Expressive Subgroup Signatures

Xavier Boyen [1] and Cécile Delerablée [2]

[1] Voltage, Palo Alto, California — xb@boyen.org
[2] Orange Labs - ENS — cecile.delerablee@orange-ftgroup.com

Abstract. In this work, we propose a new generalization of the notion of group signatures, that allows signers to cover the entire spectrum from complete disclosure to complete anonymity. Previous group signature constructions did not provide any disclosure capability, or at best a very limited one (such as subset membership). Our scheme offers a very powerful language for disclosing exactly in what capacity a subgroup of signers is making a signature on behalf of the group.

1 Introduction

Collective signatures allow an individual to make a signed statement anonymously on behalf of a coalition. Whereas ring signatures [30] are unconditionally anonymous, group signatures [17] come with an anti-abuse protection mechanism, whereby a tracing authority can lift a signature's anonymity to uncover the identity of the signer in case of necessity. In group signatures, membership to the group must be restricted and subject to a formal vetting and enrollment process of its members: these are desirable properties in many applications.

In many contexts, the blunt anonymity provided by group signatures goes too far, and it would be preferable to go half-way between full anonymity and full disclosure — e.g., to boost the credibility of a statement without completely engaging the individual responsibility of the signer. This is especially important in groups with many members, or with members of differing competences, or any time several signers wish to sign a joint statement while retaining some anonymity within a larger group.

The "Ad Hoc Group Signatures" from [20] at Eurocrypt 2004 provided a partial answer, by allowing a signer to disclose that he or she belongs to a particular subset of the group, not just the entire group. The "Mesh Signatures" from [11] at Eurocrypt 2007 went further by providing a very expressive language that signer(s) could use to make very specific statements as to the capacity in which they created the signature (such as, "undersigned by, either, five senators, or, two deputies and the prime minister"). Unfortunately mesh signatures were a generalization of ring signatures with no provision for traceability.

In this work, we propose a group signature with a mesh-like expressive language for proving and verifying complex propositions about group membership, including those whose fulfillment requires credentials from more than one group member. Given a valid signature, anyone can verify that it was created by a subgroup of signers that satisfied a certain condition (stated as an expression given with the signature), and learn nothing else. However, the tracing authority is able to unblind the signature and find out exactly how the condition was fulfilled, and thus who the signers are.

The construction we propose is based primarily on the mesh signatures of [11], which we modify in order to equip with a tracing capability. The tracing mechanism is inspired by a number of recent group signature constructions [12, 1, 13, 21], which all made use of zero-knowledge proof systems in bilinear groups of composite order [10, 22, 23]. Compared to those, however, the present work provides a technical novelty: the composite algebraic group and the zero-knowledge proofs had to be flipped "inside out" in order to be compatible with the more expressive language that we implement.

Our signatures are efficient, both in the asymptotic and the practical sense: the signature size will be linear in the size of the expression that it represents, with a small proportionality constant. Although it would surely be nice to shrink the cryptographic part of the signature further down to a constant-size component, this is not of great importance here since the total signature size and verification time would still have to be linear or worse — because the verification expression has to be stated and used somewhere, and the access structure it represents is likely to change from one signature to the next. (Contrast this with regular group signatures, where it is more desirable to have signatures of constant size, because the group composition is fixed and need not be repeated.) For comparison, our fine-grained group signature is essentially as short and efficient as the mesh signature of [11], which is the most relevant benchmark for it is the only known anonymous signature that is as expressive as ours (albeit lacking the tracing capability).

Our scheme satisfies (suitable version of) the usual two main security properties of group signatures originally defined in [5]. The two properties are here: Full Anonymity for CPA-attacks [9] and Full Traceability with Dynamic Join [6], from which other natural requirements such as existential unforgeability, non-frameability, strong exculpability, collusion resistance, etc., can be shown to follow (see [5, 6]). We shall define the two core properties precisely as they generalize to our more expressive notion of group signatures, and prove that our scheme satisfies them under previously analyzed complexity assumptions, in the standard model (unless a join protocol is used for strongly exculpability, in which case we need either random oracles or a common reference string to realize extractable commitments).

The name "Expressive Subgroup Signatures" is meant to capture that at the core these are group signatures, albeit not ones whose level of (revocable) anonymity is fixed and depends only on the group composition, but can be adjusted "in the field" with great precision, any time a new signature is created by any subgroup of signer(s) within the group boundaries.

1.1 Related Work

Ring signatures. Ring signatures were introduced in [30]. Each user in the system has a public key, and can generate a ring signature. Such a signature implicates some other users, chosen by the signer, and is such that a verifier is convinced that someone in the ring formed by the public keys of the signer and the chosen members is responsible for the signature, but nothing else. Constant-size ring signatures were constructed in [20]. Recently, a number of ring signatures without random oracles have been constructed from pairings, such as [18, 7, 31, 11].

Mesh signatures. Mesh signatures were recently proposed [11] as a powerful generalization of ring signatures, with a rich language for striking the desired balance from full disclosure to full anonymity and almost anything in between (including complex statements involving, *inter alia*, trees of conjunctions and disjunctions of multiple potential signers). The work of [11] gave the first unconditionally anonymous ring signatures without random oracles as a special case.

Group signatures. Group signatures were first proposed in [17] to allow any member of a specific group to sign a message on behalf of the group, while keeping the signer anonymous within the group. However, a tracing authority has the ability to uncover the identity of the signer, which it should only do in certain extenuating circumstances. Group signatures have attracted much attention in the research community; we mention for example the works of [1–3, 5, 6, 9, 12–16, 27, 29, 32].

For completeness, we mention the recently proposed notion of "attribute-based group signature" [26, 25], which, contrarily to what the name might suggest, is a far cry from fulfilling our goal. (These signatures are rather inflexible, as they require that the attribute trees be constructed by the setup authority, and not the signer. Furthermore, verifying each attribute tree requires a different public key which must be requested *interactively* from the setup authority.)

2 Preliminaries

2.1 Composite-Order Pairings

Our construction makes use of bilinear pairings defined over groups of composite order, whose cryptographic applications were first investigated in [10].

A (symmetric) composite-order pairing is an efficiently computable function $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$, where $\mathbb{G}$ and $\mathbb{G}_T$ are finite cyclic groups of order $N = pq$, and with the following properties:

Bilinearity: $\forall u, v \in \mathbb{G}$, $\forall a, b \in \mathbb{Z}$, we have $e(u^a, v^b) = e(u, v)^{ab \bmod N}$.

Non-degeneracy: $\exists g \in \mathbb{G}$ such that $e(g, g)$ has order N and thus generates $\mathbb{G}_T$.

Although the composite group order N can be made public, it is usually important that the factorization $N = pq$ remains a secret. The most common hardness assumption that relies on hardness of factoring in the context of bilinear maps is called the Decision Subgroup assumption.

The Decision Subgroup Assumption. Let $\mathbb{G}$ be a bilinear group of order $N = pq$. Let $\mathbb{G}_p$ be the subgroup of points of order p with no residue of order q, that is, $u \in \mathbb{G}_p$ iff $u^p = 1 \in \mathbb{G}$. Similarly, we let $\mathbb{G}_q$ be the subgroup of points of order q congruent to 1 in $\mathbb{G}_p$.

Informally, the decision subgroup assumption says that one cannot efficiently distinguish $\mathbb{G}$ from $\mathbb{G}_p$ with non-negligible advantage.

Formally, we consider an "instance generator" $\mathcal{G}$, which, on input 1^λ, outputs a tuple $(p, q, \mathbb{G}, \mathbb{G}_T, e)$, where p and q are random λ-bit primes, $\mathbb{G}$ and $\mathbb{G}_T$ are cyclic groups of order $N = pq$, and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a bilinear pairing. The subgroup decision problem is, given $(N, \mathbb{G}, \mathbb{G}_T, e)$ derived from an execution of $\mathcal{G}(1^\lambda)$, to decide whether a given element w was chosen randomly in $\mathbb{G}$ or in $\mathbb{G}_p$. The Subgroup Decision assumption states that this is infeasible in polynomial time with non-negligible probability above $1/2$, that of a random guess.

An alternative definition is to give the distinguisher two reference generators $g_N \in \mathbb{G}$ and $g_p \in \mathbb{G}_p$ in addition to $(N, \mathbb{G}, \mathbb{G}_T, e)$ and w; the task remains to decide whether $w \in \mathbb{G}$ or $w \in \mathbb{G}_p$. We use this definition to simplify our proofs.

The Poly-SDH Assumption. The traceability proof of the ESS scheme will be based on the unforgeability of the mesh signature scheme of [11], which was itself proved from the so-called Poly-SDH assumption in bilinear groups. The (q, ℓ)-Poly-SDH is a parametric assumption that mildly generalizes the earlier Strong Diffie-Hellman assumption [8] in such groups. It can be stated as:

(Poly-SDH) Given g, g^{α_1}, ..., $g^{\alpha_\ell} \in \mathbb{G}$ and $q\,\ell$ pairs $(w_{i,j},\ g^{1/(\alpha_i + w_{i,j})})$ for $1 \le i \le \ell$ and $1 \le j \le q$, choose a list of values w_1, ..., $w_\ell \in \mathbb{F}_p$ and output ℓ pairs $(w_i,\ g^{r_i/(\alpha_i + w_i)})$ such that $\sum_{i=1}^{\ell} r_i = 1$.

2.2 Group Signatures

A group signature scheme involves a group manager, an opening manager, group members and outsiders. The group manager is able to add new members by issuing private keys thanks to a master key MK, while the opening manager can use the tracing key TK to revoke the anonymity of any group signature.

Such a scheme is specified as a tuple $\mathcal{GSS} = (\mathsf{Setup}, \mathsf{Join}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Trace})$ of algorithms described as follows:

– The setup algorithm Setup generates, according to a security parameter, a public verification key PK, a master key MK, and a tracing key TK.

– The enrollment algorithm, Join, that generates a private signing key using the master key MK. The private key is then given to the new user.
– The group signing algorithm, Sign, that outputs a signature σ on a message M, using a group member's private key.
– The (usually deterministic) group signature verification algorithm, Verify, that takes as input the group verification key PK and a signature σ on a message M, and outputs either `valid` or `invalid`.
– The (usually deterministic) tracing algorithm, Trace, that takes a valid signature σ and a tracing key TK, and outputs the identity of a group member or $\perp$.

The following correctness and security properties are required.

Consistency. Given a group signature generated by a honest group member, the verify algorithm should output `valid`, and the tracing algorithm should identify the actual signer.

Security. Bellare, Micciancio, and Warinschi [5] characterize the fundamental properties of group signatures in terms of two crucial security properties from which a number of other properties follow. The two important properties are:

Full Anonymity. This requires that no PPT adversary be able to decide (with non-negligible probability over one half) whether a challenge signature σ on a message M emanates from user id_1 or id_2, where id_1, id_2, and M are chosen by the adversary. In the original definition of [5], the adversary is given access to a tracing oracle, which it may query before and after being given the challenge σ, much in the fashion of IND-CCA2 security for encryption.

Full Traceability. This requires that no coalition of users be able to generate, in polynomial time, a signature that passes the *Verify* algorithm but fails to trace to a member of the coalition under the *Trace* algorithm. According to this notion, the adversary is allowed to ask for the private keys of any user of its choice, adaptively, and is also given the secret key TK to be used for tracing—but of course not the enrollment master key MK.

It is noted in [5] that the full traceability property implies that of *exculpability* [4], which is the requirement that no party should be able to frame a honest group member as the signer of a signature he did not make, not even the group manager. However, the model of [5] does not consider the possibility of a (long-lived) group master, which leaves it as a potential framer. To address this problem and achieve the notion of *strong exculpability*, introduced in [2] and formalized in [29, 6], one also needs an interactive enrollment protocol, call *Join*, at the end of which only the user himself knows his full private key; the same mechanism may also enable concurrent dynamic group enrollment [6, 29]. In this work, we opt for this stronger notion and thus we shall explicitly describe such a Join protocol.

We remark that some of the preceding requirements have been relaxed in [9] and a fair number of subsequent works, by withholding access to the tracing oracle during the anonymity and/or traceability games, thus mirroring the notion of IND-CPA security for encryption. In our model, we shall also withhold the tracing key from the adversary in the traceability game. We argue that this is only a minor concession, since in typical group signature schemes the tracing authority is the final adjudicator without any possibility of appeal to a higher authority. It seems rather picayune to assume that the tracing authority would ever need to cheat in a traceability game, which is essentially what the full traceability model amounts to. Incidentally, the (much more frequently debated) property of strong exculpability seems better motivated in comparison.

We refer the reader mainly to [5] for more precise definitions of these and related notions.

2.3 Mesh Signatures

We now briefly recapitulate the notion of mesh signature introduced in [11].

In short, a mesh signature is a non-interactive witness-indistinguishable proof that some monotone boolean expression $\Upsilon(L_1, \ldots, L_n)$ is true, where the input literals L_i denote the validity of "atomic signatures" of the form $\{Msg_i\}_{Key_i}$ for arbitrary messages and different keys. (The special case of ring signatures [30] corresponds to Υ being a flat disjunction.)

Admissible mesh expressions Υ consist of trees of And, Or, and Threshold gates, and single-use input literals, generated by the following grammar:

$$\begin{array}{rll}
\textsc{expr} ::= & L_1 \mid \ldots \mid L_\ell & \text{single-use input symbols} \\
\mid & \geq_t \{\textsc{expr}_1, \ldots, \textsc{expr}_m\} & t\text{-out-of-}m \text{ threshold, with } 1 < t < m \\
\mid & \wedge \{\textsc{expr}_1, \ldots, \textsc{expr}_m\} & m\text{-wise conjunction, with } 1 < m \\
\mid & \vee \{\textsc{expr}_1, \ldots, \textsc{expr}_m\} & m\text{-wise disjunction, with } 1 < m
\end{array}$$

Not all literals need to be true in order for Υ to be satisfied. However the mesh signature must only attest to the truth of Υ without revealing how it is satisfied: this is the *anonymity* property. Conversely, a signer should not be able to create a mesh signature without possessing a valid atomic signature for every literal set to true: this is the *unforgeability* property.

2.4 Security of Expressive Subgroup Signatures

ESS are just as expressive as mesh signatures, and provide the same anonymity, except that the latter can be lifted by a tracing authority. We require:

ESS Anonymity. The notion of anonymity we seek is not that we wish to hide the identity of the users named in the ESS expression Υ (which is public anyway), but to hide who among the users actually created the signature.

Precisely, we require that the identity of the actual signer(s) be computationally indistinguishable in the set of all valid ESS signatures specified by the same expression Υ, even under full exposure of all user keys. This is the same notion as in the mesh signatures of [11], except that here the requirement is computational and not information-theoretic in order not to stymie the tracing authority, and is of course conditional on the secrecy of the tracing key.

ESS Traceability. Traceability is the group-signature version of unforgeability. For ESS, as for mesh signatures before them, this notion is tricky to formalize because of the potentially complex dependences that may exist between good and forged signatures. To see this, consider a coalition of two forgers, U_1, U_2, and a honest user, U_3. Suppose that the forgers fabricate a valid ESS signature σ for the expression $\Upsilon = \{m_1\}_{U_1} \vee (\{m_2\}_{U_2} \wedge \{m_3\}_{U_3})$, that can be traced the subgroup U_2, U_3. Is that a successful forgery? What if σ traced to U_2 only?

The answer is a double "yes": in the first case, because U_3 was wrongly designated by the tracer; and in the second case, because U_2 alone could not have satisfied Υ, which means that some guilty party escaped detection. If on the other hand, the finger were pointed at U_1, the signature would have a satisfactory explaination that involves only (the parties that we know to be) the culprits: this would be a failed forgery since the coalition got caught.

The ESS traceability challenge is thus, for any coalition of signers, to come up with a valid signature σ for an expression $\Upsilon(L_1, \ldots, L_n)$, such that σ, either, traces to at least one user outside of the coalition, or, traces to a subset of the coalition that does not validate Υ (that is, when Υ is "false" after setting the designated literals L_i to "true" and only those).

Strong Exculpability. This last notion is borrowed straight from group signatures [2, 29, 6], and is orthogonal to the above. It gives any user the possibility to dispute his alleged binding to any certificate that he did not request. To defend from such accusation, the group manager (who signed the disputed certificate) must produce a valid certificate request signed by the user's individual key registered in some PKI. The enrollment protocol must guarantee that only the user learns the private key associated with their certificates. Together, this prevents the ESS group manager from framing users for signatures they did not make.

2.5 Formal Security Models

We now specify the formal ESS security model in accordance with the previously stated requirements.

Anonymity The ESS anonymity game is played between a challenger and an adversary.

Initialization. The challenger gives to the adversary the public parameters of an ESS.

Interaction. The following occurs interactively, in any order, driven by the adversary.

User enrollment. The adversary may ask the challenger to enroll a polynomially bounded number of new users in the group. The adversary may either impersonate the user in the enrollemnt protocol, or ask the challenger to simulate it all by itself. The resulting public certificates are published.

Signature queries. The adversary may ask the challenger to sign any ESS expression Υ on behalf of the users it controls.

Key recovery. The adversary may ask the challenger to reveal the group signing key of any user.

The challenger processes each request before accepting the next one.

Challenge: The adversary finally output a specification Υ and two sets of assignments χ_1 and χ_2 to its literals L_i, that both cause Υ to be satisfied. These truth assignments indicate which users are supposed to sign Υ. The adversary must also supply the necessary atomic signatures for the users for which it has the keys.

The challenger chooses one assignment $b \in \{1, 2\}$ at random, and creates an ESS signature σ on the specification Υ using only atomic signatures corresponding to the true literals in χ_i. The signature σ is given to the adversary.

Guess: The adversary makes a guess b', and wins the game if $b = b'$.

The adversary's advantage in the ESS anonymity game is $\Pr[b = b'] - 1/2$, where the probability is defined over the random coins used by all the parties.

Traceability The ESS traceability game is played between a challenger and an adversary.

Initialization. The challenger gives to the adversary the public parameters of an ESS. The challenger also reserves a number ℓ of user indices to represent the "honest users" under its control.

Interaction. The following occurs interactively, in any order, driven by the adversary.

Honest user enrollment. The adversary may request that the challenger create up to ℓ honest users, kept in the challenger's control. The challenger publishes the corresponding certificates.

Corrupted user enrollment. The adversary makes polynomially user enrollment queries, for the users under the adversary's control. The adversary chooses or receives the user secret keys in accordance with the chosen enrollment protocol. The challenger computes the corresponding certificates in accordance with the enrollment protocol, and publishes them.

ESS signature queries. The adversary makes up to q ESS signature queries, one at a time, on specifications Υ_j, indicating to the challenger which ones of the honest users are supposed to issue the signature. To be acceptable, each request must be for a signature that the specified subset of honest users is supposed to be able to make based on the specification and supporting atomic signatures provided by the adversary.

The adversary may also make up to q queries for atomic signatures, to each of the users controlled by the challenger.

The challenger processes each request before accepting the next one.

Forgery: The adversary finally output a fresh valid ESS signature σ for some specification Υ of its choice. It wins the game if the list of literals L_i designated by the tracing algorithm on input σ fails to satisfy the two following properties:

1. All the designated literals L_i correspond to atomic signatures $\{Msg_i\}_{User_i}$ under the adversary's control (either because the adversary controls the corresponding user, or had obtained the atomic signature by querying the challenger).
2. The specification formula $\Upsilon(..., L_i, ...)$ can be satisfied by setting all the designated literals to "true" ($\top$) and all the other literals to "false" ($\bot$).

The adversary's advantage in the ESS traceability game is simply the probability that it wins the game. The probability is defined over the random coins used by all the parties.

3 Construction

Our Expressive Subgroup Signature construction will bring together a number of different techniques.

To get the anonymity properties we seek, we will naturally start with the ring/mesh signatures from [11], which comes with a powerful language and proof system. We use it to create an anonymous group identification mechanism for certificates issued by the group manager. Since we need a signature scheme and not just an identification scheme, we shall extend the certificates into certificate chains ending with actual signatures from users' keys. This part is easy to do using the mesh language, so this step will be a simple matter of specifying how the various terminal signatures and their supporting certificates should be assembled. This gives us a multi-user anonymous signature with a central authority. However, we still lack traceability.

To get traceability, we need to build a trapdoor that will remove the blinding from the mesh signatures. Recall that the ring and mesh signatures from [11] consist of one signature element per ring or mesh member. Some of those elements are "live", meaning that they were created using the member's actual secret key.

The remaining elements are "blank" and do not contribute to the verification equation. Since it would be easy to tell who the signers were just by finding the live elements, the elements are information-theoretically blinded so that they all look the same. Here, to get traceability, we shall swap out the perfect blinding for one that has a trapdoor. Since the mesh signatures require a bilinear pairing for their verification, we shall add the trapdoor into the bilinear group, using a standard trick used in several previous constructions [22]. We simply translate the signatures into a bilinear group of composite order, and restrict the blinding elements to one of its two algebraic subgroups. An adversary will just see smoke under the proper hardness assumption [10]; but a tracing authority that knows the order of the subgroups will be able to cancel the blinding (by pairing each signature component with a neutral element in that subgroup), and hence distinguish which signature components are live and which ones are blank.

In the following subsections, we explain step-by-step how to construct expressive subgroup signatures. We work in an algebraic group $\mathbb{G}$ of composite order $N = pq$ and equipped with a bilinear pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. We call $\mathbb{G}$ the bilinear group and $\mathbb{G}_T$ the targer group; both are of order N. Bilinearity is the property that $\forall u, v \in \mathbb{G}$, $\forall a, b \in \mathbb{Z}$, $e(u^a, v^b) = e(u, v)^{ab \bmod N}$.

3.1 User Credentials

Users must be affiliated with the group in order to create signatures, which means that they must have acquired proper credentials from the group manager (which controls the user vetting and enrollment process).

In its most basic instantiation, a certificate for user i could simply be a secret key for the Boneh-Boyen signature scheme [8]. The secret key $(y_i, z_i) \in \mathbb{Z}_p^2$ would be securely handed over to the user, and the corresponding public key $(g^{y_i}, g^{z_i}) \in \mathbb{G}^2$ signed by the group manager and perhaps published as part of the group description. A signature on $m \in \mathbb{Z}_p$ would be a random pair $(t, S) \in \mathbb{Z}_p \times \mathbb{G}$, where $S = g^{1/(y_i+m+z_i t)}$, which is verifiable by testing $e(S, g^{y_i} g^m g^{z_i t}) = e(g, g)$. The drawback is that the group manager would know y_i and z_i and would thus be able to create signatures on the user's behalf. We would also need to embed a tracing trapdoor into all user-generated signatures.

In the preferred instantiation, a group certificate will depend on a secret component that is known only to the user, to prevent users from being framed. It should also depend on a secret from the manager, to guarantee traceability. Using a technique close to the one proposed by Delerablée and Pointcheval [19], we let the credentials for user i consist of a secret key (x_i, y_i, z_i) and a public certificate (A_i, B_i, C_i), where $A_i = g^{y_i/(\gamma+x_i)}$, $B_i = g^{1/(\gamma+x_i)}$, and $C_i = g^{z_i/(\gamma+x_i)}$, for some random x_i. Here, γ and $\Gamma = h^\gamma$ are respectively the secret and public key of the group manager, and g and h are two fixed generators of the group $\mathbb{G}$. The certificate of user i is the triple (A_i, B_i, C_i) signed by the group manager. For randomly chosen $t \in \mathbb{Z}_p$, an "atomic signature" on $m \in \mathbb{Z}_p$ will be a pair:

$$\sigma = (t, S) \in \mathbb{Z}_p \times \mathbb{G} \qquad \text{s.t.} \qquad S = (\Gamma h^{x_i})^{1/(y_i+m+z_i t)} .$$

The verification equation is thus: $e(S, A_i B_i^m C_i^t) = e(h, g)$.

For simplicity reasons, we can merely suppose a enrollment protocol where the user chooses (y_i, z_i), sends (g^{y_i}, g^{z_i}) to the group manager along with a proof of knowledge, and receives (x_i, A_i, B_i, C_i) in return. Nevertheless, following a technique from [19], in Section 3.7 we present a more complex "dynamic" enrollment protocol, which renders our scheme secure under concurrent join [28], and provides strong user exculpability [6] against dishonest managers.

3.2 Atomic Signatures

Using their credentials, users are able to create atomic signatures on any message of their choice, which for simplicity we assume represented as an integer $m \in \mathbb{Z}_p$. Atomic signatures provide no anonymity; they merely serve as building blocks in more complex assemblies.

An atomic signature created from credentials as above is a pair $(t, S) \in \mathbb{Z}_p \times \mathbb{G}$ that satisfies a verification equation of the form,

$$e(S, \underbrace{A_i B_i^m C_i^t}_{R}) = e(h, g) \ ,$$

with respect to a publicly verifiable certificate (A_i, B_i, C_i) associated to user i. We observe for later use that this is exactly a Boneh-Boyen signature, and that the right-hand side $e(h, g)$ in the verification equation is the same for all users.

3.3 Ring Signatures

Once we have atomic signatures of the previous form, we can easily construct an information-theoretically anonymous ring signature, based on the approach proposed in [11]. Suppose that there are n users with public certificates (A_1, B_1, C_1) through (A_n, B_n, C_n), and consider the following verification equation for some message m, or more generally, for respective user messages m_1 through m_n:

$$\prod_{i=1}^{n} e(S_i, \underbrace{A_i B_i^{m_i} C_i^{t_i}}_{R_i}) = e(h, g) \ .$$

Any one of the n users is able to create, by himself, a vector of n pseudo-signatures (t_i, S_i) for $i = 1, \ldots, n$ that will jointly verify the preceding equation. In order to do so, the user will need his own key and everyone else's certificates. For example, user 1 would pick random $r_2, \ldots, r_n$, and $t_1, \ldots, t_n$, and set:

$$S_1 = (\Gamma h^{x_1})^{1/(y_1 + m_1 + z_1 t_1)} \cdot \left[\prod_{i=2}^{n} R_i^{r_i} \right] \ , \quad S_2 = \left[R_1^{-r_2} \right] \ , \quad \ldots \ , \quad S_n = \left[R_1^{-r_n} \right] \ .$$

It is easy to see that, for any random choice of $r_i \in \mathbb{Z}_p$, the blinding terms in the square brackets will cancel each other in the product of pairings in the verification equation; *e.g.*, $e(R_2^{r_2}, R_1)$ from S_1 will cancel $e(R_1^{-r_2}, R_2)$ from S_2. What is left is the Boneh-Boyen signature component $(\varGamma h^{x_1})^{1/(y_1+m_1+z_1 t_1)}$ in S_1, which in the verification equation will produce the value $e(h, g)$ we seek.

For the example of user 1 being the actual signer, the cancellation that occurs is, *in extenso*, if we let $S'_1 = (\varGamma h^{x_1})^{1/(y_1+m_1+z_1 t_1)}$:

$$\begin{aligned}\prod_{i=1}^{n} e(S_i, R_i) &= e(S_1, R_1) \cdot e(\prod_{i=2}^{n} R_i^{r_i}, R_1) \cdot \prod_{i=2}^{n} e(S_i, R_i) \\ &= e(S'_1, R_1) \cdot e(\prod_{i=2}^{n} R_i^{r_i}, R_1) \cdot \prod_{i=2}^{n} e(R_1^{-r_i}, R_i) \\ &= e(h, g) \cdot \prod_{i=2}^{n} e(R_i, R_1)^{r_i} \cdot \prod_{i=2}^{n} e(R_1, R_i)^{-r_i} \qquad = e(g, h)\end{aligned}$$

The point is that user 2 (or any other user j) could have achieved the same result by using his own secret key inside S_2 (or S_j), but nobody else could, without one of the users' key. Also, because there are $2n$ components in the signature, but $2n-1$ randomization parameters and 1 perfectly symmetric verification equation, it is easy to see that the joint distribution of the full signature $(t_i, S_i)_{i=1}^{n}$ is the same regardless of which one of the n listed users created it.

Hence, this is a ring signature, *i.e.*, a witness-indistinguishable proof for the disjunction "$\{m_1\}_{\text{user}_1} \vee \{m_2\}_{\text{user}_2} \vee \ldots \vee \{m_n\}_{\text{user}_n}$". The signature can be shown to be unconditionally anonymous, and existentially unforgeable under the n-Poly-SDH assumption [11], which slightly generalizes the SDH assumption [9].

3.4 Mesh Signatures

The next step is to turn those ring signatures into something that is much more expressive. Recall that ring signatures can be viewed as witness-indistinguishable "disjunctions" of individual signatures. Since the disjunction $L_1 \vee L_2 \vee \ldots \vee L_n$ is the least restrictive of all (non-trivial) propositional expressions over $L_1, \ldots, L_n$, it should be possible to express different statements by adding more constraints to the signature. *E.g.*, we could require that supplemental verification equations be satisfied conjointly. The "mesh signatures" of [11] are based on this principle.

A classic result from [24] shows that any monotone propositional expression over a set of literals can be represented efficiently and deterministically using a system of linear equations $\{\sum_{i=1}^{n} \lambda_{i,j} \nu_i = \lambda_j\}_{j=1}^{k+1}$ over the same number of variables: a literal L_i will be true in a true assignment if and only if the corresponding variable ν_i has a non-zero value in the corresponding system solution (of which there may be many).

In the construction of [11], the linear system coefficients $\lambda_{i,j}$ will become public exponents in the verification equations. Depending on the expression it represents, a mesh signature $(t_i, S_i)_{i=1}^{n}$ requires $1 \leq k+1 \leq n$ verification equations (with the usual $R_i = A_i B_i^{m_i} C_i^{t_i}$ computable from public values):

$$\prod_{i=1}^{n} e(S_i, R_i) = e(h, g) \ ,$$
$$\prod_{i=1}^{n} e(S_i, R_i)^{\lambda_{i,1}} = 1 \ ,$$
$$\cdots$$
$$\prod_{i=1}^{n} e(S_i, R_i)^{\lambda_{i,k}} = 1 \ .$$

To make a signature, the signer, or coalition of signers, must prove that the propositional expression has a solution, *i.e.*, that there is a vector of S_i that passes all the equations. This can be done by setting $S_i \leftarrow ((\mathit{\Gamma} h^{x_i})^{1/(y_1+m_i+z_1t_1)})^{\nu_i}$ given any solution vector $(\nu_1, \ldots, \nu_n)$ of the linear system. However, for every index i with a non-zero solution $\nu_i \neq 0$, the signer(s) will be unable to create S_i unless they possess or are able to create the atomic signature $(\mathit{\Gamma} h^{x_i})^{1/(y_1+m+z_1t_1)}$. Only for $\nu_i = 0$ can they get by without it.

This procedure results in a valid signature, but not in an anonymous one. The last step is thus to hide the witness, *i.e.*, the vector $(\nu_1, \ldots, \nu_n)$ used to build the S_i. This is done by adding blinding terms to the S_i just as in the ring signature. The result is an unconditionally anonymous signature for arbitrary monotone propositional expressions.

The entire mesh signing process and its security proofs are somewhat more involved. Full mesh signatures also require a presence of a dummy user "in the sky" (with a public random public key and no known secret key), who will "sign" a hash of the entire mesh expression in order to "seal" it. We refer the reader to [11] for details.

3.5 Tracing Trapdoor

We now have an expressive anonymous signature, albeit not a traceable one. To make mesh signatures traceable, we need to redefine the mesh signature scheme in bilinear groups of composite order N. The factorization $N = pq$ is a trapdoor that is only known to the tracing authority. Let thus $\mathbb{G}_N \simeq \mathbb{G}_p \otimes \mathbb{G}_q$.

ESS signatures are defined as mesh signatures in a composite-order group $\mathbb{G}$. We do require however that the "main" generator g generate only the subgroup $\mathbb{G}_p$ of order p. That is, we impose that $g^p = 1 \in \mathbb{G}$ or equivalently $g \equiv 1 \in \mathbb{G}_q$. Since the A_i, B_i, C_i, and thus the R_i, are powers of g, all those elements will belong in the subgroup $\mathbb{G}_p$ of order p. It is easy to see that, since the verification equation is of the form $\prod e(S_i, R_i) = e(h, g)$, both sides will always evaluate into the target subgroup of order p, with no contribution of order q. It follows that only the $\mathbb{G}_p$ components of the S_i will matter for ESS verification.

In order to provide a tracing capability, we pick h as a generator of the entire group $\mathbb{G}$, hence with a non-trivial component $h \not\equiv 1 \in \mathbb{G}_q$. The same will be true for the public key $\mathit{\Gamma} = h^\gamma$. As a result, all the user-created atomic signatures of the form $S = (\mathit{\Gamma} h^{x_i})^{1/(\dots)}$ will also contain a non-trivial component $S \not\equiv 1 \in \mathbb{G}_q$, which has no effect on the ESS verification equation, per the preceding argument. These order-q components will be our tracers, since they appear in all atomic signatures (which are powers of $h \in \mathbb{G}$), but not in any of the blinding coefficients (which are powers of $g \in \mathbb{G}_p$).

Since we now work in a composite-order group of order N, we redefine the user's signing exponents in $\mathbb{Z}_N$ instead of $\mathbb{Z}_p$.

Remark that if h had no residue of order q, then everything would be in $\mathbb{G}_p$. It would be as if the subgroup $\mathbb{G}_q$ did not exist, and the ESS scheme would reduce to an information-theoretically untraceable mesh signature in $\mathbb{G}_p$. As the Decision Subgroup assumption [10] states that $h \in \mathbb{G}$ and $h \in \mathbb{G}_p$ should look the same to an outsider, it follows that our tracing mechanism cannot be public and will thus require some trapdoor (in this case, the factorization of N).

3.6 Tracing Procedure

The presence of a non-trivial residue of order q in h will act as a silent tracer for lifting the anonymity of any signature, using the factorization of N as trapdoor.

To unblind an ESS signature $(t_i, S_i)_{i=1}^n$, the tracing authority raises each S_i to the power of p, to strip it from all components of order p. Then, for each i:

– If the residue $(S_i)^p = 1$, there was no contribution from h in S_i, hence $\nu_i = 0$, and thus the truth value of the associated literal L_i is "false". Conclusion: user i did not participate in the creation of the ESS signature.
– If the residue $(S_i)^p \neq 1$, there was some h contribution in S_i, hence $\nu_i \neq 0$, and thus the truth value of the associated literal L_i is "true". Conclusion: (an atomic signature issued by) user i took part in the ESS signature.

The tracer can thus efficiently determine the exact set of users that are involved (and in what capacity).

We emphasize that, unlike tracing schemes in many other contexts that can only guarantee that one of the guilty parties will be exposed, here the tracing authority finds out exactly how the signature was constructed, and thus uncovers the identity of *all* of the culprits.

Notice also that such detailed "exhaustive tracing" requires signatures whose size is (at least) linear in the size of the propositional expression. Hence, in that respect, our scheme is optimally compact up to a constant factor.

3.7 Concurrent Join Protocol

As in [19] we can define a Join protocol, using some standard techniques: an extractable commitment (Ext-Commit), a zero-knowledge proof of equality of the discrete logarithms (NIZKPEqDL), and a zero-knowledge proof of knowledge of a discrete logarithm (NIZKPoKDL). During this protocol, a future group member ($\mathcal{U}_i$) interacts with the group manager ($\mathcal{GM}$), in order to obtain a valid group certificate (A_i, B_i, C_i), with a private key (x_i, y_i, z_i), with (y_i, z_i) not known by the group manager. We suppose, as in [19] that there is a separated PKI environment: each user $\mathcal{U}_i$ has a personal secret key $\mathsf{usk}[i]$ and the corresponding certified public key $\mathsf{upk}[i]$.

The details of the Join protocol are presented on figure 1 in the Appendix.

3.8 The Full ESS Construction

The step-by-step construction outlined above gives us the complete ESS scheme. The only operational differences with the mesh signature scheme of [11] are:

1. the setup, which asks for a bilinear group $\mathbb{G}$ of composite order $N = pq$, two generators $g \in \mathbb{G}_p$ and $h \in \mathbb{G}$, and a group manager's public key $\Gamma = h^\gamma$;
2. the existence of two additional algorithms or protocols: one for joining the group, the other for tracing a signature.

The full details of the construction are given in the Appendix.

4 Security

Theorem 1 (Anonymity). *There is no PPT algorithm that wins the Expressive Subgroup Signature anonymity game over $\mathbb{G}$ with advantage ϵ in time τ, unless the Decision Subgroup problem in $\mathbb{G}$ is decidable in time $\tau' \approx \tau$ with advantage $\epsilon' \geq \epsilon/2$.*

Theorem 2 (Traceability). *There is no PPT algorithm that wins the Expressive Subgroup Signature traceability game over $\mathbb{G}$ with advantage ϵ in time τ, unless the Decision Subgroup problem is decidable in time τ' with advantage ϵ', and mesh signatures in $\mathbb{G}_p$ can be existentially forged in time τ'' with advantage ϵ'', where $\tau' + \tau'' \approx \tau$ and $\epsilon' + \epsilon'' \geq \epsilon/2$.*

5 Conclusion

In this work, we have proposed a new generalization of the notion of group signatures, that allows signers to cover the entire spectrum from complete disclosure to complete anonymity. Previous group signature constructions did not provide any disclosure capability, or at best a very limited one (such as subset membership). Our scheme offers a very powerful language for disclosing exaclty in what capacity a subgroup of signers is making a signature on behalf of the group.

References

1. Giuseppe Ateniese, Jan Camenisch, Susan Hohenberger, and Breno de Medeiros. Practical group signatures without random oracles. Cryptology ePrint Archive, Report 2005/385, 2005. `http://eprint.iacr.org/`.
2. Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. A practical and provably secure coalition-resistant group signature scheme. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 255–270, Santa Barbara, CA, USA, August 20–24, 2000. Springer-Verlag, Berlin, Germany.
3. Giuseppe Ateniese, Dawn Xiaodong Song, and Gene Tsudik. Quasi-efficient revocation in group signatures. In Matt Blaze, editor, *FC 2002*, volume 2357 of *LNCS*, pages 183–197, Southampton, Bermuda, March 11–14, 2002. Springer-Verlag, Berlin, Germany.
4. Giuseppe Ateniese and Gene Tsudik. Some open issues and new directions in group signatures. In Matthew Franklin, editor, *FC'99*, volume 1648 of *LNCS*, pages 196–211, Anguilla, British West Indies, February 1999. Springer-Verlag, Berlin, Germany.
5. 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 2003*, volume 2656 of *LNCS*, pages 614–629, Warsaw, Poland, May 4–8, 2003. Springer-Verlag, Berlin, Germany.
6. Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of group signatures: The case of dynamic groups. In Alfred Menezes, editor, *CT-RSA 2005*, volume 3376 of *LNCS*, pages 136–153, San Francisco, CA, USA, February 14–18, 2005. Springer-Verlag, Berlin, Germany.
7. Adam Bender, Jonathan Katz, and Ruggero Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 60–79, New York, NY, USA, March 4–7, 2006. Springer-Verlag, Berlin, Germany.
8. Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 56–73, Interlaken, Switzerland, May 2–6, 2004. Springer-Verlag, Berlin, Germany.
9. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55, Santa Barbara, CA, USA, August 15–19, 2004. Springer-Verlag, Berlin, Germany.
10. Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In Joe kilian, editor, *TCC 2005*, volume 3378 of *LNCS*, pages 325–341, Cambridge, MA, USA, February 10–12, 2005. Springer-Verlag, Berlin, Germany.
11. Xavier Boyen. Mesh signatures. In Moni Naor, editor, *EUROCRYPT*, volume 4515 of *LNCS*, pages 210–227. Springer, 2007.
12. Xavier Boyen and Brent Waters. Compact group signatures without random oracles. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 427–444, St. Petersburg, Russia, May 28 – June 1, 2006. Springer-Verlag, Berlin, Germany.
13. 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 *LNCS*, pages 1–15. Springer, 2007.

14. Jan Camenisch. Efficient and generalized group signatures. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 465–479, Konstanz, Germany, May 11–15, 1997. Springer-Verlag, Berlin, Germany.
15. Jan Camenisch and Jens Groth. Group signatures: Better efficiency and new theoretical aspects. In *Proceedings of SCN 2004*, pages 120–133, 2004.
16. Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 61–76, Santa Barbara, CA, USA, August 18–22, 2002. Springer-Verlag, Berlin, Germany.
17. David Chaum and Eugène van Heyst. Group signatures. In Donald W. Davies, editor, *EUROCRYPT'91*, volume 547 of *LNCS*, pages 257–265, Brighton, UK, April 8–11, 1991. Springer-Verlag, Berlin, Germany.
18. Sherman S. M. Chow, Victor K.-W. Wei, Joseph K. Liu, and Tsz Hon Yuen. Ring signatures without random oracles. In *ASIACCS 06*, pages 297–302, Taipei, Taiwan, 2006. ACM Press.
19. Cécile Delerablée and David Pointcheval. Dynamic fully anonymous short group signatures. In Phong Q. Nguyen, editor, *VIETCRYPT 2006*, volume 4341 of *LNCS*, pages 193–210. Springer, 2006.
20. Yevgeniy Dodis, Aggelos Kiayias, Antonio Nicolosi, and Victor Shoup. Anonymous identification in ad hoc groups. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 609–626, Interlaken, Switzerland, May 2–6, 2004. Springer-Verlag, Berlin, Germany.
21. Jens Groth. Fully anonymous group signatures without random oracles. In Kaoru Kurosawa, editor, *ASIACRYPT*, volume 4833 of *LNCS*, pages 164–180. Springer, 2007.
22. Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for NP. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 339–358, St. Petersburg, Russia, May 28 – June 1, 2006. Springer-Verlag, Berlin, Germany.
23. Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. Cryptology ePrint Archive, Report 2007/155, 2007. `http://eprint.iacr.org/`.
24. Mauricio Karchmer and Avi Wigderson. On span programs. In *Proceedings of Structures in Complexity Theory*, pages 102–111, 1993.
25. Dalia Khader. Attribute based group signature with revocation. Cryptology ePrint Archive, Report 2007/241, 2007. `http://eprint.iacr.org/`.
26. Dalia Khader. Attribute based group signatures. Cryptology ePrint Archive, Report 2007/159, 2007. `http://eprint.iacr.org/`.
27. Aggelos Kiayias and Moti Yung. Extracting group signatures from traitor tracing schemes. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 630–648, Warsaw, Poland, May 4–8, 2003. Springer-Verlag, Berlin, Germany.
28. Aggelos Kiayias and Moti Yung. Group signatures: Provable security, efficient constructions and anonymity from trapdoor-holders. Cryptology ePrint Archive, Report 2004/076, 2004. `http://eprint.iacr.org/`.
29. Aggelos Kiayias and Moti Yung. Group signatures with efficient concurrent join. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 198–214, Aarhus, Denmark, May 22–26, 2005. Springer-Verlag, Berlin, Germany.
30. Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 552–565, Gold Coast, Australia, December 9–13, 2001. Springer-Verlag, Berlin, Germany.

31. Hovav Shacham and Brent Waters. Efficient ring signatures without random oracles. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *Public Key Cryptography*, LNCS, pages 166–180. Springer, 2007.
32. Dawn Xiaodong Song. Practical forward secure group signature schemes. In *ACM CCS 01*, pages 225–234, Philadelphia, PA, USA, November 5–8, 2001. ACM Press.

A The Complete ESS Scheme

For reference purposes, in this section we describe the complete ESS construction in full detail. The construction follows exactly the outline given in the main body of the paper. Most of the technicalities are borrowed directly from the mesh signature scheme of [11], with which the ESS scheme shares many similarities.

A.1 Algebraic Representation of ESS Specifications

An ESS expression is specified as a monotone propositional formula or "access structure" Υ of a number of literals $L_1, ..., L_\ell$. As explained earlier, we need to transform Υ into an algebraic form suitable for manipulation in the exponents of pairing-based atomic signatures represented by the $L_1, ..., L_\ell$. Recall that Υ is generated by $\Upsilon ::= N$ in the grammar:

$$N ::= L_1 \mid ... \mid L_\ell \mid \geq_t\{N_1, ..., N_m\} \mid \wedge\{N_1, ..., N_m\} \mid \vee\{N_1, ..., N_m\} \ .$$

For technical reasons (see [11]), we need to consider one additional literal L_0 that does not correspond to any user key, whose meaning will be specified later. Let thus $\tilde{\Upsilon}$ be as above with $\ell + 1$ input literals $L_0, ..., L_\ell$.

We show how to convert the recursive expression of $\tilde{\Upsilon}$ into an algebraic representation consisting of a system of $\ell + 1$ linear equations in some number $\theta + 1 \leq \ell + 1$ of variables. The form of the polynomials and the number of variables will depend on the structure of $\tilde{\Upsilon}$. The transformation relies on Linear Secret Sharing Structures [24].

The principle is as follows. To each input symbol L_i we associate a degree-1 homogeneous polynomial $\pi_i = \sum_{j=0}^{\ell} y_{i,j}\, Z_j$, where the variables $Z_0, ..., Z_\ell$ are common to all polynomials and the integer coefficients $y_{i,j}$ are constant. The polynomials are such that, if the formula $\tilde{\Upsilon}$ is satisfied by setting some subset of symbols to $\top$ (*i.e.*, "true"), then the span of the corresponding polynomials will contain the pure monomial Z_0; conversely, any set of polynomials whose span contains the monomial Z_0 indicates a satisfying assignment.

The following algorithm computes such a representation from $\tilde{\Upsilon}$. Proceeding recursively, it assigns temporary polynomials to the interior nodes as it walks down the tree from the root to the leaves (*i.e.*, from the output gate to the input symbols):

1. Initialize a counter $k_c \leftarrow 0$.

The counter k_c is used for allocating new variables, so that each Z_{k+k_c} is always a "fresh" variable that is never used before or after in the algorithm.

2. Label the root node N_0 with the polynomial $\pi_{N_0} \leftarrow Z_0$.
3. Select a non-leaf node N with non-empty label $\pi_N \neq \emptyset$.
   (a) Denote by $N_1, ..., N_m$ the $m \geq 2$ children of N.
   (b) If N is $\vee\{N_1, ..., N_m\}$, then $\forall i = 1, ..., m$ let $\pi_{N_i} = \pi_N$.
   (c) If N is $\wedge\{N_1, ..., N_m\}$, then $\forall i = 1, ..., m$ let $\pi_{N_i} = \pi_N + \sum_{k=1}^{m-1} l_{i,k}\, Z_{k+k_c}$ where $l_{i,k} \in \mathbb{Z}$. The selection of $l_{i,k}$ is explained below.
   (d) If N is $\geq_t\{N_1, ..., N_m\}$, then $\forall i = 1, ..., m$ let $\pi_{N_i} = \pi_N + \sum_{k=1}^{t-1} l_{i,k}\, Z_{k+k_c}$ where $l_{i,k} \in \mathbb{Z}$.
   (e) Label each child N_i with the polynomial π_{N_i}.
   (f) Unlabel node N, *i.e.*, set $\pi_N \leftarrow \emptyset$.
   (g) Increment $k_c \leftarrow k_c + t - 1$ (using $t = 1$ and $t = m$ for $\vee$- and $\wedge$-gates).
   (h) Continue at Step 3 if an eligible node remains, otherwise skip to Step 4.
4. Let $\vartheta \leftarrow k_c$ and output the polynomials $(\pi_0, ..., \pi_\ell)$ associated with the leaf nodes $L_0, ..., L_\ell$. Each polynomial π_i is represented as a vector of coefficients $(y_{i,0}, ..., y_{i,\vartheta}) \in \mathbb{F}_p^{\vartheta+1}$ such that $\pi_i = \sum_{k=0}^{\vartheta} y_{i,k}\, Z_k$ is the result of the sequence of operations in Steps 3b, 3c and 3d.

We note that the only variables with non-zero coefficients in the output polynomials are $Z_0, ..., Z_\vartheta$, where $\vartheta = k_c$ is the final counter value and may be equal to or lesser than ℓ.

In Steps 3c and 3d, the coefficients $l_{i,k}$ must ensure that no linear relation exists in any set of π_i of size $< m$ or $< t$. (By construction, m or t of them will always be linearly dependent.) To ensure this property, we let $(l_{i,k})$ form a Vandermonde matrix in $\mathbb{F}_p^{m\times(m-1)}$ or $\mathbb{F}_p^{m\times(t-1)}$, *i.e.*, set $l_{i,k} = a_i^k$ for distinct $a_i \in \mathbb{F}_p$; independence follows from the existence of polynomial interpolation. We also require that $(l_{i,k})$ be constructed deterministically, so that anyone can verify that the π_i faithfully encode $\tilde{\Upsilon}$ simply by reproducing the process.

The following lemma states the equivalence between the recursive specification of $\tilde{\Upsilon}$ and its flattened representation.

Lemma 1. [24] *Let $\tilde{\Upsilon}$ be an arborescent monotone threshold circuit as defined, and $(\pi_0, ..., \pi_\ell)$ a flattened representation of it per the above algorithm. A minimal truth assignment $\chi : \{L_0, ..., L_\ell\} \to \{\bot, \top\}$ satisfies $\tilde{\Upsilon}(\chi(L_0), ..., \chi(L_\ell)) = \top$ if and only if there exist integer coefficients $(\nu_0, ..., \nu_\ell)$ such that,*

$$\sum_{i=0}^{\ell} \nu_i\, \pi_i = Z_0 \ , \qquad \textit{and} \qquad \forall i \ : \ \nu_i = 0 \iff \chi(L_i) = \bot \ .$$

Composite-order Groups. Since the signers who compute these representations work in a $\mathbb{G}$ without knowing the factorization $N = pq$, they will define all those polynomials over $\mathbb{Z}_N$ instead of $\mathbb{F}_p$. By the CRT, the preceding lemma will apply in both projections of order p and order q, although only the projection of order p will actually matter (since in the ESS scheme the coefficients will be applied exclusively on the generator $\mathbb{G}_p$ of order p).

A.2 Computational Blinding of Atomic Signatures

In the ESS signature scheme (yet to be described), we use both the polynomials $(\pi_0, ..., \pi_\ell)$ and the linear combination $(\nu_0, ..., \nu_\ell)$ from Lemma 1: the latter to create a signature, and the former to indicate how to verify it. However, since the linear coefficients ν_i reveal which of the L_i are true, they must be kept secret, as explained earlier. In the actual signature, these coefficients appear not as integers but as exponents of elements of $\mathbb{G}$, and so they are already computationally hidden in the sense of one-wayness. However, this is not enough and we need to take an extra step to ensure indistinguishability.

By Lemma 1 we know that $\sum_{i=0}^{\ell} \nu_i \, \pi_i = Z_0$, where each $\nu_i \in \mathbb{F}_p$ and each $\pi_i \in \mathbb{F}_p[Z_0, ..., Z_\vartheta]_1$. We hide the linear coefficients ν_i using random blinding terms $(h_0, ..., h_\ell)$ such that $\sum_{i=0}^{\ell} h_i \, \pi_i = 0$. Since $\sum_{i=0}^{\ell} (\nu_i + h_i) \, \pi_i = Z_0$, the blinded coefficients $\nu_i + h_i$ still bear witness that $\tilde{\Upsilon}(L_0, ..., L_\ell) = \top$. However, these witnesses have been rendered information-theoretically indistinguishable, because the distribution of $(\nu_0 + h_0, ..., \nu_\ell + h_\ell)$ is conditionally independent of the truth values of the L_i given that $\tilde{\Upsilon}(L_0, ..., L_\ell) = \top$.

The difficulty is that no scalar h_i will satisfy $\sum_{i=0}^{\ell} h_i \, \pi_i = 0$ when the π_i contain uninstantiated variables. However, given a specific set of π_i, it is easy to build h_i that have polynomial values.

1. Draw a random vector $\boldsymbol{s} = (s_1, ..., s_\ell) \in \mathbb{F}_p^\ell$ of scalar coefficients.
2. For $i = 1, ..., \ell$, define $h_i = -s_i \, \pi_0$, and set the remaining term $h_0 = \sum_{j=1}^{\ell} s_j \, \pi_j$.

In the actual scheme, these polynomials are evaluated "in the exponent" for unknown assignments to the Z_k, but regardless of their values we have $\sum_{i=0}^{\ell} h_i \, \pi_i = (\sum_{j=1}^{\ell} s_j \, \pi_j) \, \pi_0 + \sum_{i=1}^{\ell} (-s_i \, \pi_0) \, \pi_i = 0$, and so the blinding terms $(h_0, ..., h_\ell)$ meet our requirements.

Remark that the random vector $\boldsymbol{s}$ can be chosen independently of the π_i. This is important for the actual signature scheme, where the relevant polynomials will have coefficients that involve discrete logarithms not known explicitly (in addition to the Z_k being instantiated as discrete logarithms of random group elements). In spite of this, we will be able to select a suitable vector $\boldsymbol{s}$ and compute the blinding terms h_i "in the exponent".

Composite-order Groups. As before, since the users who compute these blinding polynomial work modulo $N = pq$ without knowing p, those users will have to perform the preceding steps in $\mathbb{Z}_N$ instead of $\mathbb{F}_p$. The projections modulo q will not matter since the blinding exponents will be applied to (various powers of) the generator $\mathbb{G}_p$ of order p.

In the same vein, and although our discussion suggests that the blinding will be perfect, this will only be true in the subgroup $\mathbb{G}_p$ of order p where the blinding takes place. However, atomic signatures do also also contain non-vanishing components of order q, which will not be blinded. This is of course by design, to ensure that the ESS tracing authority is able to see through the blinding. The effectiveness of the blinding with respect to outsiders will depend on the hardness of distinguishing random elements of order p from random elements of order $N = pq$, as we mentioned earlier.

A.3 ESS Algorithms

The full ESS signature scheme can now be described as follows.

Group Setup: Given a security parameter κ the group manager generates a bilinear group $\mathbb{G}$ and target group $\mathbb{G}_T$ of composite order $N = pq$, where p and q are two random primes of suitable size for the security parameter. Let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be the associated bilinear pairing. Let $g \in \mathbb{G}_p$ be a random point of order p in the bilinear group, and let $h \in \mathbb{G}$ be a random point of order N in the bilinear group.

Given an ESS expression size parameter λ, the group manager publishes a random string K that specifies $\lambda + 1$ elements $g_0, g_1, ..., g_\lambda$ in $\mathbb{G}_p$. It also publishes $\lambda + 1$ random triples $(A_{0,k},\ B_{0,k},\ C_{0,k}) \in \mathbb{G}_p^3$ for $k \in \{0, ..., \lambda\}$; the latter constitute a public verification key "in the sky" that does not correspond to any user signing key (it may be viewed as the manager's group signing key, although the manager signs anything with it).

The group manager also specifies a hash function $H : \{0,1\}^* \to \mathbb{F}_p$ from a collision-resistant family.

Lastly, the group manager selects a secret random $\gamma \in \mathbb{Z}_N$ and publishes $\mathit{\Gamma} = h^\gamma \in \mathbb{G}$. Notice that h and $\mathit{\Gamma}$ are the only two published elements that have non-null residues in $\mathbb{G}_q$.

User Enrollment: User $\#i$ may join the group either by receiving his credentials directly from the group manager, or by engaging in an interactive enrollment protocol that allows the user to keep his signing exponents secret from the manager.

In either case, upon being successfully admitted into the group, User $\#i$ obtains a random signing key $(x_i,\ y_i,\ z_i) \in (\mathbb{Z}_N^\times)^3$. The corresponding public certificate is a triple $(A_{i,k}, B_{i,k}, C_{i,k})$, where $A_{i,k} = g_k^{y_i/(\gamma+x_i)}$, $B_{i,k} = g_k^{1/(\gamma+x_i)}$, and $C_{i,k} = g_k^{z_i/(\gamma+x_i)}$, for each $k \in \{0, ..., \lambda\}$.

ESS Signing: On input the following ESS signature specification:

- formal atomic signature statements $\{\mathit{Msg}_i\}_{User_i}$ and boolean flags L_i, for $i = 1, ..., \ell$;
- a well-formed ESS formula $\mathit{\Upsilon}$ with ℓ boolean inputs; and a truth assignment $\chi : \{L_1, ..., L_\ell\} \to \{\perp, \top\}$ such that $\mathit{\Upsilon}(L_1, ..., L_\ell) = \top$;
- for each i such that $\chi(L_i) = \top$, a valid atomic signature in $\mathbf{G}$ for the the statement $\{\mathit{Msg}_i\}_{User_i}$, namely, a pair:

$$\sigma_i = \left(\ u_i = (\mathit{\Gamma} h^x)^{1/(y+w_i+t_i z)} \in \mathbb{G},\ \ t_i \in \mathbb{Z}_N\ \right) ,$$

where $w_i = Msg_i$ and (x, y, z) is the signing key for the signer mentioned in the specification clause $\{Msg_i\}_{User_i}$.

The signer firsts extends Υ into Υ' that involves the public key "in the sky":

1. Compute $Msg_0 = H(\{Msg_1\}_{User_1}, ..., \{Msg_\ell\}_{User_\ell}, \Upsilon)$ by hashing the full ESS formal specification, and associate the special literal L_0 to the supplemental formal clause $\{Msg_0\}_{User_0}$.
   The purpose of the hash is to "seal" the specification in order to prevent someone from weakening a finished ESS signature by OR-ing it with additional clauses (which would otherwise give a valid signature).
2. Construct $\tilde{\Upsilon} = L_0 \vee \Upsilon$, which is a well-formed ESS formula.
3. Set the truth assignment χ so that $\chi(L_0) = \bot$. We have to use the value "false" since we do not have an atomic signature for L_0.

The signer then builds the ESS signature from the ESS formula $\tilde{\Upsilon}$, the truth assignment χ, and the atomic signatures (u_i, t_i) known for such i that $\chi(L_i) = \top$, as follows:

4. Create a flattened representation of $\tilde{\Upsilon}$ and χ as discussed in Section A.1. Accordingly, let $\pi_0, ..., \pi_\ell \in \mathbb{Z}_N[Z_0, ..., Z_\vartheta]$ be public degree-1 multivariate polynomials that encode $\tilde{\Upsilon}$, and $\nu_0, ..., \nu_\ell \in \mathbb{Z}_N$ the secret scalar coefficients of a linear combination that expresses χ. Explicitly determine all the coefficients $y_{j,k} \in \mathbb{Z}_N$ in all polynomials $\pi_j = \sum_{k=0}^{\vartheta} y_{j,k}\, Z_k$.
5. Create a random blinding vector $\boldsymbol{s} = (s_1, ..., s_\ell) \in \mathbb{Z}_N^\ell$ as in Section A.2.
6. For all $i \in \{0, ..., \ell\} : \chi(L_i) = \bot$, pick $t_i \in \mathbb{Z}_N$ and fix $u_i = g^0 = 1 \in \mathbb{G}$.
7. For all $j = 0, ..., \ell$ and $k = 0, ..., \vartheta$, let $m_j = Msg_j$ and calculate,
$$v_{j,k} = \left(A_{j,k}\, B_{j,k}^{m_j}\, C_{j,k}^{t_j}\right)^{y_{j,k}} \quad , \qquad v_j = \prod_{k=0}^{\vartheta} v_{j,k} \; .$$
8. Compute, for $i = 1, ..., \ell$, and $k = 0, ..., \vartheta$, respectively,
$$S_i = {u_i}^{\nu_i}\, {v_0}^{-s_i} \quad , \qquad P_k = \prod_{j=1}^{\ell} {v_{j,k}}^{s_j} \; .$$
   (The value of any intervening u_i such that $\chi(L_i) = \bot$ is unimportant since then $\nu_i = 0$; this is true in particular for the fictitious User 0.)
9. Output the ESS signature, consisting of the formal ESS statement Υ and the tuple,
$$\sigma = (\ t_0,\ ...,\ t_\ell,\ S_1,\ ...,\ S_\ell,\ P_0,\ ...,\ P_\vartheta\ ) \ \in\ \mathbb{Z}_N^{\ell+1} \times \mathbb{G}^{\ell+\vartheta+1} \; .$$

ESS Verification: A fully qualified ESS signature consists of:

– $\ell + 1$ propositions $\{Msg_0\}_{User_0}, ..., \{Msg_\ell\}_{User_\ell}$ viewed as inputs to,
– an arborescent monotone threshold circuit $\tilde{\Upsilon} : \{\bot, \top\}^{\ell+1} \to \{\bot, \top\}$,
– an ESS signature $\sigma = (t_0, ..., t_\ell, S_1, ..., S_\ell, P_0, ..., P_\vartheta) \in \mathbb{Z}_N^{\ell+1} \times \mathbb{G}^{\ell+\vartheta+1}$.

To verify such a signature, the verifier proceeds as follows:

1. Ascertain that $\tilde{\Upsilon}(\top, \star, ..., \star) = \top$, extract from $\tilde{\Upsilon}(L_0, ..., L_\ell)$ the subcircuit $\Upsilon(L_1, ..., L_\ell)$ such that $\tilde{\Upsilon} = \Upsilon \vee L_0$, and verify that $Msg_0 = H(\{Msg_1\}_{User_1}, ..., \{Msg_\ell\}_{User_\ell}, \Upsilon)$.

2. Compute the representation $(\pi_0, ..., \pi_\ell)$ of the formula $\tilde{\Upsilon}$ by reproducing the deterministic conversion of Section A.1.
3. For $i = 0, ..., \ell$, determine the coefficients $y_{i,k} \in \mathbb{Z}_N$ of the polynomials $\pi_i = \sum_{k=0}^{\vartheta} y_{i,k}\, Z_k$.
4. For $i = 0, ..., \ell$ and $k = 0, ..., \vartheta$, retrieve $(A_{i,k}, B_{i,k}, C_{i,k})$ from the certificate of User i, let $m_i = Msg_i$, and calculate,

$$v_{i,k} = \left(A_{i,k}\, B_{i,k}^{m_i}\, C_{i,k}^{t_i}\right)^{y_{i,k}} \quad , \qquad\qquad v_i = \prod_{k=0}^{\vartheta} v_{i,k} \ .$$

5. Using the pairing, verify the $\theta + 1$ equalities, for all $k = 0, ..., \vartheta$,

$$e\left(P_k,\ v_0\right) \cdot \prod_{i=1}^{\ell} e\left(S_i,\ v_{i,k}\right) = \begin{cases} e(h,\ g_0) & \text{for } k = 0 \\ 1 \in \mathbb{G}_T & \text{for } k = 1, ..., \vartheta \end{cases} \ .$$

6. Accept the signature if and only if all $\vartheta + 1$ equalities hold in the target group $\mathbb{G}_T$.

Trapdoor Tracing: To trace a (fully qualified) ESS signature σ that passes the verification test, the tracing authority proceeds as follows:
1. For each $i = 1, ..., \ell$, compute $\bar{S}_i = (S_i)^p \in \mathbb{G}$, using the knowledge of p as a trapdoor.
2. For each i such that $\bar{S}_i = 1 \in \mathbb{G}$, say, "User i did not take part in σ".
3. For each i such that $\bar{S}_i \neq 1 \in \mathbb{G}$, say, "User i took part in signing σ".

B Optional Join Protocol

As discussed earlier, user enrollment into the ESS group can be done using an interactive protocol in order to preserve the strong exculpability property. This protocol is essentially the same as the protocol studied in the group signatures of [19], where more details can be found.

C Proof Details

C.1 Anonymity

Proof (of Theorem 1). We use a game hopping argument that starts with G_0, the real ESS anonymity game, and ends with G_1, a game that is identical to the previous one except that the generator g now generates all of $\mathbb{G}$, not just the subgroup $\mathbb{G}_p$. The advantage of the adversary in G_0 is ϵ. We denote his advantage in G_1 by ϵ_1.

In Lemma 2, we show that the games G_0 and G_1 must be indistinguishable to the adversary, unless the Decision Subgroup problem in $\mathbb{G}$ is easy.

$\mathcal{U}_\rangle$ $(\mathsf{upk}[i], \mathsf{usk}[i])$		$\mathcal{GM}$ (γ, gmsk)
$(y_i, z_i) \xleftarrow{R} \mathbb{Z}_p^2$, $C \leftarrow g^{y_i}$, $C' \leftarrow g^{z_i}$		
$U \leftarrow \begin{pmatrix} c = \mathsf{Ext\text{-}Commit}(y_i), \\ \mathsf{NIZKPEqDL}(c, C, g), \\ c' = \mathsf{Ext\text{-}Commit}(z_i), \\ \mathsf{NIZKPEqDL}(c', C', g), \end{pmatrix}$	$\xrightarrow{C,C',U}$	Verifies $(C, C') \in \mathbb{G}^2$, checks U
		$x_i \xleftarrow{R} \mathbb{Z}_p$, $A_i \leftarrow C^{1/(\gamma+x_i)}$
		$B_i \leftarrow g^{1/(\gamma+x)}$, $C_i \leftarrow C'^{1/(\gamma+x)}$
		$V \leftarrow e(C, g)/e(B_i, \Gamma)$
		$D \leftarrow e(B_i, g)$
	$\xleftarrow{A_i,B_i,C_i,W}$	$W \leftarrow \mathsf{NIZKPoKDL}(V, D)$
$V \leftarrow e(C, g)/e(B_i, \Gamma)$		
$X \leftarrow e(B_i, g)$		
Verifies $B_i \in \mathbb{G}$, $A_i = B_i^{y_i}$, $C_i = B_i^{z_i}$		
Checks W		
$\sigma \leftarrow Sign_{\mathsf{usk}[i]}(A_i, B_i, C_i)$	$\xrightarrow{\sigma}$	Checks S w.r.t. $\mathsf{upk}[i]$ and A
Checks that $A^{x+\gamma} \overset{?}{=} g^y$	$\xleftarrow{x_i}$	adds $(\mathsf{upk}[i], A_i, B_i, C_i, \sigma)$
i.e. $e(A_i, g)^{x_i} \cdot e(A_i, \Gamma) = e(g, g)^{y_i}$		

Fig. 1. Join Protocol

In lemma 3, we then appeal to an information-theoretic argument to show that signatures in G_1 enjoy unconditional and perfect anonymity, and thus that the adversary's advantage in that game must be nil.

The theorem follows from these two results. □

In this proof, we note that the details of the ESS anonymity game G_0 are not important, since the Decision Subgroup assumption makes ESS signatures computationally indistinguishable from the mesh signatures, which have perfect anonymity. The only important point in the ESS anonymity game is that no tracing queries are allowed.

Lemma 2. *For all τ'-time adversaries, $\epsilon - \epsilon_1 \leq 2\epsilon'$.*

Proof (of Lemma 2). Consider a distinguisher $\mathcal{D}$ for the subgroup decision problem. When presented with a subgroup decision challenge $(N, \mathbb{G}, \mathbb{G}_T, e, w)$, the algorithm $\mathcal{D}$ creates public parameters for the ESS scheme by first setting $g = g_p$ and $h = w$ and then choosing the remaining public parameters exactly as in the ESS scheme. It then transmits the public information to the adversary and plays the ESS anonymity game with it.

If w was drawn from the full group $\mathbb{G}$, then this is exactly the ESS game, which we called G_0. If w belongs in the subgroup $\mathbb{G}_p$, then all the elements now live in $\mathbb{G}_p$, and thus the scheme reduces to a mesh signature in the prime-order

$\mathbb{G}_p$; we call this game G_1. In either game, $\mathcal{D}$ can answer all the private key and signature queries, since it chose the parameters and knows all the secret keys.

At some point, the adversary will choose a message M, an ESS expression Υ, and two assignments θ_1 and θ_2 to the literals of Υ that cause it to evaluate to "true". The simulator $\mathcal{D}$ will pick a random $b \in \{1, 2\}$ and output an ESS signature created using atomic signatures as specified by the truth assignment θ_b. The adversary will then make a guess b' for the value of the bit b. If the adversary's guess is correct, that is, if $b' = b$, the distinguisher $\mathcal{D}$ returns 1, otherwise, $\mathcal{D}$ returns 0.

Denote by ϵ' the advantage of the $\mathcal{D}$ in the Decision Subgroup problem. Since in the Decision Subgroup challenge we have $\Pr[w \in \mathbb{G}_p] = \Pr[w \in \mathbb{G}] = 1/2$, we have:

$$\begin{aligned} \epsilon - \epsilon_1 &= \Pr[b = b' | w \in \mathbb{G}] - \Pr[b = b' | w \in \mathbb{G}_p] \\ &= \frac{\Pr[b = b', w \in \mathbb{G}]}{\Pr[w \in \mathbb{G}]} - \frac{\Pr[b = b', w \in \mathbb{G}_p]}{\Pr[w \in \mathbb{G}_p]} \\ &= \frac{\Pr[b = b']}{1/2} = 2\epsilon' \ . \end{aligned}$$

Since the total computation time τ' required of the distinguisher is dominated by the time τ consumed by the adversary, we also have $\tau' \approx \tau$. The claim follows. $\square$

Simulation with Dynamic Join. In the case where group enrollment is performed via the Dynamic Join protocol given in Section 3.7, the group manager normally cannot learn the users' private keys as part of the enrollment process. However, since the join protocol uses extractable commitments, in the simulation above the distinguisher is still able to recover all the users' secrets by setting the proper trapdoor into the commitment protocol's common reference string (or by extracting the secrets from queries to the commitment scheme's random oracles, if a random-oracle commitment scheme is preferred). In short, the simulator always knows all the secrets, and is thus able to answer all the queries. See [19] for the full details.

Lemma 3. *For all adversaries,* $\epsilon_1 = 0$.

Proof (of Lemma 3). The game G_1 is obtained by setting up the ESS system with both g and h in the subgroup $\mathbb{G}_p$ of order p. As a result, all group elements in the scheme end up in $\mathbb{G}_p$, and all their projections onto $\mathbb{G}_q$ vanishes. Such an ESS setup is therefore isomorphic to a regular mesh signature setup in any prime-order bilinear group $\mathbb{G}'$ of order p.

We then invoke the unconditional perfect anonymity of mesh signatures [11] to deduce that the signatures in the game G_1 must then be unconditionally and perfectly anonymous against unbounded adversaries.

The advantage of any adversary in the anonymity game G_1 is thus exactly $\epsilon_1 = 0$, as required. $\square$

In the preceding proof, we note that it may not be easy to compute the isomorphism between a regular mesh scheme in $\mathbb{G}'$ and the scheme of game G_1, without at least knowing the factorization of $N = pq$. However, it is obvious that the isomorphism exists, and that is all we need to deduce that the scheme in G_1 leaks no information about the signer.

C.2 Traceability

Proof (of Theorem 2). We begin with a game hopping argument, to show that the adversary's advantage in the real traceability game is essentially the same as in a synthetic game with two independent mesh signature schemes in the respective subgroups of order p and q. We will use this indistinguishability to conclude that a successful traceability attack in the ESS scheme implies a successful forgery attack against (at least) one of the mesh signature schemes, which is know from [11] is hard under the PolySDH assumption.

The starting game is G_0, the real ESS traceability game. The adversary's advantage in G_0 is $\epsilon_0 = \epsilon$.

The next game is G_1, a game that is identical to the previous one, except that the challenger selects a random g of order N instead of order p (that is, g will generate all of $\mathbb{G}$ instead of $\mathbb{G}_p$). The adversary's advantage in G_1 is ϵ_1.

In Lemma 4, we show that the games G_0 and G_1 are indistinguishable to the adversary, unless the Decision Subgroup problem in $\mathbb{G}$ is easy.

We remark that the signatures in G_1 are no longer traceable (that is, when $g \in \mathbb{G}$). The signature setup that the adversary is faced with in G_1 is identical to the mesh signature setup from [11], except that it is not defined in a prime-order bilinear group, but in the composite-order group $\mathbb{G}$. (That is, all the random generators are drawn from $\mathbb{G}$, and all the random exponents are drawn from $\mathbb{Z}_N$.)

The next game is G_2, a game where we once again change how the signature scheme is constructed. Instead of using an order-N "composite-order" mesh signature as in G_1, we use two simultaneous independent mesh signatures in the subgroups $\mathbb{G}_p$ and $\mathbb{G}_q$, that are combined in $\mathbb{G}$ using the Chinese Remainder Theorem (the simulator will know the factorization).

Precisely, the challenger in G_2 will set up two independent mesh signature schemes in the respective prime-order groups $\mathbb{G}_p$ and $\mathbb{G}_q$. We call them the p-scheme and the q-scheme. Then, for every integer $x_1 \in \mathbb{N}$ (or group element $X_1 \in \mathbb{G}$) that used to play some role R in the mesh scheme in G_1, we define a corresponding $x_2 \in \mathbb{N}$ (or $X_2 \in \mathbb{G}$) in G_2, uniquely defined by the two integers $x_2 \bmod p$ and $x_2 \bmod q$ (or group elements $X_2^q \in \mathbb{G}_p$ and $X_2^p \in \mathbb{G}_q$) that respectively play the same role R in the p-scheme and the q-scheme. Notice that it is easy to go back and forth between the decomposed and the composite representation using the CRT.

In Lemma 5, we show that the view of the adversary in games G_1 and G_2 in exactly the same.

To conclude the proof, it suffices to show that any successful traceability attack against the scheme in G_2 leads to a successful mesh signature forgery in the p-scheme. To do so, we need to construct a simulator for the game G_2, which will act as challenger in the ESS traceability game, and as opponent in the mesh signature unforgeability game.

We give the details of this final reduction in Lemma 6. The theorem follows. □

Lemma 4. *For all τ'-time adversaries, $\epsilon_0 - \epsilon_1 \leq 2\epsilon'$.*

Proof (of Lemma 4). The proof is analogous to that of Lemma 2 and is omitted. (The main difference is that we let h and not g be equal to the Decision Subgroup challenge value w.) □

Lemma 5. *For all adversaries, $\epsilon_1 - \epsilon_2 = 0$.*

Proof (of Lemma 5). It follows by inspection of the mesh scheme in [11] that the following two entities are exactly isomorphic: (1) a single random instance of the scheme in any bilinear group $\mathbb{G}_N$ of order of $N = pq$; and (2) two independent random instances of the scheme in any respective bilinear groups $\mathbb{G}_p$ and $\mathbb{G}_q$ of order p and q.

The isomorphism can furthermore be efficiently computed using the Chinese Remainder Theorem, if $\mathbb{G}_p$ and $\mathbb{G}_q$ happen to be subgroups of $\mathbb{G}_N$. The details of this computation are omitted. □

Lemma 6. *There exists a tight and computationally efficient reduction from the ESS traceability game G_2 and the mesh signature eUF-CMA unforgeability game in the p-scheme.*

Proof (of Lemma 6). The construction of the simulator is as follows. The simulator is given a set of mesh signature parameters in $\mathbb{G}_p$. It generates independent random mesh parameters in $\mathbb{G}_q$, combines the two using the CRT as outlined above, and gives the resulting set of composite-order mesh parameters to the adversary.

To enroll adversarially-controlled users, the simulator does the following steps. For the user of index i, the simulator will first select a random key $(a_i, b_i, c_i) \in \mathbb{Z}_N^3$, or recover it from the extractable commitments in case the enrollment protocol of Section 3.7 is employed. Using the CRT, the simulator separates each of those exponents into its projections modulo p and q, and retains the two resulting triples $(a_i^{(p)}, b_i^{(p)}, c_i^{(p)}) \in \mathbb{Z}_p^3$ and $(a_i^{(q)}, b_i^{(q)}, c_i^{(q)}) \in \mathbb{Z}_q^3$ as mesh private keys for user i in the p-scheme and the q-scheme. The simulator then constructs the corresponding public keys $(A_i^{(p)}, B_i^{(p)}, C_i^{(p)}) \in \mathbb{G}_p^3$ and $(A_i^{(q)}, B_i^{(q)}, C_i^{(q)}) \in \mathbb{G}_q^3$ in the p-scheme and the q-scheme, and combines them into a single public-key triple $(A_i, B_i, C_i) \in \mathbb{G}^3$, which it publishes in the group manifest.

The adversary may also make any number of enrollment queries for users that should remain in the control of the simulator. To process such a request, the simulator will proceed differently in the p-scheme and the q-scheme. In the q-scheme, the simulator picks a secret key and constructs the corresponding public key from it in the normal way. In the p-scheme, the simulator will ask the mesh challenger for a mesh public key (without learning the corresponding signing key). The simulator then uses the CRT to combine the two public keys into a single one, $(A_i, B_i, C_i) \in \mathbb{G}^3$, which it signs and publishes in the group description.

To answer ESS signature queries, the simulator proceeds as follows. A signature query is given as a propositional expression indicating what the signature is meant to represent, and a list of atomic signatures needed to construct it. The list may be incomplete, as the adversary is not supposed to provide atomic signatures on behalf of any of the simulator-controlled users. To answer the query, the simulator simply passes on the request to the p-scheme mesh challenger, and obtains a mesh signature in the p-scheme in return. The simulator also constructs a mesh signature in the q-scheme, which it can do explicitly since it knows all the secret keys in that scheme. With component-wise use of the CRT, The simulator combines the two prime-order mesh signatures into a single composite-order one, which it gives to the adversary as the requested ESS signature.

Eventually, the adversary outputs an ESS forgery that purportedly cannot be satisfactorily attributed to a coalition of adversarially-controlled users. The simulator simply projects down the forgery onto the subgroup of order p, and outputs the result as its own forgery in the mesh p-scheme.

Although the simulator does not know whether the forgery is a valid one, it is easy to see that this will be the case if and only if the adversary's ESS forgery is valid (of course, in the game G_2 no tracing is possible, but the adversary's view of G_2 is essentially the same as in G_0, where it is). $\square$