf{corrupt}, \mathsf{revoke}$ *and* sign.
- ***Output:*** *At some period* $t^* \in \{1, ..., \mathbb{T}\}$, *$\mathcal{A}$ outputs a message M^*, a predicate Υ^* and a group signature σ^*.*

$\mathcal{A}$ wins if
(1) $\texttt{Verify}(params, gpk, t^*, M^*, \Upsilon^*, \sigma^*, RL_t) = 1$,
(2) $\texttt{Open}(params, gpk, ok_{user}, t^*, \sigma^*, M^*, \Upsilon^*, r\vec{e}g) = i^*$ *and*
(3) $\nexists \zeta \in \mathcal{A}_{i^*} : \Upsilon(\zeta) = 1$.
The advantage of $\mathcal{A}$ is defined as the probability that $\mathcal{A}$ wins.

Thus it should be impossible for any PPT adversary to satisfy the predicate with invalid set of attributes.

ABGS scheme preserves collusion resistance of attribute certificates if it is computationally hard for group members to collude by pooling their attribute certificates to satisfy the predicate and to produce a valid group signature.

Definition 7.2.8 (`Collusion resistance of attributes`) *We say that the VLR-ABGS scheme preserves* collusion resistance of attributes *if for all PPT $\mathcal{A}$, the probability that $\mathcal{A}$ wins the following game is negligible.*

- ***Setup:*** *The challenger $\mathcal{C}$ runs $(gpk, ik, ok_{user}) \leftarrow \texttt{KeyGen}(params)$. $\mathcal{C}$ gives gpk, ok_{user} to $\mathcal{A}$.*
- ***Queries:*** *$\mathcal{A}$ is given access to the oracles:* joinP, joinA, corrupt, revoke *and* sign.
- ***Output:*** *At some period t^* $\mathcal{A}$ outputs a message M^*, a predicate Υ^* and a group signature σ^*.*

$\mathcal{A}$ wins if
(1) $\texttt{Verify}(params, gpk, t^*, M^*, \Upsilon^*, \sigma^*, RL_t) = 1$, *and*
(2) $\mathcal{A}$ has obtained $sk_{i_1}, ..., sk_{i_k} : \Upsilon^*(\cup_{j=1}^{k} \mathcal{A}_{i_j}) = 1$ and $\Upsilon^*(\mathcal{A}_{i_j}) \neq 1$ for $j = 1, ..., k$.
The advantage of $\mathcal{A}$ is defined as the probability that $\mathcal{A}$ wins.

Thus the users with valid set of attributes each, cannot collude with each other to pool a valid attribute set for producing a valid group signature.

7.3 Construction

A construction of a VLR-ABGS scheme with attribute anonymity and backward unlinkability features is presented in this section. For our construction we use the membership certificate format of [24; 48] to achieve non-frameability. We use Groth-Sahai non-interactive proof system under SXDH assumption (see Sec. 2.4.7.1) to

generate the NIWI and NIZK proofs for the relations in the group signature. We make use of VLR-GS scheme of [82] as a base construction. Similar to [51] we generate the public values of the access tree representing a predicate. We device an `BuildTree-Validity` algorithm which gives provision to publicly verify the correctness of the generated public values of the predicate and with this we reduce the trust on group manager in producing public keys of the predicates. We device an idea to achieve attribute anonymity. In contrast to other existing ABGS schemes [51; 72; 73], our scheme is built in the standard model with attribute anonymity and achieves a constant size signature, i.e. independent of the number of attributes. To highlight the merits of the proposed scheme, we compare our scheme with other schemes in terms of efficiency, features and assumptions.

Let T_Υ be an access tree representing the predicate Υ, $\mathcal{T}_\Upsilon$ the public values associated with T_Υ, (upk_i, usk_i) the verification/signing key of a signature scheme $DSig$ for user U_i[1], $\mathcal{A}_i$ the membership certificate for U_i, $\{T_{i,j}\}_{\mathrm{att}_j \in \mathcal{A}_i}$ denotes the attribute certificates of U_i and $T_{i,j}$ is the attribute certificate of the attribute $att_j \in Att$ of user U_i.
Let $E : \mathbb{G}_1 \times \mathbb{G}_2^2 \to \mathbb{G}_T^2$ be a coordinate-wise pairing such that, for any $h \in \mathbb{G}_1$ and any vector $\vec{v} = (g, f) \in \mathbb{G}_2^2, E(h, \vec{v}) = (e(h, g), e(h, f))$. As in [68], we make use of asymmetric bilinear map $F : \mathbb{G}_1^2 \times \mathbb{G}_2^2 \to \mathbb{G}_T^4$ which is defined as, for any vectors $\vec{X} = (X_1, X_2) \in \mathbb{G}_1^2$ and $\vec{Y} = (Y_1, Y_2) \in \mathbb{G}_2^2, F(\vec{X}, \vec{Y})$ is the matrix of entry-wise pairings (i.e. containing $e(X_i, Y_j)$ in its entry (i, j)).
Also, for any $z \in \mathbb{G}_T, \iota_T(z)$ denotes the 2×2 matrix containing z in position $(2, 2)$ and 1 elsewhere. For group elements $X \in \mathbb{G}_1$, the notation $\iota_1(X)$ will denote the vector $(1, X) \in \mathbb{G}_1^2$ and for $Y \in \mathbb{G}_2, \iota_2(Y) = (1, Y) \in \mathbb{G}_2^2$.
For a polynomial number of scalars $z_i \in \mathbb{Z}_p^*$, and a pair $(g, g^y) \in \mathbb{G}_1^2$, the values $g^{1/(y+z_i)}$ looks to be random and independent [50]. We used this to build our identifier, $\mathrm{ID}(y, z_i) = g^{1/(y+z_i)}$, in the group signature. In the proof of user anonymity, the simulator will be able to choose the z_i prior to any interaction with the adversary and we depend on q-DDHI assumption [50].

[1] we assume that each group member has a personal public key upk_i, established and certified by a PKI, independently of any group authority, so that it has a means to sign a message, using a matching personal private key usk_i with him.

- `Setup`(1^k): It generates the system parameters, $params = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, \mathcal{H}, \mathbb{T}, Att)$; where

  (i) $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are the cyclic groups of prime order p, where $2^{k-1} < p < 2^k$.

  (ii) $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a bilinear map.

  (iii) g_1 and g_2 are the generators of the groups $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively.

  (iv) $\mathcal{H} : \{0,1\}^* \rightarrow \{0,1\}^{m'}$ is a collision-resistant hash function, where $m' = O(k)$.

  (v) $\mathbb{T}$ is the number of time periods.

  (vi) $Att = \{\text{att}_1, ..., \text{att}_m\}$, for $m = O(k)$ is the universal set of attributes[1].

- `KeyGen`($params$): It takes an input system parameters $params$ and outputs a group public key gpk, an issuing key ik and a user opening key ok_{user}.

  (i) Select the independent generators $k_1, v', v_1, ..., v_{m'} \in \mathbb{G}_1$, $h_1, ..., h_{\mathbb{T}} \in \mathbb{G}_2$ and define the Waters function, $\mathcal{F} : \{0,1\}^{m'} \rightarrow \mathbb{G}_1$, for $\mathcal{M} = (\mu_1, ..., \mu_{m'}) \in \{0,1\}^{m'} \mathcal{F}(\mathcal{M}) = v' \Pi_{j=1}^{m'} v_j^{\mu_j}$.

  (ii) Select $\gamma \in_R \mathbb{Z}_p^*$, and computes $\omega = g_2^{\gamma}$.

  (iii) For each $\text{att}_j \in Att$, choose a secret $s_j \in_R \mathbb{Z}_p^*$, sets $S = \{s_j\}_{\text{att}_j \in Att}$, and computes $g_{\text{att}_j} = g_1^{s_j}, \forall \text{att}_j \in Att$.

  (iv) For the Groth-Sahai proof under the instantiation based on SXDH assumption, choose a vectors $\vec{u} = (\vec{u}_1, \vec{u}_2 = \vec{u}_1^t)$ and $\vec{v} = (\vec{v}_1, \vec{v}_2 = \vec{v}_1^{t'})$, where $\vec{u}_1 = (g_1, g_1^{\alpha}) \in \mathbb{G}_1^2$ and $\vec{v}_1 = (g_2, g_2^{\alpha'}) \in \mathbb{G}_2^2$ for $t, t', \alpha, \alpha' \in_R \mathbb{Z}_p^*$. α and α' are trapdoor keys.

  (v) Outputs

  $$gpk = (k_1, h_1, ..., h_{\mathbb{T}}, \omega, \mathcal{F}, \{g_{\text{att}_j}\}_{\text{att}_j \in Att}, \vec{u}, \vec{v}), ik = (\gamma, S), ok_{user} = (\alpha, \alpha').$$

  The description of $\mathcal{F}$ includes the generators $v', v_1, ..., v_{m'}$.

- `BuildTree`(p, S, Υ):

[1]For $m = 1$ it becomes a group signature scheme, where all members possess that attribute.

(i) Let T_Υ be the tree that represents the predicate Υ.

(ii) Get extension tree $T^{\text{ext}} \leftarrow \texttt{AddDummyNode}(T_\Upsilon)$.

(iii) Get secret value for each dummy node and the secret value of the root of T^{ext} using $(\{s_{d_j}\}_{d_j \in D_T}, s_T) \leftarrow \texttt{AssignedVaule}(S, T^{\text{ext}})$.

(iv) Output the public values of Υ,

$$\mathcal{T}_\Upsilon = (\{s_{d_j}\}_{d_j \in D_{T_\Upsilon}}, v_T = g_2^{s_T}, T^{ext}).$$

The predicate public values $\mathcal{T}_\Upsilon$ are verifiable through `BuildTree-Validity` algorithm.

A verifier with the predicate approaches the `GM` for a group signature. `GM` runs `BuildTree` algorithm to generate the public values of the predicate Υ and stores it in a public repository. Then anyone among the group members who are eligible will generate a group signature by using the predicate public values. If predicate public values are already present in the public repository then `GM` need not invoke `BuildTree` algorithm.

- `BuildTree-Validity`$(params, gpk, \mathcal{T}_\Upsilon)$:

  (i) Randomly choose an attribute set, $Leaves \subseteq Att : \Upsilon(Leaves) = 1$ And gets the corresponding $\Delta_{\text{att}_j}(\forall \text{att}_j \in Leaves)$, and $\Delta_{d_j}(\forall d_j \in D_{T_\Upsilon}^{Leaves})$ by running `MakeSimplifiedTree`$(Leaves,$
  $T^{ext})$.

  (ii) Compute $g_{root} = \prod_{\text{att}_j \in Leaves} g_{\text{att}_j}^{\Delta_{\text{att}_j}} \times \prod_{d_j \in D_{T_\Upsilon}^{Leaves}} g_1^{s_{d_j}\Delta_{d_j}}$
  $= \prod_{\text{att}_j \in Leaves} g_1^{s_j \Delta_{\text{att}_j}} \times \prod_{d_j \in D_{T_\Upsilon}^{Leaves}} g_1^{s_{d_j}\Delta_{d_j}}$
  $= g_1^{\sum_{\text{att}_j \in Leaves} \Delta_{\text{att}_j} s_j + \sum_{d_j \in D_{T_\Upsilon}^{Leaves}} \Delta_{d_j} s_{d_j}} = g_1^{s_T}$ from (2.6).

  (iii) Verify whether $e(g_{root}, g_2) \stackrel{?}{=} e(g_1, v_T)$. If not then $\mathcal{T}_\Upsilon$ is the invalid public values of the predicate Υ.

- `Join`$(< params, gpk, t, ik, upk_i, \mathcal{A}_i >, < params, gpk, t, upk_i, usk_i >)$: A user U_i with the pair of keys (upk_i, usk_i) in the PKI, interacts with the group manager, with the issuing key ik and the set of attributes $\mathcal{A}_i$, to join the group.

This is similar to the `Join` protocol in [24; 48]. At the completion of this protocol, U_i gets private key $sk_i = ((A_i, X_i, y_i), \{T_{i,j}\}_{\mathrm{att}_j \in \mathcal{A}_i})$, where (A_i, X_i, y_i) is a *membership certificate*, $\{T_{i,j}\}_{\mathrm{att}_j \in \mathcal{A}_i}$ is the set of *attribute certificates* and $\mathcal{A}_i \subseteq Att$ is the set of U_i's attributes. And `GM` ends with the updated $r\vec{e}g$ and $g\vec{r}t$. The protocol begins as follows,

(i) U_i picks $y_i' \in_R \mathbb{Z}_p^*$, computes and sends $Y_i' = g_1^{y_i'}$, an extractable commitment of y_i'. Note that the trapdoor of the commitment will not be known to anybody except to the simulator in the security proof to be able to extract y_i'.

(ii) `GM` selects new $x_i \in \mathbb{Z}_p^*$ and a random $y_i'' \in_R \mathbb{Z}_p^*$, computes $A_i = (k_1 Y_i' Y_i'')^{\beta/(\gamma + x_i)}$, $X_{i,2} = g_2^{x_i}$ and $T_{i,j} = A_i^{s_j} (\forall \mathrm{att}_j \in \mathcal{A}_i)$, where $Y_i'' = g_1^{y_i''}$ and sends $y_i'', A_i, X_{i,2}, \{T_{i,j}\}_{\forall \mathrm{att}_j \in \mathcal{A}_i}$.

(iii) U_i checks whether $e(A_i, \omega g_2^{x_i}) = e(k_1, g_2) e(g_1, g_2)^{y_i' + y_i''}$. Then U_i computes $y_i = y_i' + y_i''$ and makes a signature $s_i = DSig_{usk_i}(A_i, X_{i,2}, Y_i)$.

(iv) `GM` verifies s_i under upk_i, appends the tuple $(i, upk_i, A_i, X_i = (X_{i,1} = g_1^{x_i}, X_{i,2}), Y_i, s_i)$ to $r\vec{e}g$ and add $\{grt[i][j] = h_j^{x_i}\}_{j=t}^{\mathbb{T}}$ to $g\vec{r}t$. Then `GM` sends $X_{i,1}$.

(v) U_i checks the relation $e(X_{i,1}, g_2) = e(g_1, X_{i,2})$. U_i owns an valid membership certificate (A_i, X_i, y_i) and attribute certificates $\{T_{i,j}\}_{\forall \mathrm{att}_j \in \mathcal{A}_i}$, where y_i is known to him only.

- `GM` chooses $s_{m+1} \in \mathbb{Z}_p^*$, and computes $g_{\mathrm{att}_{m+1}} = g_1^{s_{m+1}}$ when a new attribute att_{m+1} is added. Let U_i be issued $T_{i,m+1}$. Then `GM` computes and sends $T_{i,m+1} = A_i^{s_{m+1}}$ to U_i and also publish $g_{\mathrm{att}_{m+1}}$ into gpk. Thus attributes can be added anytime.
- When user U_i is to be revoked at time interval t, `GM` needs to simply add $grt[i][t]$ to the revocation list RL_t for the interval t.
- For the group members to verify the validity of their issued attribute certificates $\{T_{i,j}\}$, `GM` builds one general access tree such that it is satisfied by each individual attribute. `GM` will publish its public values and every

member U_i can verify their individual certificates by using the following equation

$$e(T_{i,j}^{\Delta_{\text{att}_j}}.A_i^{\Delta_{d_j}s_{d_j}}, g_2)=e(A_i, v_T) \tag{7.1}$$

where $(\Delta_{\text{att}_j}, \Delta_{d_j}) \leftarrow \texttt{MakeSimplifiedTree}(s_j, T^{ext})$ and s_{d_j} is the corresponding dummy node of s_j, such that $e(g_{att_j}^{\Delta_{\text{att}_j}}.g_1^{\Delta_{d_j}s_{d_j}}, k_2) \stackrel{?}{=} e(g_1, v_T)$.

- `Sign`$(params, gpk, sk_i, t, \zeta, j, M, \Upsilon)$: It generates a group signature σ on message $M \in \{0,1\}^*$ during the period t with the user private key sk_i who satisfy the predicate Υ with his subset of attributes $\zeta \subseteq \mathcal{A}_i : \Upsilon(\zeta) = 1$.

  (i) Get the public values of Υ, $\mathcal{T}_\Upsilon = (\{s_{d_j}\}_{d_j \in D_{T_\Upsilon}}, v_T, T^{ext})$, from the public repository.

  (ii) Let $s_{T_1} = \Sigma_{\text{att}_j \in \zeta} \Delta_{\text{att}_j} s_j$ and $s_{T_2} = \Sigma_{d_j \in D_T^\zeta} \Delta_{d_j} s_{d_j}$. Then from (2.6), $s_{T_1} + s_{T_2} = s_T$.

  (iii) Get $(\{\Delta_{\text{att}_j}\}_{(\forall \text{att}_j \in \zeta)}, \{\Delta_{d_j}\}_{(\forall d_j \in D_{T_\Upsilon}^\zeta)}) \leftarrow \texttt{MakeSimplifiedTree}(\zeta, T^{ext})$ and compute $s_{T_2} = \Sigma_{d_j \in D_T^\zeta} \Delta_{d_j} s_{d_j}$.

  (iv) Compute the hash value $\mathcal{M} = \mu_1, ..., \mu_{m'} = \mathcal{H}(t||M) \in \{0,1\}^{m'}$.

  (v) Creates an ephemeral ID, $\text{ID}(y_i, z) = g_1^{1/(z+y_i)}$, with a random $z \in \mathbb{Z}_p^*$.

  (vi) Select $r, \delta \in_R \mathbb{Z}_p^*$ and set $\rho_1 = \Pi_{\text{att}_j \in \zeta} T_{i,j}^{\Delta_{\text{att}_j}}.A_i^{s_{T_2}} = A_i^{s_T}, \rho_2 = y_i, \rho_3 = X_i = (g_1^{x_i}, g_2^{x_i}), \rho_4 = \text{ID}(y_i, z), \rho_5 = (g_1^z, g_2^z), \rho_6 = k_1^z \mathcal{F}(\mathcal{M})^r, \rho_7 = g_2^r, \rho_8 = h_t^\delta, \rho_9 = h_t, T_1 = g_1^\delta$ and $T_2 = e(g_1^{x_i}, h_t)^\delta$.

  (vii) Commit the group elements $\sigma_i = \mathcal{C}(\rho_i)$, for $i = \{1, 3, 8, 9\}$ and $\sigma_2 = (\mathcal{C}^{(1)}(\rho_2), \mathcal{C}^{(2)}(\rho_2))$.

  (viii) Compute the NIWI Groth-Sahai proofs for the committed variables ρ_1, ρ_2, ρ_3

satisfy the following equations

$$
\begin{aligned}
e(\rho_1, \omega\rho_{3,2}) &= e(k_1, v_T) \times e(g_1^{\rho_2}, v_T) && (7.2)\\
e(\rho_4, g_2^{\rho_2}\rho_{5,2}) &= e(g_1, g_2) && (7.3)\\
e(\rho_6, g_2) &= e(k_1, \rho_{5,2}) \times e(\mathcal{F}(\mathcal{M}), \rho_7) && (7.4)\\
e(g_1^{\rho_2}, g_2) &= e(g_1, g_2^{\rho_2}) && (7.5)\\
e(\rho_{3,1}, g_2) &= e(g_1, \rho_{3,2}) && (7.6)\\
e(\rho_{5,1}, g_2) &= e(g_1, \rho_{5,2}) && (7.7)
\end{aligned}
$$

(ix) Compute the NIZK Groth-Sahai proofs for the committed variables $\rho_{3,1}, \rho_8, \rho_9$ satisfy the following equations

$$
\begin{aligned}
T_2 &= e(\rho_{3,1}, \rho_8) && (7.8)\\
e(T_1, \rho_9) &= e(g_1, \rho_8) && (7.9)\\
e(g_1, \rho_9) &= e(g_1, h_t) && (7.10)
\end{aligned}
$$

(x) Output the signature

$$\sigma = (\{\sigma_i\}_{i=1}^{3}, \{\rho\}_{i=4}^{7}, \sigma_8, \sigma_9, T_1, T_2) \in \mathbb{G}_1^4 \times \mathbb{G}_2^2 \times \mathbb{G}_1^2 \times \mathbb{G}_2^2 \times \mathbb{G}_1^2 \times \mathbb{G}_2 \times \mathbb{G}_1 \times \mathbb{G}_2^5 \times \mathbb{G}_1 \times \mathbb{G}_T$$

We add the corresponding Groth-Sahai proofs $\{\Theta_i\}_{i=1}^{9}$ to the signature to prove the validity of the above pairing equations. Equation (7.2) is a pairing-product equation, establishes that the signer has a valid membership certificate issued through the `Join` algorithm (i.e. A_i is well-formed) as well as establishes that he possess the required attributes (attribute certificates) that satisfy the predicate Υ, and the Groth-Sahai proof requires 4 elements in each group, $\Theta_1 = (\vec{\pi_1}, \vec{\theta_1})$. Equation (7.3) is a linear pairing-product, establishes that ρ_4 is a well formed ID, and the proof requires only 2 extra elements in $\mathbb{G}_1$, $\Theta_2 = (\vec{\theta_2})$. Equation (7.4) does not use any committed data so it can be directly checked, it establishes that (ρ_6, ρ_7) is a Waters signature of $\mathcal{M}$ under the public key ρ_5, thus $\Theta_3 = \phi$. Equation (7.5) is a quadratic equation, establishes that same y_i is

committed in both groups, and it can prove with 2 elements in each group, $\Theta_4 = (\vec{\pi_4}, \vec{\theta_4})$. Equation (7.6) is a pairing-product equation, establishes that X_i is well-formed, and the proof requires 4 elements in each group, $\Theta_5 = (\vec{\pi_5}, \vec{\theta_5})$. Equation (7.7) does not use any committed data so it can be checked directly, thus $\Theta_6 = \phi$. Equation (7.8) is a pairing-product equation and the proof requires 4 elements in each group, $\Theta_7 = (\vec{\pi_7}, \vec{\theta_7})$. Equation (7.9)-(7.10) are linear and requires 2 elements of $\mathbb{G}_1$ each, $\Theta_8 = (\vec{\theta_8})$ and $\Theta_9 = (\vec{\theta_9})$. Equation (7.8)-(7.10) establish that the signer has produced the signature during the time interval t. Overall we need 30 group elements in $\mathbb{G}_1$, 24 in $\mathbb{G}_2$ and 1 element of $\mathbb{G}_T$. The axillary variable ρ_9 is needed to prove the equations in NIZK. Note that the signature is independent of number of attributes $|\zeta|$.

- `Verify`$(params, gpk, t, M, \sigma, \Upsilon, RL_t)$:
  It parses $\sigma = (\{\sigma_i\}_{i=1}^{3}, \{\rho\}_{i=4}^{7}, \sigma_8, \sigma_9, T_1, T_2, \Theta_1, \Theta_2, \Theta_4, \Theta_5, \{\Theta_i\}_{i=7}^{9})$ and returns 1 if and only if all proofs are valid and σ passes revocation test:

  (i) It verifies to see whether all the pairing equations hold according to Groth-Sahai proof system. Here we give the abstract construction of proof elements for clarity which is useful in the proof of anonymity. Following equations must satisfy with the proof elements $\Theta_1, \Theta_2, \Theta_4, \Theta_5, \Theta_7, \Theta_8, \Theta_9$

  (a) $F(\vec{\sigma_1}, \iota_2(\omega).\vec{\sigma_{3,2}}) = F(\iota_1(k_1), \iota_2(v_T)) \odot F(\vec{\sigma_{2,1}}, \iota_2(v_T)) \odot F(\vec{u_1}, \vec{\pi_{1,1}}) \odot F(\vec{u_2}, \vec{\pi_{1,2}}) \odot F(\vec{\theta_{1,1}}, \vec{v_1}) \odot F(\vec{\theta_{1,2}}, \vec{v_2})$

  (b) $E(\rho_4, \vec{\sigma_{2,2}}\iota_2(\rho_{5,2})) = E(g_1, \iota_2(g_2)) \odot E(\theta_{2,1}, \vec{v_1}) \odot E(\theta_{2,2}, \vec{v_2})$

  (c) $e(\rho_6, g_2) = e(k_1, \rho_{5,2}) \times e(\mathcal{F}(\mathcal{M}), \rho_7)$

  (d) $F(\vec{\sigma_{2,1}}, \vec{\varphi'}) = F(\vec{\varphi}, \vec{\sigma_{2,2}}) \odot F(\vec{u_1}, \vec{\pi_4}) \odot F(\vec{\theta_4}, \vec{v_1})$

  (e) $F(\vec{\sigma_{3,1}}, \iota_2(g_2)) = F(\iota_1(g_1), \vec{\sigma_{3,2}}) \odot F(\vec{u_1}, \vec{\pi_{5,1}}) \odot F(\vec{u_2}, \vec{\pi_{5,2}}) \odot F(\vec{\theta_{5,1}}, \vec{v_1}) \odot F(\vec{\theta_{5,2}}, \vec{v_2})$

  (f) $e(\rho_{5,1}, g_2) = e(g_1, \rho_{5,2})$

  (g) $F(\vec{\sigma_{3,1}}, \vec{\sigma_8}) = \iota_T(T_2) \odot F(\vec{u_1}, \vec{\pi_{7,1}}) \odot F(\vec{u_2}, \vec{\pi_{7,2}}) \odot F(\vec{\theta_{7,1}}, \vec{v_1}) \odot F(\vec{\theta_{7,2}}, \vec{v_2})$

  (h) $E(T_1, \vec{\sigma_9}) = E(g_1, \vec{\sigma_8}) \odot E(\theta_{8,1}, \vec{v_1}) \odot E(\theta_{8,2}, \vec{v_2})$

  (i) $E(g_1, \vec{\sigma_9}) = E(g_1, \iota_2(h_t)) \odot E(\theta_{9,1}, \vec{v_1}) \odot E(\theta_{9,2}, \vec{v_2})$

(ii) Revocation check: for all $B_{it} = h_t^{x_i} \in RL_t$,

$$T_2 \neq e(T_1, B_{it}) \tag{7.11}$$

- $\texttt{Open}(params, gpk, ok_{user}, t, \sigma, M, \Upsilon, r\vec{e}g)$: For the valid group signature the Opener just opens the commitment of $\rho_1 = A_i^{s_T}$ in σ_1, and outputs the corresponding identity i from the $r\vec{e}g$ with respect to A_i, if $e(\rho_1, g_2) \stackrel{?}{=} e(A_i, v_T)$, otherwise outputs $\perp$. If issuer and opener are the same entity then he can open the signature in O(1) time complexity, as he knows s_T value, he simply computes $A_i = \rho_1^{1/s_T}$.

7.4 Security Analysis

Theorem 7.4.1 *The proposed VLR-ABGS scheme is correct.*

Proof The correctness follows from the Groth-Sahai proof system. □

Theorem 7.4.2 *The proposed VLR-ABGS scheme preserves attribute anonymity under SXDH assumption.*

Proof According to the definition (7.2.3) it is sufficient to show that for any predicate Υ and for any subset of attributes $\mathcal{A}_i : \exists \zeta_1, \zeta_2 \subseteq \mathcal{A}_i$ that satisfies predicate i.e., $\Upsilon(\zeta_1) = \Upsilon(\zeta_2) = 1$, the output of $\texttt{Sign}(gpk, sk_i, t, \zeta_1, M, \Upsilon)$ is indistinguishable from the output of $\texttt{Sign}(gpk, sk_i, t, \zeta_2, M, \Upsilon)$, subject to the constraint that they pass the verification algorithm.
For any signature, $\sigma = (\{\sigma_i\}_{i=1}^{3}, \{\rho\}_{i=4}^{7}, \sigma_8, \sigma_9, T_1, T_2, \Theta_1, \Theta_2, \Theta_4, \Theta_5, \{\Theta_i\}_{i=7}^{9})$, the attribute certificates are hidden and used in σ_1 computation and it is easy to observe that for any two given group signatures of the same user U_i the uncommitted values $\rho_1 = A_i^{s_T}$ of both signature are indistinguishable among themselves, since both are identical because of same s_T value. Thus it will not reveal the underlying subset of attributes, but only it proves that it satisfies the predicate. Thus the proposed scheme preserves attribute anonymity unconditionally. □

Theorem 7.4.3 *If there exists an adversary $\mathcal{A}$ that can break the user anonymity of the scheme, then there exists an adversary $\mathcal{B}$ that can break either ℓ-DDHI problem in $\mathbb{G}_1$, DTDH problem or SXDH assumption. And we have*

$$Adv^{\text{BU}-\text{user}-\text{anon}} \leq 1/n.(2.Adv^{\text{SXDH}} + Adv^{\ell-\text{DDHI}}) + 1/(\mathbb{T}.n).(2.Adv^{\text{SXDH}} + Adv^{\text{DTDH}}) \tag{7.12}$$

where n is the maximal number of join queries, ℓ is the maximal number of signing queries for a user and $\mathbb{T}$ is the number of time periods.

Proof A part of the proof is similar to [24] and other part follows the approach of [82] under SXDH assumption. The proof is organized as a sequence of games such that adversary has no advantage in final game where as the first game is the real attack game as given in definition (7.2.4). Let S_i denote the event that the adversary wins in the game G_i with advantage $Adv_i = |Pr[S_i] - 1/2|$.

G_1: This is the real game as define in the definition (7.2.4). Challenger $\mathcal{B}$ sets up the scheme and defines the parameters as in the real scheme,

$$\begin{aligned} params &= (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, \mathcal{H}, \mathbb{T}, Att) \\ gpk &= (k_1, h_1, ..., h_{\mathbb{T}}, \omega, \mathcal{F}, \{g_{\text{att}_j}\}_{\text{att}_j \in Att}, \vec{u}, \vec{v}) \\ ik &= (\gamma, S) \\ ok_{user} &= (\alpha, \alpha') \end{aligned}$$

$\mathcal{B}$ gives gpk to $\mathcal{A}$. With this $\mathcal{B}$ answers all the queries made by adversary. At challenge phase $\mathcal{A}$ chooses 2 unrevoked and uncorrupted users U_{i_0}, U_{i_1} and is given a challenge signature σ^* on behalf of U_{i_κ}, $\kappa \in_R \{0,1\}$. In the output phase, adversary outputs her guess $\kappa' \in \{0,1\}$ and the advantage is $Adv_1 = |Pr[S_1] - 1/2|$.

G_2: Let n be the total number of passive join queries, joinP queries. In this game we modify the simulation G_1, $\mathcal{B}$ picks a challenge user id $i^\star$. In the challenge phase, $\mathcal{B}$ aborts if $i_\kappa \neq i^\star$. $\mathcal{B}$ also aborts if $i^\star$ is queried to $\mathsf{corrupt}$ or revoke oracle before or

during challenge phase period. The probability that $\mathcal{B}$ succeed in picking correct $i^\star$ is $1/n$. Therefore, $Adv_2 = Adv_1/n$.

$\mathsf{G_3}$: In this game, we modify the simulation of $\mathsf{G_2}$ where $\mathcal{B}$ randomly picks the challenge phase period $t^\star$. In the challenge phase, $\mathcal{B}$ aborts if $t^* \neq t^\star$. The probability that $\mathcal{B}$ picks correct $t^\star$ is $1/\mathbb{T}$. Therefore we can write $Adv_3 = Adv_2/\mathbb{T}$.

$\mathsf{G_{2a}}$: We modify the simulation of $\mathsf{G_2}$. $\mathcal{B}$ chooses $y \in_R \mathbb{Z}_p^*$ and define the $\ell-$DDHI like tuple $A = (g, g^y, ..., g^{y^\ell}) \in \mathbb{G}_1^{\ell+1}, D = g^{1/y} \in \mathbb{G}_1$. $\mathcal{B}$ chooses different random values $z^*, z_1, ..., z_{\ell-1} \in \mathbb{Z}_p^\ell$, and define the polynomial $f(X) = \Pi_{i=1}^{\ell-1}(X - z_i)$, of degree $\ell - 1$. From the above tuple, $\mathcal{B}$ can compute $g_1 = g^{f(X)}$. $\mathcal{B}$ chooses $\nu_t \in \mathbb{Z}_p^*$ and defines $h_t = g_2^{\nu_t}$ for $t \in \{1, ..., \mathbb{T}\}$. The future challenge user $i^\star$ will virtually have $y_{i^\star} = y - z^*$. $\mathcal{B}$ compute $g^{y_{i^\star}} = g_1^y / g_1^{z^*}$ from the above tuple. The membership certificate for the challenge user is $(g_1^{x_{i^\star}}, g_2^{x_{i^\star}}, g_1^y/g_1^{z^*}, A_{i^\star} = (k_1 g_1^y / g_1^{z^*})^{\beta/(\gamma + x_{i^\star})}, \{T_{i^\star,j} = A_{i^\star}^{s_j}\}_{att_j \in \mathcal{A}_{i^\star}})$. The setup is indistinguishable from $\mathsf{G_2}$, since all keys are having same distribution.
$\mathcal{B}$ answers all queries according to definition (7.2.4) and for the challenge user $U_{i^\star}$, the j-th signing queries, he computes $\rho_4 = g_1^{1/(y_{i^\star} + z_j)} = g^{\Pi_{i \neq j}(y + z_i)}$, that can be done from the defined tuple, the rest is done as in the real scheme using z_j as random. $\mathcal{B}$ can also answer any corruption query, that should not happen for the challenge user, even if we know y in this game.
For the challenge signing query, he does the same as above with the ephemeral value z^*, and the expected ID, $\rho_4 = g_1^{1/(y_{i^\star} + z^*)} = g^{f(y)/y} = g^{f'(y)} g^{\Pi(z_i)/y}$, where $f'(X) = (\Pi_{i=1}^{\ell}(X + z_i) - \Pi(z_i))/X$ is a polynomial of degree $\ell - 1$ and thus $g^{f'(y)}$ can be computable from the tuple. $\mathcal{B}$ thus compute $\rho_4 = g^{f'(y)}.D^{\Pi(z_i)}$ and returns the challenge signature σ^*. Therefore, $Adv_{2a} = Adv_2$.

$\mathsf{G_{2b}}$: We modify the game $\mathsf{G_{2a}}$. Here we initialize Groth-Sahai commitment keys in a perfectly hiding setting with the trapdoor, to allow the simulator to cheat in the proofs. Then all the proofs are simulated. This game is indistinguishable from the previous one under the SXDH. Thus $|Pr[S_{2b}] - Pr[S_{2a}]| = 2.Adv^{\text{SXDH}}$.

G_{2c}: In this game, we do not know anymore y, that we did not use anymore anyway, and thus this game is perfectly indistinguishable from previous one. Thus $Pr[S_{2c}] = Pr[S_{2b}]$.

G_{2d}: In this game, we replace the defined $\ell-$DDHI tuple with the actual $\ell-$DDHI challenge instance, where y is unknown to $\mathcal{B}$. Thus this game is indistinguishable from the previous one under the $\ell-$DDHI assumption. Therefore, $|Pr[S_{2d}] - Pr[S_{2c}]| \leq Adv^{\ell-\text{DDHI}}$.

G_{3a}: We modify the game G_3. In the setup phase we consider the group elements $Z_1 = g_1^{z_1}, Z_1' = g_2^{z_1}, Z_2 = g_1^{z_2}, Z_2' = g_2^{z_2}$ in constructing gpk and membership certificates. $\mathcal{B}$ chooses $\nu_t \in \mathbb{Z}_p^*$ and defines $h_t = g_2^{\nu_t}$ for $t \in \{1, ..., \mathbb{T}\} \backslash \{t^\star\}$, where as $h_{t^\star} = Z_2'$. $\mathcal{B}$ chooses $u \in_R \mathbb{Z}_p^*$, set $k_1 = g_1^u$. For the appropriate computed value s_T, corresponding to the predicate, $\mathcal{B}$ set $v_T = (Z_1'\omega)^{\beta s_T} = g_2^{s_T\beta(z_1+\gamma)}$ for a random $\beta \in \mathbb{Z}_p^*$ (so that implicitly we are setting $s_T = s_T\beta(z_1 + \gamma)$, therefore we assume that adversary wont invoke `BuildTree-Validity` algorithm to check the validity of public values of the predicate). Then all the join queries (except for $i = i^\star$) will answer by setting membership certificate $A_i = (Z_1^u Z_1^{y_i} k_1^\gamma g_1^{y_i\gamma})^{\beta/(\gamma+x_i)}$ and other values are same as define in the game G_3. For the target user $U_{i^\star}$, $\mathcal{B}$ set $sk_{i^\star} = (Z_1, Z_1', y_i, A_{i^\star} = (k_1 g_1^{y_i})^\beta, \{T_{i^\star,j} = A_{i^\star}^{s_j}\}_{att_j \in \mathcal{A}_{i^\star}})$, which implicitly defines $x_{i^\star} = z_1$. We note that, for periods $t \neq t^\star$, the revocation tokens $h_t^{x_{i^\star}}$ are computable as $Z_2'^{\nu_t}$. The revocation token $h_{t^\star}^{x_{i^\star}}$, for the period $t^\star$, for the user $U_{i^\star}$, is not computable and it is not needed, since $\mathcal{A}$ will not query for it unless the abortion rule of G_2 occurs. With this $\mathcal{B}$ can answer all the queries made by $\mathcal{A}$ even he does not explicitly use z_1, z_2, the discrete log of Z_1, Z_2 respectively. $\mathcal{A}$ will not notice the changes in the game. Therefore, $Pr[S_{3a}] = Pr[S_3]$.

G_{3b}: We modify the game G_{3a}. Here we initialize Groth-Sahai commitment keys in a perfectly hiding setting with the trapdoor, (t, t'). Then all the proofs are simulated. This game is indistinguishable from the previous one under the SXDH. Thus

$|Pr[S_{3b}] - Pr[S_{3a}]| = 2.Adv^{\mathrm{SXDH}}$.

$\mathsf{G_{3c}}$: In this game, we modify the generation of challenge signature σ^* and uses trapdoor (t, t') to simulate NIZK proofs. We assume that $\mathcal{B}$ known's the values $(Z_1 = g_1^{z_1}, Z_1' = g_2^{z_1}, Z_2 = g_1^{z_2}, Z_2' = g_2^{z_2}, Z_3 = g_1^{z_3}, \eta = g_2^{z_1 z_2 z_3})$, which is like DTDH tuple but with fix η. $\mathcal{B}$ use $Z_1, Z_1', Z_2 = g_1^{z_2}, Z_2'$ as in game $\mathsf{G_{3a}}$ and uses Z_3 to create the challenge signature. $\mathcal{B}$ implicitly defines $\delta = z_3$ by setting $T_1 = Z_3$ and $T_2 = e(g_1, \eta)$. $\mathcal{B}$ calculate the commitments $\sigma_1, \sigma_2, \sigma_3$ as specified by the scheme and similarly computes the proofs $(\Theta_1, \Theta_2, \Theta_4, \Theta_5)$. Here we calculate σ_8 as a commitment to $1_{\mathbb{G}_2}$: namely, $\sigma_8 = \vec{v_1}^{r_8} \odot \vec{v_2}^{s_8}$, where $r_8, s_8 \in_R \mathbb{Z}_p^*$. Then, $\mathcal{B}$ generates a proof $\Theta_7 = (\vec{\pi_7}, \vec{\theta_7})$, where

$$\vec{\pi_{7,1}} = \iota_2(\eta)^{-t}, \vec{\pi_{7,2}} = \iota_2(\eta), \vec{\theta_{7,1}} = \sigma_{3,1}^{r_8}, \vec{\theta_{7,2}} = \sigma_{3,1}^{s_8}.$$

Note that for generating this proof the value $\rho_8 = h_{t^*}^{\delta} = g_2^{z_2 z_3}$ is not used instead it takes the advantage of η. The proof Θ_8 is generated as the real proof using the variable assignment $\rho_8 = \rho_9 = 1_{\mathbb{G}_2}$ that satisfies the relation $e(T_1, \rho_9) = e(g_1, \rho_8)$ and the committed value $\sigma_9 = \vec{v_1}^{r_9} \odot \vec{v_2}^{s_9}$, a commitment to $1_{\mathbb{G}_2}$. The $\Theta_9 = (\vec{\theta_9})$ is computed as,

$$\theta_{9,1} = g_1^{r_9}.Z_2^{-t'}, \theta_{9,2} = g_1^{s_9}.Z_2.$$

This satisfy the last verification equation

$$E(g_1, \vec{\sigma_9}) = E(g_1, \iota_2(h_t)) \odot E(\theta_{9,1}, \vec{v_1}) \odot E(\theta_{9,2}, \vec{v_2})$$

since $\vec{v_2} = (g_2^{t'}, g_2^{\alpha' t' - 1})$. As in [68] the simulated proofs are randomized, uniform in the space of valid proofs, and achieve perfect witness indistinguishability. This game is perfectly indistinguishable from the previous game and $Pr[S_{3c}] = Pr[S_{3b}]$.

$\mathsf{G_{3d}}$: This is same as previous game but η is random from $\mathbb{G}_2$. This modification is not noticeable to $\mathcal{A}$ under DTDH assumption. Thus $|Pr[S_{3d}] - Pr[S_{3c}]| = 2.Adv^{\mathrm{DTDH}}$. Also it is easy to observe that $Pr[S_{3d}] = 1/2$. The elements T_1 and T_2 are completely independent of $x_{i^*} = z_1$ (and thus of U_{i^*}). In game $\mathsf{G_{3d}}$ under WI setting, the values

$\{\sigma_i\}_{i=1}^{3}, \{\rho\}_{i=5}^{7}, \sigma_8, \sigma_9$ and the proofs $\Theta_1, \Theta_2, \Theta_4, \Theta_5, \{\Theta_i\}_{i=7}^{9}$ reveals no information of the user U_{i^*}.

When we combine all the probabilities we obtain the upper bound (7.12) on $\mathcal{A}$'s advantage in game $\mathsf{G_1}$.

Theorem 7.4.4 *The proposed scheme preserves the attribute unforgeability under KEA1 and DL assumptions.*

Proof Lemma 7.4.5 implies the Theorem 7.4.4. □

Lemma 7.4.5 *Under the DL and KEA1 assumptions there exists no PPT adversary $\mathcal{A}$ which passes verification with forged attributes with non negligible probability.*

Proof The input to the simulator $\mathcal{B}$ is an instance of the DL problem, $(g, g') \in \mathbb{G}_1^2$. Let $\xi = log_g g'$.
Setup: According to the VLR-ABGS scheme setup $\mathcal{B}$ generates the system parameters, *params*. $\mathcal{B}$ sets $g_1 = g$ and $k_1 = g'$, and generate the remaining parameters, gpk, ik, ok_{user}. $\mathcal{B}$ gives gpk and ok_{user} to $\mathcal{A}$.
Queries: As $\mathcal{B}$ knows all the keys, it can answer all the queries generated by an adversary $\mathcal{A}$ according to the definition of attribute unforgeability .
Output: Finally, $\mathcal{A}$ outputs a signature σ^* with forged attribute certificates on message M^*, a predicate Υ^* whose public values are $\mathcal{T}_{\Upsilon^*} = (\{s_{d_j}\}_{d_j \in D_{T_{\Upsilon^*}}}, v_T = g_2^{s_T}, T^{\text{ext}})$, and signer's secret key sk_{i^*} such that $\Upsilon(\mathcal{A}_{i^*}) \neq 1$ and $\Upsilon(\mathcal{A}_{i^*} \cup att_j) = 1$. As it is a valid signature which passes verification algorithm and from (7.2) $\rho_1^* = A_{i^*}^{\frac{s_T}{\gamma + x_{i^*}}}$. This can be viewed as $\rho_1^* = (k_1 g_1^{y_{i^*}})^{\frac{s_T}{\gamma + x_{i^*}}} = k_1^{\frac{s_T}{\gamma + x_{i^*}}} X$ and $k_1^{s_T}$ can be extracted by raising the power $\gamma + x_{i^*}$, where $X = g_1^{\frac{y_{i^*} s_T}{\gamma + x_{i^*}}}$. Note that $k_1^{s_T}$ is unknown to $\mathcal{A}$ but she is producing it in signature.
It is like $\mathcal{B}$ is giving input $(g_1 = g, g_1^{s_T} = g_{att_j}^{\Delta_{att_j}} \times g_1^{s_{T_2}} = g^{s_T})$ to $\mathcal{A}$ and $\mathcal{A}$ implicitly returns $(k_1 = g', k_1^{s_T} = g'^{s_T})$. Then by KEA1 assumption, $\mathcal{B}$ can utilize the extractor $\bar{\mathcal{A}}$ to extract a value ξ. Under DL assumption it can be done with negligible probability.

Thus the signature produced by the forged attribute certificates can pass verification with negligible probability.
To be more particular, we can assume that the $\mathcal{A}$ is missing one attribute certificate, say $T_{i^*,j} = A_{i^*}^{s_j} = (k_1 g_1^{y_{i^*}})^{\frac{s_j}{\gamma + x_{i^*}}}$, to satisfy the predicate, but he knows $g_{att_j} = g_1^{s_j}$. And $\mathcal{A}$ is producing it in forged group signature σ^*. Then similar to above from KEA1 and DL assumptions it is negligible to produce such signatures. □

Theorem 7.4.6 *The proposed scheme preserves the collusion resistance of attribute.*

Proof Lemma 7.4.7 implies the Theorem 7.4.6. □

Lemma 7.4.7 *Even if some malicious participants* $\mathrm{U}_{i_1}, ..., \mathrm{U}_{i_k} (k > 1)$ *with the set of attributes* $\zeta_{i_1}, ..., \zeta_{i_k}$ *collude, they cannot make a valid signature associated with a predicate* Υ, *where* $(\cup_{j=1}^{k} \Upsilon(\zeta_{i_j}) = 1)$ *and* $\Upsilon(\zeta_{i_j}) \neq 1$ *for* $j = 1, ..., k$ *with non-negligible probability.*

Proof Without loss of generality, we assume that U_{i_1} with ζ_{i_1} and U_{i_2} with ζ_{i_2} represent malicious participants. U_{i_1} and U_{i_2} attempt to make a valid signature associated with Υ which satisfies $\Upsilon(\zeta_{i_1} \cup \zeta_{i_2}) = 1, \Upsilon(\zeta_{i_1}) \neq 1$ and $\Upsilon(\zeta_{i_2}) \neq 1$. They can satisfy the relations (7.2) because they have a valid membership certificate $(A_{i_1}, X_{i_1}, y_{i_1})$. We assume that $T_{i_1,j}^t = T_{i_2,j}$, where $t \in \mathbb{Z}_p^*$. Note that the probability of $t = 1$ is negligible. And they tries to compute

$$\rho_1 = \Pi_{\mathrm{att}_j \in \mathcal{A}_{\mathrm{i}_1}} T_{i_1,j}^{\Delta_{\mathrm{att}_j}} \times \Pi_{\mathrm{att}_j \in \mathcal{A}_{\mathrm{i}_2}} T_{i_2,j}^{\Delta_{\mathrm{att}_j}} \times A_{i_1}^{s_{T_2}}$$

$$= A_{i_1}^{(\Sigma_{\mathrm{att}_j \in \mathcal{A}_{\mathrm{i}_1}} \Delta_{\mathrm{att}_j} s_j + t\Sigma_{\mathrm{att}_j \in \mathcal{A}_{\mathrm{i}_2}} \Delta_{\mathrm{att}_j} s_j + s_{T_2})}$$

Then from (2.6)

$$\Sigma_{\mathrm{att}_j \in \mathcal{A}_{\mathrm{i}_1}} \Delta_{\mathrm{att}_j} s_j + t\Sigma_{\mathrm{att}_j \in \mathcal{A}_{\mathrm{i}_2}} \Delta_{\mathrm{att}_j} s_j + s_{T_2} \neq s_T$$

holds. But $t \neq 1$ means that they cannot collude. □

Theorem 7.4.8 *If there exists an adversary $\mathcal{A}$ that breaks the traceability of the scheme, then we can build an adversary $\mathcal{B}$ that can break the q-HHSDH assumption, where q is the maximal number of users.*

Proof Since the membership certificate format is similar to the one proposed in [24; 48], the proof directly reduces to the q-HHSDH assumption. The simulator $\mathcal{B}$ receives q-HHSDH challenge $(g_1, k_1, g_2, \omega = g_2^{\gamma}, (g_1^{x_i}, g_2^{x_i}, y_i, A_i = (k_1 g_1^{y_i})^{1/(\gamma+x_i)})_{i\in[1,q]})$ and tries to solve it, from $\mathcal{A}$ that breaks the traceability of our scheme.
Setup: $\mathcal{B}$ generates the commitment keys, attribute secret and public values, and other parameters as in the ABGS scheme by using the q-HHSDH challenge values. $\mathcal{B}$ gives gpk and ok_{user} to $\mathcal{A}$.
Queries: To answer the i-th join queries, if this is an active join, $\mathcal{B}$ extracts y_i' chooses his y_i'' so that $y_i' + y_i'' = y_i$, if it is a passive join, $\mathcal{B}$ directly chooses y_i. Thus $\mathcal{B}$ can answer all the queries according to traceability definition.
Output: After atmost q join queries, $\mathcal{A}$ outputs a new signature with a new certificate tuple with non-negligible probability. As $\mathcal{B}$ knows the trapdoor of the commitment scheme, he can obtain $(g_1^x, g_2^x, g_1^y, g_2^y, A = (k_1 g_1^y)^{1/(\gamma+x)})$. Thus $\mathcal{B}$ answers the challenge q-HHSDH instance with the same advantage of $\mathcal{A}$. □

Theorem 7.4.9 *If there exists an adversary $\mathcal{A}$ that breaks the non-frameability of the scheme, then we can build an adversary $\mathcal{B}$ that can either break the q-HSDH or the CDH$^+$ computational problems, or the 1-DDHI or the SXDH decisional problems, where q is the maximal number of signing queries for a user.*

Proof The proof is similar to the proof of non-frameability in the Blazy and Pointcheval [24] traceable signature. There exists two types of adversary, one breaks the non-frameability by forging the new ID, ρ_4, on an uncorrupted user and another breaks the non-frameability by reusing an existing ID with the corresponding certificate but on a new message. With 1/2 probability $\mathcal{B}$ decides which type of adversary it is.
Type 1: The simulator $\mathcal{B}$ receives q-HSDH challenge $((g_1, g_2, g_1^y, g_2^y), (g_1^{t_i}, g_2^{t_i}, g_1^{1/(y+t_i)})_{i\in[1,q]})$ and tries to solve it, from an adversary $\mathcal{A}$ that breaks the non-frameability of our scheme by forging a new ID, ρ_4, on an uncorrupted user.

Setup: $\mathcal{B}$ generates the gpk, ik and ok_{user} as the real settings and gives it to $\mathcal{A}$. $\mathcal{B}$ selects the target user on which he expects the attack and sets his membership certificate corresponding to one with y as a secret key.

Queries: $\mathcal{B}$ can answer any joinP query as he knows ik and can answer corrupt query on any user except the target user, otherwise the simulation fails. $\mathcal{B}$ can answer the sign queries and can answer to atmost q sign queries for the target user with the help of challenge q-HSDH tuple.

Output: After all the queries and atmost q signing queries for target user, $\mathcal{A}$ succeeds in breaking the non-frameability with non-negligible probability by generating a new tuple $(\rho_4 = g_1^{1/(y+t)}, \rho_5 = (g_1^t, g_2^t))$, on an uncorrupted user. Thus $\mathcal{B}$ solves the q-HSDH challenge with non-negligible probability.

Type 2: The simulator $\mathcal{B}$ is given an asymmetric Waters public key $(pk = (g_1^{\xi}, g_2^{\xi})$ for the global parameters $(g_1, g_2, k_1, \mathcal{F}))$. $\mathcal{B}$ tries to break this signature, and thus the CDH$^+$ problem, from an adversary $\mathcal{A}$ breaking the non-frameability of our scheme by reusing an existing tuple ρ_4, ρ_5 on a new message.
In the first game, G_1, $\mathcal{B}$ knows the discrete logarithm value ξ, generates a new ik, ok_{user} and gives ik, ok_{user} to $\mathcal{A}$ together with the public key $gpk = (k_1, \omega, \mathcal{F},$ $\{g_{\mathrm{att}_j}\}_{\mathrm{att}_j \in Att})$. $\mathcal{B}$ can answer any joinP query as he knows ik and extract the secret keys from the extraction key of the commitment scheme, one of those uncorrupted user is expected to be a challenge user, with the secret key y, the one $\mathcal{A}$ has to frame. $\mathcal{B}$ can answer any signing queries. On one of them for the challenge user, say on M, he will use the above ξ as ephemeral Waters public key (for the z), and thus computes a $\rho_4 = g_1^{1/(y+\xi)}$ with the corresponding Groth-Sahai proof. This way $\mathcal{A}$ possesses a valid signature on M, $\mathcal{M} = \mathcal{H}(t||M)$, with $\rho_5 = (g_1^{\xi}, g_2^{\xi}), \rho_6 = k_1^{\xi}\mathcal{F}(\mathcal{M})^s, \rho_7 = g_2^s$. With non-negligible probability $\mathcal{A}$ breaks the non-frameability of our scheme, by hypothesis $\mathcal{A}$ does it by reusing an existing $\rho_1, ..., \rho_5$, as uncorrupted users are indistinguishable, $\mathcal{A}$ frames our challenge user with non-negligible probability, and as the signing queries are finite, he will use $\rho_5 = (g_1^{\xi}, g_2^{\xi})$ with non-negligible probability.

Therefore, with non-negligible probability at some period t^* $\mathcal{A}$ outputs a new valid signature on $\mathcal{M}' = \mathcal{H}(t||M')$ with $\rho_5 = (g_1^{\xi}, g_2^{\xi})$, this means we have (ρ_5, ρ_6, ρ_7) such that $e(\rho_{5,1}, g_2) = e(g_1, \rho_{5,2}), e(\rho_6, g_2) = e(k_1, \rho_{5,2}) \times e(\mathcal{F}(\mathcal{M}'), \rho_7)$, and thus $\mathcal{B}$ can output a valid forgery on the Waters challenge for the public key (g_1^{ξ}, g_2^{ξ}). But in this game, we know ξ.
In a second game, $\mathsf{G_2}$, the Groth-Sahai setup is used as hiding one, so that the proofs can be simulated, and namely without using ξ. This is indistinguishable from the previous game under the SXDH assumption.
In the third game, $\mathsf{G_3}$, replace ρ_4 by a random value, still simulating the proofs. A random ρ_4 is indistinguishable from the real one under the DDHI problem as seen in user anonymity proof. Furthermore, here there is only one elements, hence the $1-$DDHI assumption. In the last game, one does not need to know ξ anymore, and thus the signature forgery reduces to breaking the asymmetric CDH$^+$. □

7.5 Comparison

In the construction, the group signature contains 30 elements from $\mathbb{G}_1$, 24 elements from $\mathbb{G}_2$ and 1 element of $\mathbb{G}_T$. Let $\Phi = |\zeta|$, where ζ is the set of attributes associated with a signature and $m = |Att|$. Let $\hat{m}$ be the average number of attributes assigned to any user and m' the size of the message. RO means Random oracle model, e represents the paring operation and r represents the number of revoked members. In Table 7.1, we compare the efficiency of our scheme with other schemes. Note that the verification cost of the proposed scheme is independent of the number of attributes, where as in other schemes the verification cost is linear in terms of number of attributes. From the table it can be noticed that non-frameability is not achieved by standard model combined scheme of Herranz et al. [70] and Libert et al. [82]. Further, the combined scheme has verification cost that is not independent of the number of attributes and all the key lengths are large.

Table 7.1: Comparison of VLR-ABGS scheme in standard model with other schemes

	Khader [72]	Emura et al. [52]	Herranz et al. [70] +Libert et al. [82]	Our Scheme
Attribute anonymity	no	no	yes	yes
VLR	yes	no	yes	yes
Backward Unlinkability	no	no	yes	yes
Non-frameability	no	yes	no	yes
Attribute revocation	yes	no	no	no
Predicate	monotone	monotone	threshold	monotone
Signature Length	$\mathrm{O}(\Phi)$	$\mathrm{O}(\Phi)$	$\begin{pmatrix}(15\lvert\mathbb{G}_p\rvert)+\\(46\lvert\mathbb{G}_p\rvert+1\lvert\mathbb{G}_T\rvert)\end{pmatrix}=\mathrm{O}(1)$	$30\lvert\mathbb{G}_1\rvert+24\lvert\mathbb{G}_2\rvert+1\lvert\mathbb{G}_T\rvert=\mathrm{O}(1)$
User's Secret Key Length	$(\hat{m}+1)\lvert\mathbb{G}_1\rvert+\lvert\mathbb{Z}_p^*\rvert$	$(\hat{m}+1)\lvert\mathbb{G}_1\rvert+2\lvert\mathbb{Z}_p^*\rvert$	$((\hat{m}+m-1)\lvert\mathbb{G}_p\rvert)+(3\lvert\mathbb{G}_p\rvert)$	$(\Phi+1)\lvert\mathbb{G}_1\rvert+2\lvert\mathbb{Z}_p^*\rvert$
Assumption	DLIN, q-SDH	DL,DDH, q-SDH	$\begin{matrix}\text{DLin},(\ell,m,t)\text{-aMSE-CDH},\\ \ell\text{-HSDH,DTDH}\end{matrix}$	$\begin{matrix}\text{DL},\ell\text{-DDH},q\text{-HHSDH},\text{SXDH},\\ \text{DTDH},\text{CDH}^+,\text{KEA1}\end{matrix}$
Model	RO	RO	standard	standard
Signing cost	$\begin{pmatrix}(7+2\Phi)\mathbb{G}_1+(5+\Phi)\mathbb{G}_T\\+(\Phi+1)e\end{pmatrix}$	$\begin{pmatrix}(9+3\Phi)\mathbb{G}_1+(1+\Phi)\mathbb{G}_2\\+8\mathbb{G}_T+3e\end{pmatrix}$	$\begin{matrix}((6m+6m'+\Phi(\Phi-1)+14)\mathbb{G}_p)\\+((m'+440)\mathbb{G}_p)\end{matrix}=\mathrm{O}(\Phi^2)+\mathrm{O}(m)$	$\mathrm{O}(\Phi)$
Verification cost	$\begin{pmatrix}(6+2r)\mathbb{G}_1+(8+2\Phi)\mathbb{G}_T\\+(\Phi+2r+1)e\end{pmatrix}$	$\begin{pmatrix}(11+2\Phi)\mathbb{G}_1+(\Phi+1)\mathbb{G}_2\\+14\mathbb{G}_T+6e\end{pmatrix}$	$\begin{matrix}((4m+6m')\mathbb{G}_p+21\mathbb{G}_T+33e)\\+((m'+3)\mathbb{G}_p+48\mathbb{G}_T+(75+r)e)\end{matrix}=\mathrm{O}(r)+\mathrm{O}(m)$	$\mathrm{O}(r)$

7.6 Summary

In this chapter, we have proposed a VLR-ABGS scheme which achieves attribute anonymity and backward unlinkability with constant signature length and proven that it is secure under the standard model. Our scheme is dynamic with respect to both users and attributes i.e. anytime a user can join and attributes can be added without regenerating the keys. We note that our scheme is efficient compared to the other ABGS schemes in terms of verification cost and signature length. We also note that our scheme has non-frameability as an added feature compared to the recently proposed VLR-GS scheme by Libert et al. [82]. Further, our scheme can also be used as VLR-GS scheme. In the next chapter, we conclude our thesis.

Chapter 8

Conclusion

In this thesis we addressed all the research questions listed in Section 1.2 except the last one, i.e. attribute revocation. In chapter 2, we formalized the definition of ABGS scheme with the new security definitions viz. attribute anonymity and attribute unforgeability. In chapter 3, we proposed an ABGS scheme with attribute anonymity and proven that it is secure under random oracle model. We also gave a revocation mechanism for the scheme. The signature size is around 15 group elements. In chapter 4, we proposed an ABGS scheme with VLR feature having attribute anonymity and backward unlinkability. We proved that the scheme is secure under random oracle model. The signature size is around 10 group elements. In chapter 5, we proposed two ABGS schemes with attribute anonymity having short signature length. We proved that the schemes are secure under the standard model but does not preserve non-frameability. The signature size of the first construction is 6 group elements, and for the second construction is 5 group elements. In chapter 6, we proposed an ABGS scheme which preserves non-frameability and proven that it is secure under the standard model. The signature size is around 19 group elements. In chapter 7, we proposed an ABGS scheme with VLR feature also which preserves non-frameability and proven that it is secure under standard model. The signature size is around 21 group elements.

Our schemes are dynamic with respect to the users and attributes, i.e. anytime a user can join and attributes can be added without reissuing the keys. We compared

the proposed schemes with the other schemes in the literature and found that the schemes are efficient in terms of verification cost, user's secret key length, constant signature length and signing cost with extra added features. This is tabulated in Table 8.1. We observe that the schemes with random oracle model are efficient than the schemes with standard model. All the proposed schemes can be used as group signature schemes under special setting. The VLR-ABGS scheme preserves non-frameability in contrast to VLR-GS scheme of Libert et al.. All the ABGS schemes without VLR feature can be used as ABS scheme with an extra feature of signer tracing. Our ABGS schemes supports monotone predicates in contrast to the short ABS scheme of Herranz et al. which supports threshold predicates and has the signature size of 15 group elements. Moreover the verification cost in our scheme is constant where as the verification cost of Herranz et al.'s ABS scheme is linear in terms of the number of attributes. In the proposed ABGS schemes, signer has to contact group manager if the predicate public values are not present in the public repository whereas in ABS scheme no such communication is required.

We use Boneh's membership certificate format [28], $A_i = (g_1 F_i)^{1/(\gamma+x_i)}$, where the user's and group manager's secrets, x_i and γ respectively, appear in the exponent of A_i in the form $\frac{1}{\gamma+x_i}$. This allows us to revoke the membership certificate using the method given in [48]. But in attribute certificate the user's attribute secret, s_j, does not appear in the required form. Therefore, we cannot use the method given in [48]. We have also tried to include user's attribute secret in the attribute certificate as a third component in the exponent of A_i as $\frac{1}{\gamma+x_i+s_j}$, but still we cannot use the method given in [48]. Further, we have tried to have attribute certificate without binding with the group manager's secret, $\frac{1}{x_i+s_j}$, but we cannot use Lagrange's interpolation property to achieve attribute anonymity. Therefore, we leave attribute revocation for future work.

It is challenging to device an `Build-Tree` algorithm which runs without group secret key. This allows the group manager to remain off-line.

Table 8.1: Comparison of all the proposed schemes with other schemes

	UA	AA	VLR	BU	Nf	AR	AT	Pred	Model	Signature length	User′s Secret Key Length	Assumption	Signing cost	Verf cost
Khader [72]	CPA	no	yes	no	no	yes	no	monotone	RO	$O(\Phi)$	$(\hat{m}+1)\|\mathbb{G}_1\|+\|\mathbb{Z}_p^*\|$	DLIN, q-SDH	$O(\Phi)$	$O(\Phi+r)$
Emura et al. [52]	CCA2	no	no	no	yes	no	no	monotone	RO	$O(\Phi)$	$(\hat{m}+1)\|\mathbb{G}_1\|+2\|\mathbb{Z}_p^*\|$	DL,DDH, q-SDH	$O(\Phi)$	$O(\Phi)$
Scheme 1 [2]	CCA2	yes	no	no	yes	no	yes	monotone	RO	$4\|\mathbb{G}_1\|+3\|\mathbb{G}_2\|+5\|\mathbb{Z}_p^*\|=O(1)$	$\|\mathbb{G}_1\|+\|\mathbb{Z}_p^*\|$	DDH,DLDH, q-SDH, DL	$O(\Phi)$	$O(1)$
Scheme 2	CPA	yes	yes	yes	yes	no	no	monotone	RO	$4\|\mathbb{G}_p\|+6\|\mathbb{Z}_p^*\|=O(1)$	$\|\mathbb{G}_1\|+2\|\mathbb{Z}_p^*\|$	DL,$q-$SDH, DLIN, KEA1	$O(\Phi)$	$O(r)$
Herranz et al. [70] + Boyen et al. [34]	CPA	yes	no	no	no	no	no	threshold	Std	$21\|\mathbb{G}_p\|=O(1)$	$(m+\hat{m}+3)\|\mathbb{G}_p\|$	DLin,$(\ell,m,t)-$aMSE$-$CDH SGD, $\ell-$HSDH,	$O(\Phi^2+m)$	$O(m)$
Scheme 3 [4]	CPA	yes	no	no	no	no	no	monotone	Std	$6\|\mathbb{G}_p\|=O(1)$	$(3+\hat{m})\|\mathbb{G}\|$	SGD, $\ell-$HSDH, DL, KEA	$O(\Phi)$	$O(1)$
Herranz et al. [70] + Liang et al. [79]	CPA	yes	no	no	no	no	no	threshold	Std	$15\|\mathbb{G}_p\|+5\|\mathbb{G}_n\|=O(1)$	$(m+m')\|\mathbb{G}_p\|+\|\mathbb{G}_n\|+\|\mathbb{Z}_p\|$	DLin ,$(\ell,m,t)-$aMSE$-$CDH SGD, $\ell'-$MOMSDH	$O(\Phi^2+m)$	$O(m)$
Scheme 4 [3]	CPA	yes	no	no	no	no	no	monotone	Std	$5\|\mathbb{G}_n\|=O(1)$	$(m'+1)\|\mathbb{G}_n\|+\|\mathbb{Z}_n^*\|$	SGD, $\ell-$SDH, $\ell'-$MOMSDH, DL, KEA	$O(\Phi)$	$O(1)$
Scheme 5 [61]	CCA2	yes	no	no	yes	no	yes	monotone	Std	$29\|\mathbb{G}_1\|+20\|\mathbb{G}_2\|=O(1)$	$(2+\hat{m})\|\mathbb{G}_1\|+\|\mathbb{G}_2\|+\|\mathbb{Z}_p^*\|$	DL,$q-$HSDH, SXDH,$q-$HHSDH, $\ell-$DDHI, ,CDH$^+$, KEA	$O(\Phi)$	$O(1)$
Herranz et al. [70] + Libert et al. [82]	CCA2	yes	yes	yes	no	no	no	threshold	Std	$(15\|\mathbb{G}_p\|)+(46\|\mathbb{G}_p\|+1\|\mathbb{G}_T\|))=O(1)$	$((\hat{m}+m-1)\|\mathbb{G}_p\|)+3\|\mathbb{G}_p\|$	DLin,$(\ell,m,t)-$aMSE$-$CDH, $\ell-$HSDH,DTDH	$O(\Phi^2+m)$	$O(r+m)$
Scheme 6 [5]	CCA2	yes	yes	yes	yes	no	no	monotone	Std	$30\|\mathbb{G}_1\|+24\|\mathbb{G}_2\|+1\|\mathbb{G}_T\|=O(1)$	$(\Phi+1)\|\mathbb{G}_1\|+2\|\mathbb{Z}_p^*\|$	DL,$\ell-$DDHI,$q-$HHSDH, SXDH, DTDH,CDH$^+$,KEA1	$O(\Phi)$	$O(r)$

Notations and Abbreviations	
$\Phi=\|\zeta\|$	Number of attributes associated with a signature.
$m=\|Att\|$	Size of the universal set of attributes.
$\hat{m}$	Average number of attributes assigned to any user.
m'	Size of the message.
RO	Random oracle model.
Std	Standard model.
e	Paring operation.
r	Number of revoked members.
UA	User Anonymity
AA	Attribute Anonymity
VLR	Verifier-Local Revocation
BU	Backward Unlinkability
Nf	Non-frameability
AR	Attribute Revocation
AT	Attribute Tracing
Pred	Predicate

Bibliography

[1] Michel Abdalla, Céline Chevalier, and David Pointcheval. Smooth projective hashing for conditionally extractable commitments. In Shai Halevi, editor, *CRYPTO*, volume 5677 of *Lecture Notes in Computer Science*, pages 671–689. Springer, 2009. 2.4.5.1

[2] Syed Taqi Ali and B. B. Amberker. A dynamic constant size attribute-based group signature scheme with attribute anonymity. *IJIPSI*, 1(4):312–343, 2013. 1.3.1, 3.1, 8.1

[3] Syed Taqi Ali and B B Amberker. Short Attribute-Based group signature without random oracles with attribute anonymity. In *International Symposium on Security in Computing and Communications (SSCC-2013)*, Mysore, India, August 2013. 1.3.3, 5.1, 5.4, 8.1

[4] Syed Taqi Ali and B. B. Amberker. Attribute-based group signature without random oracles with attribute anonymity. *Int. J. Inf. Comput. Secur.*, 6(2):109–132, October 2014. 1.3.3, 5.1, 5.3, 8.1

[5] Syed Taqi Ali and B. B. Amberker. Dynamic attribute-based group signature with verifier-local revocation and backward unlinkability in the standard model. *IJACT*, 3(2):148–165, 2014. 1.3.5, 7.1, 8.1

[6] Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. A practical and provably secure coalition-resistant group signature scheme. In Mihir Bellare, editor, *CRYPTO*, volume 1880 of *Lecture Notes in Computer Science*, pages 255–270. Springer, 2000. 1

[7] Giuseppe Ateniese and Breno de Medeiros. Efficient group signatures without trapdoors. In Chi-Sung Laih, editor, *ASIACRYPT*, volume 2894 of *Lecture Notes in Computer Science*, pages 246–268. Springer, 2003. 1

[8] Giuseppe Ateniese, Dawn Xiaodong Song, and Gene Tsudik. Quasi-efficient revocation in group signatures. In Matt Blaze, editor, *Financial Cryptography*,

volume 2357 of *Lecture Notes in Computer Science*, pages 183–197. Springer, 2002. 1, 2.4.8.1

[9] Giuseppe Ateniese and Gene Tsudik. Some open issues and new directions in group signatures. In Matthew K. Franklin, editor, *Financial Cryptography*, volume 1648 of *Lecture Notes in Computer Science*, pages 196–211. Springer, 1999. 1

[10] Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick Drew McDaniel, editors. *Proceedings of the 11th ACM Conference on Computer and Communications Security, CCS 2004, Washington, DC, USA, October 25-29, 2004*. ACM, 2004. 8

[11] László Babai. Trading group theory for randomness. In Robert Sedgewick, editor, *STOC*, pages 421–429. ACM, 1985. 2.4.3

[12] László Babai and Shlomo Moran. Arthur-merlin games: a randomized proof system, and a hierarchy of complexity class. *J. Comput. Syst. Sci.*, 36(2):254–276, April 1988. 2.4.3

[13] Paulo S. L. M. Barreto, Steven D. Galbraith, Colm O'Eigeartaigh, and Michael Scott. Efficient pairing computation on supersingular abelian varieties. *Des. Codes Cryptography*, 42(3):239–271, 2007. 2.2

[14] Paulo S. L. M. Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In Bart Preneel and Stafford E. Tavares, editors, *Selected Areas in Cryptography*, volume 3897 of *Lecture Notes in Computer Science*, pages 319–331. Springer, 2005. 2.2

[15] Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In Ernest F. Brickell, editor, *CRYPTO*, volume 740 of *Lecture Notes in Computer Science*, pages 390–420. Springer, 1992. 2.4.3.1

[16] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In Eli Biham, editor, *EUROCRYPT*, volume 2656 of *Lecture Notes in Computer Science*, pages 614–629. Springer, 2003. 1, 2.4.8

[17] Mihir Bellare and Adriana Palacio. The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In Franklin [57], pages 273–289. 2.3.15

[18] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi

Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM Conference on Computer and Communications Security*, pages 62–73. ACM, 1993. 2.4.4, 2.5.3

[19] Mihir Bellare and Phillip Rogaway. Code-based game-playing proofs and the security of triple encryption. *IACR Cryptology ePrint Archive*, 2004:331, 2004. 2.5.1

[20] Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of group signatures: The case of dynamic groups. In Alfred Menezes, editor, *CT-RSA*, volume 3376 of *Lecture Notes in Computer Science*, pages 136–153. Springer, 2005. 1, 2.4.8, 3.2, 3.2, 1, 2, 6.2, 7.2

[21] Ian Blake, Gadiel Seroussi, and Nigel Smart. Elliptic curves in cryptography london mathematical society, lecture note series 265, 1999. 2.2

[22] Ian Blake, Gadiel Seroussi, Nigel Smart, and JWS Cassels. *Advances in Elliptic Curve Cryptography (London Mathematical Society Lecture Note Series)*. Cambridge University Press, 2005. 2.2

[23] Olivier Blazy, Georg Fuchsbauer, David Pointcheval, and Damien Vergnaud. Signatures on randomizable ciphertexts. In Catalano et al. [41], pages 403–422. 2.3.13, 2.4.15

[24] Olivier Blazy and David Pointcheval. Traceable signature with stepping capabilities. In David Naccache, editor, *Cryptography and Security*, volume 6805 of *Lecture Notes in Computer Science*, pages 108–131. Springer, 2012. 1.3.4, 1.3.5, 2.3.12, 2.4.8, 2.5.1, 6.2, 6.3, 6.3, 6.4, 6.4, 6.4, 7.2, 7.3, 7.3, 7.4, 7.4, 7.4

[25] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In Janos Simon, editor, *STOC*, pages 103–112. ACM, 1988. 2.4.6

[26] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin, Jan Camenisch, and Jan Camenisch, editors, *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 56–73. Springer, 2004. 2.4.13, 2.4.14, 4.5

[27] Dan Boneh and Xavier Boyen. Short signatures without random oracles and the sdh assumption in bilinear groups. *J. Cryptology*, 21(2):149–177, 2008. 2.3.5, 2.5.4

[28] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Franklin [57], pages 41–55. 1, 1.2, 2.3.3, 2.3.4, 2.3.14, 2.4.12, 3.3, 3.3, 4.5, 8

[29] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In Joe Kilian, editor, *TCC*, volume 3378 of *Lecture Notes in Computer Science*, pages 325–341. Springer, 2005. 2.3.6

[30] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. *J. Cryptology*, 17(4):297–319, 2004. 3.6

[31] Dan Boneh and Hovav Shacham. Group signatures with verifier-local revocation. In Atluri et al. [10], pages 168–177. 1, 2.4.8.1, 4.1, 4.2, 4.2, 7.2

[32] Xavier Boyen. The uber-assumption family. In Steven D. Galbraith and Kenneth G. Paterson, editors, *Pairing*, volume 5209 of *Lecture Notes in Computer Science*, pages 39–56. Springer, 2008. 2.3.11

[33] Xavier Boyen and Brent Waters. Compact group signatures without random oracles. In Serge Vaudenay, editor, *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 427–444. Springer, 2006. 5.2

[34] Xavier Boyen and Brent Waters. Full-domain subgroup hiding and constant-size group signatures. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *Public Key Cryptography*, volume 4450 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2007. 1.2, 1.3.3, 1.3.4, 2.3.7, 2.4.13, 2.4.13.1, 2.4.14, 5.3, 5.3.1, 5.1, 5.4.1, 6.1, 6.5, 6.1, 8.1

[35] Ernest F. Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In Atluri et al. [10], pages 132–145. 1

[36] Ernie Brickell and Jiangtao Li. A pairing-based daa scheme further reducing tpm resources. In Alessandro Acquisti, Sean W. Smith, and Ahmad-Reza Sadeghi, editors, *TRUST*, volume 6101 of *Lecture Notes in Computer Science*, pages 181–195. Springer, 2010. 1

[37] Julien Bringer, Hervé Chabanne, David Pointcheval, and Sébastien Zimmer. An application of the boneh and shacham group signature scheme to biometric authentication. In Kanta Matsuura and Eiichiro Fujisaki, editors, *IWSEC*, volume 5312 of *Lecture Notes in Computer Science*, pages 219–230. Springer, 2008. 1

[38] Julien Bringer and Alain Patey. Backward unlinkability for a vlr group signature scheme with efficient revocation check. *IACR Cryptology ePrint Archive*, 2011:376, 2011. 1, 2.4.8.1, 4.1

[39] Jan Camenisch and Victor Shoup. Practical verifiable encryption and decryption of discrete logarithms. In Dan Boneh, editor, *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 126–144. Springer, 2003. 2.5.1

[40] Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups (extended abstract). In Burton S. Kaliski Jr., editor, *CRYPTO*, volume 1294 of *Lecture Notes in Computer Science*, pages 410–424. Springer, 1997. 1, 2.4.4

[41] Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors. *Public Key Cryptography - PKC 2011 - 14th International Conference on Practice and Theory in Public Key Cryptography, Taormina, Italy, March 6-9, 2011. Proceedings*, volume 6571 of *Lecture Notes in Computer Science*. Springer, 2011. 8

[42] David Chaum. An efficient protocol for anonymously providing assurance of the container of the private key. *Submission to the Trusted Computing Group*, 2003. 4.2

[43] David Chaum and Eugène van Heyst. Group signatures. In *EUROCRYPT*, pages 257–265, 1991. 1, 2.4.8

[44] Lidong Chen and Torben P. Pedersen. New group signature schemes (extended abstract). In *EUROCRYPT*, pages 171–181, 1994. 1

[45] Henri Cohen, Gerhard Frey, Roberto Avanzi, Christophe Doche, Tanja Lange, Kim Nguyen, and Frederik Vercauteren. *Handbook of elliptic and hyperelliptic curve cryptography*. Chapman and Hall/CRC, 2010. 2.2

[46] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Krawczyk [77], pages 13–25. 2.5.1, 2.5.5

[47] Ivan Damgård. Non-interactive circuit based proofs and non-interactive perfect zero-knowledge with proprocessing. In Rainer A. Rueppel, editor, *EUROCRYPT*, volume 658 of *Lecture Notes in Computer Science*, pages 341–355. Springer, 1992. 2.4.6

[48] Cécile Delerablée and David Pointcheval. Dynamic fully anonymous short group signatures. In Phong Q. Nguyen, editor, *VIETCRYPT*, volume 4341 of *Lecture Notes in Computer Science*, pages 193–210. Springer, 2006. 1.3.1, 1.3.2, 1.3.4, 1.3.5, 3.1, 3.3, 3.3, 3.4, 3.4, 3.4, 4.3, 4.3, 4.4, 4.4, 4.4, 6.3, 6.3, 6.4, 7.3, 7.3, 7.4, 8

[49] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. 2.5

[50] Yevgeniy Dodis and Aleksandr Yampolskiy. A verifiable random function with short proofs and keys. In Serge Vaudenay, editor, *Public Key Cryptography*, volume 3386 of *Lecture Notes in Computer Science*, pages 416–431. Springer, 2005. 6.3, 7.3

[51] Keita Emura, Atsuko Miyaji, and Kazumasa Omote. A dynamic attribute-based group signature scheme and its application in an anonymous survey for the collection of attribute statistics. *JIP*, 17(1):216–231, 2009. 1, 1.3.1, 1.3.3, 1.3.4, 1.3.5, 2.4.11, 2.4.11, 2.4.17, 3.1, 3.2, 3.3, 3.4, 3.4, 3.1, 3.6, 5.1, 5.3, 5.1, 5.4.2, 5.2, 6.1, 6.2, 6.3, 6.1, 7.2, 7.3

[52] Keita Emura, Atsuko Miyaji, and Kazumasa Omote. A dynamic attribute-based group signature scheme and its application in an anonymous survey for the collection of attribute statistics. In *ARES*, pages 487–492. IEEE Computer Society, 2009. 1, 1, 1.2, 1.3.2, 4.1, 4.3, 4.3, 4.4, 4.4, 4.1, 5.2, 7.1, 7.1, 8.1

[53] Alex Escala, Javier Herranz, and Paz Morillo. Revocable attribute-based signatures with adaptive security in the standard model. In Abderrahmane Nitaj and David Pointcheval, editors, *AFRICACRYPT*, volume 6737 of *Lecture Notes in Computer Science*, pages 224–241. Springer, 2011. 1, 1, 2.4.16, 2.5.5

[54] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In *FOCS*, pages 308–317. IEEE Computer Society, 1990. 2.4.6

[55] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986. 2.4.4

[56] Pierre-Alain Fouque and David Pointcheval. Threshold cryptosystems secure against chosen-ciphertext attacks. In Colin Boyd, editor, *ASIACRYPT*, volume 2248 of *Lecture Notes in Computer Science*, pages 351–368. Springer, 2001. 3.3

[57] Matthew K. Franklin, editor. *Advances in Cryptology - CRYPTO 2004, 24th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, volume 3152 of *Lecture Notes in Computer Science*. Springer, 2004. 8

[58] David Freeman, Michael Scott, and Edlyn Teske. A taxonomy of pairing-friendly elliptic curves. *J. Cryptology*, 23(2):224–280, 2010. 2.2

[59] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008. 2.2

[60] Steven D. Galbraith and Victor Rotger. Easy decision-diffie-hellman groups. *IACR Cryptology ePrint Archive*, 2004:70, 2004. 3.3

[61] Benedikt Gierlichs, Sylvain Guilley, and Debdeep Mukhopadhyay, editors. *Security, Privacy, and Applied Cryptography Engineering - Third International Conference, SPACE 2013, Kharagpur, India, October 19-23, 2013. Proceedings*, volume 8204 of *Lecture Notes in Computer Science*. Springer, 2013. 1.3.4, 8.1

[62] Oded Goldreich. *The Foundations of Cryptography - Volume 1, Basic Techniques*. Cambridge University Press, 2001. 2.4.5

[63] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *J. Cryptology*, 7(1):1–32, 1994. 2.4.3.3

[64] Shafi Goldwasser and Mihir Bellare. Lecture notes on cryptography, 2001. 2.4.1, 2.4.2

[65] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989. 2.4.3, 2.4.3

[66] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM Conference on Computer and Communications Security*, pages 89–98. ACM, 2006. 1.2, 2.4.9

[67] Jens Groth. Fully anonymous group signatures without random oracles. In Kaoru Kurosawa, editor, *ASIACRYPT*, volume 4833 of *Lecture Notes in Computer Science*, pages 164–180. Springer, 2007. 5.4.1

[68] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *EUROCRYPT*, volume 4965 of *Lecture Notes in Computer Science*, pages 415–432. Springer, 2008. 1.3.3, 1.3.4, 1.3.5, 2.4.6, 2.4.6.1, 2.4.7, 2.4.7, 2.4.7.1, 2.4.7.1, 5.3.1, 7.3, 7.4

[69] Satoshi Hada and Toshiaki Tanaka. On the existence of 3-round zero-knowledge protocols. In Krawczyk [77], pages 408–423. 2.3.15

[70] Javier Herranz, Fabien Laguillaumie, Benoît Libert, and Carla Ràfols. Short attribute-based signatures for threshold predicates. In Orr Dunkelman, editor, *CT-RSA*, volume 7178 of *Lecture Notes in Computer Science*, pages 51–67. Springer, 2012. 1, 1.2, 1.3.3, 1.3.4, 1.3.5, 2.4.16, 4.1, 4.5, 4.1, 5.1, 5.4.2, 5.2, 6.1, 6.5, 6.1, 7.1, 7.5, 7.1, 8.1

[71] Rafael Hirschfeld, editor. *Financial Cryptography, Second International Conference, FC'98, Anguilla, British West Indies, February 23-25, 1998, Proceedings*, volume 1465 of *Lecture Notes in Computer Science*. Springer, 1998. 1

[72] Dalia Khader. Attribute based group signature with revocation. *IACR Cryptology ePrint Archive*, 2007:241, 2007. 1, 1.2, 1.3.1, 1.3.2, 1.3.3, 1.3.4, 1.3.5, 2.4.9, 2.4.17, 3.1, 3.1, 3.6, 4.1, 4.3, 4.1, 5.1, 5.3, 5.1, 5.4.2, 5.2, 6.1, 6.2, 6.3, 6.1, 7.1, 7.2, 7.3, 7.1, 8.1

[73] Dalia Khader. Attribute based group signatures. *IACR Cryptology ePrint Archive*, 2007:159, 2007. 1, 1, 1.2, 1.3.3, 1.3.4, 1.3.5, 2.4.17, 3.1, 5.1, 5.2, 5.3, 6.1, 6.3, 7.3

[74] Joe Kilian and Erez Petrank. An efficient noninteractive zero-knowledge proof system for np with general assumptions. *J. Cryptology*, 11(1):1–27, 1998. 2.4.6

[75] Joe Kilian and Erez Petrank. Identity escrow. In Krawczyk [77], pages 169–185. 1

[76] Neal Koblitz and Alfred Menezes. Another look at generic groups. *IACR Cryptology ePrint Archive*, 2006:230, 2006. 2.5.4

[77] Hugo Krawczyk, editor. *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings*, volume 1462 of *Lecture Notes in Computer Science*. Springer, 1998. 8

[78] Fabien Laguillaumie, Pascal Paillier, and Damien Vergnaud. Universally convertible directed signatures. In Bimal K. Roy, editor, *ASIACRYPT*, volume 3788 of *Lecture Notes in Computer Science*, pages 682–701. Springer, 2005. 2.3.10, 2.3

[79] Xiaohui Liang, Zhenfu Cao, Jun Shao, and Huang Lin. Short group signature without random oracles. In Sihan Qing, Hideki Imai, and Guilin Wang, editors, *ICICS*, volume 4861 of *Lecture Notes in Computer Science*, pages 69–82. Springer, 2007. 1.2, 1.3.3, 2.3.8, 2.4.14, 2.4.14.1, 5.4, 5.4.1, 5.4.1, 5.4.2, 5.2, 8.1

[80] Benoît Libert, Thomas Peters, and Moti Yung. Group signatures with almost-for-free revocation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 571–589. Springer, 2012. 1.3.4

[81] Benoît Libert, Thomas Peters, and Moti Yung. Scalable group signatures with revocation. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT*, volume 7237 of *Lecture Notes in Computer Science*, pages 609–627. Springer, 2012. 1.3.4

[82] Benoît Libert and Damien Vergnaud. Group signatures with verifier-local revocation and backward unlinkability in the standard model. In Juan A. Garay, Atsuko Miyaji, and Akira Otsuka, editors, *CANS*, volume 5888 of *Lecture Notes in Computer Science*, pages 498–517. Springer, 2009. 1, 1.2, 1.3.2, 1.3.5, 2.4.8.1, 2.5.1, 2.5.5, 4.1, 4.3, 4.5, 4.1, 7.1, 7.2, 7.3, 7.4, 7.5, 7.1, 7.6, 8.1

[83] Ben Lynn. *On the implementation of pairing-based cryptosystems*. PhD thesis, Stanford University, 2007. 2.2

[84] Hemanta K. Maji, Manoj Prabhakaran, and Mike Rosulek. Attribute-based signatures. *IACR Cryptology ePrint Archive*, 2010:595, 2010. 1

[85] Hemanta K. Maji, Manoj Prabhakaran, and Mike Rosulek. Attribute-based signatures. In Aggelos Kiayias, editor, *CT-RSA*, volume 6558 of *Lecture Notes in Computer Science*, pages 376–392. Springer, 2011. 1, 1.3.4, 2.4.16, 2.4.16, 2.5.4, 7.2

[86] Alfred Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996. 2.4.2

[87] Victor S. Miller. The weil pairing, and its efficient calculation. *J. Cryptology*, 17(4):235–261, 2004. 2.2

[88] Atsuko Miyaji, Masaki Nakabayashi, and Shunzou Takano. New explicit conditions of elliptic curve traces for fr-reduction. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 84(5):1234–1243, 2001. 3.6

[89] Atsuko Miyaji, Masaki Nakabayashi, and Shunzou TAKANO. New explicit conditions of elliptic curve traces for fr-reduction, 2001. 4.5

[90] Toru Nakanishi and Nobuo Funabiki. A short verifier-local revocation group signature scheme with backward unlinkability. In Hiroshi Yoshiura, Kouichi Sakurai, Kai Rannenberg, Yuko Murayama, and Shin ichi Kawamura, editors, *IWSEC*, volume 4266 of *Lecture Notes in Computer Science*, pages 17–32. Springer, 2006. 1, 2.4.8.1

[91] Toru Nakanishi and Nobuo Funabiki. A short verifier-local revocation group signature scheme with backward unlinkability. *IEICE Transactions*, 90-A(9):1793–1802, 2007. 1.3.2, 4.1, 4.3

[92] Tatsuaki Okamoto and Katsuyuki Takashima. Efficient attribute-based signatures for non-monotone predicates in the standard model. In Catalano et al. [41], pages 35–52. 1, 2.4.16, 2.5.5

[93] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *EUROCRYPT*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 1999. 3.3

[94] M. O. Rabin. Digitalized signatures and public-key functions as intractable as factorization. Technical report, Cambridge, MA, USA, 1979. 2.5.2

[95] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978. 1, 2.4.1

[96] Alfredo De Santis, Giovanni Di Crescenzo, and Giuseppe Persiano. Randomness-optimal characterization of two np proof systems. In José D. P. Rolim and Salil P. Vadhan, editors, *RANDOM*, volume 2483 of *Lecture Notes in Computer Science*, pages 179–193. Springer, 2002. 2.4.6

[97] Claus-Peter Schnorr. Efficient signature generation by smart cards. *J. Cryptology*, 4(3):161–174, 1991. 2.4.4

[98] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266. Springer, 1997. 2.5.4

[99] Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. *IACR Cryptology ePrint Archive*, 2004:332, 2004. 2.5.1

[100] Dawn Xiaodong Song. Practical forward secure group signature schemes. In *ACM Conference on Computer and Communications Security*, pages 225–234, 2001. 1, 2.4.8.1

[101] Ahren Studer, Elaine Shi, Fan Bai, and Adrian Perrig. Tacking together efficient authentication, revocation, and privacy in vanets. In *SECON*, pages 1–9. IEEE, 2009. 1

[102] Brent Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 114–127. Springer, 2005. 2.4.15

List of Publications

1. International Journals

(i) Syed Taqi Ali and B B Amberker, A dynamic constant size attribute-based group signature scheme with attribute anonymity, *Int. J. Information Privacy, Security and Integrity (IJIPSI)*, Vol. 1, No. 4, pp. 312-343, Apr. 2013.

(ii) Syed Taqi Ali and B B Amberker, Dynamic attribute-based group signature with verifier-local revocation and backward unlinkability in the standard model, *International Journal of Applied Cryptography (IJAC)*, Vol. 3, No. 2, pp. 148-165, Jun. 2014.

(iii) Syed Taqi Ali and B B Amberker, Attribute-based group signature without random oracles with attribute anonymity, *International Journal of Information and Computer Security (IJICS)*, Vol. 6, No. 2, pp. 109-132, Oct. 2014.

2. International Conferences

(i) Syed Taqi Ali and B B Amberker, Short Attribute-Based Group Signature without Random Oracles with Attribute Anonymity, *International Symposium on Security in Computing and Communications (SSCC-2013)*, Mysore, India, Aug. 2013.

(ii) Syed Taqi Ali and B B Amberker, Dynamic Attribute Based Group Signature with Attribute Anonymity and Tracing in the Standard Model, *Third International Conference on Security, Privacy, and Applied Cryptography Engineering (SPACE 2013)*, Kharagpur, India, Oct. 2013.

Acknowledgment

I would like to express my gratitude to my Research Supervisor Prof. B. B. Amberker, for his guidance and support in every stage of my work, and motivating me to work peacefully. I feel lucky to be his student.

I would like to thank my Doctoral Scrutiny Committee (DSC) Members: Dr. D. V. L. N. Somayajulu (Chairman), Dr. R. B. V. Subramaanyam (Member) and Dr. D. Srinivasacharya (Member) for their regular support and encouragement.

I would like to thank the Institute and the staff members for providing sufficient resources in successful completion of my thesis work.

I would like to thank my family for their love, patience and support. I owe special thank to my mother who is not present to share this moment. May god give peace to her soul.

I would like to thank all my friends, especially my M.Tech and Ph.D colleagues for their company and support.

SYED TAQI ALI

Index