Full version of an extended abstract to be published in Advances in Cryptology—EUROCRYPT 2007, Springer-Verlag, 2007. Available online from the IACR Cryptology ePrint Archive as Report 2007/094. See also: `http://www.cs.stanford.edu/~xb/eurocrypt07b/`

Mesh Signatures

How to Leak a Secret with Unwitting and Unwilling Participants

Xavier Boyen

April 30, 2007

Abstract

We introduce the mesh signature primitive as an anonymous signature that borrows from ring signatures, but with added modularity and a much richer language for expressing signer ambiguity. The language can represent complex access structures, and in particular allows individual signature components to be replaced with modular certificate chains. As a result, withholding one's public key from view is no longer a shield against being named as a possible cosignatory; and hence, a mesh signature may be used as a ring signature substitute with compulsory enrollment.

We give an efficient construction based on bilinear maps in the common random string model. Our mesh signatures have linear size, achieve everlasting perfect anonymity, and as a special case induce the most efficient and first unconditionally anonymous ring signatures without random oracles or trusted setup authorities. We prove non-repudiation from a mild extension of the SDH assumption, which we introduce and justify meticulously.

1 Introduction

We introduce mesh signatures, which are similar in spirit and purpose to the ring signatures of Rivest, Shamir, and Tauman [27], and extend them in a few crucial ways.

Ring signatures are pseudonymous signatures that are issued in the name of a "ring" of users, and created by one of them without the participation of the others, in a way that preserves the instigator's anonymity. The canonical application is for an individual "to leak a secret" non-repudiably on behalf of a crowd. Technically, ring signatures can be viewed as a witness-indistinguishable disjunction of regular signatures, but because of this, only people who have previously published a verification key are eligible to be enrolled in such a crowd. Ring signatures can thus only ever implicate individuals who, by the very act of publishing their key, are proclaiming their consent.

Mesh signatures generalize this notion to mononote access structures representable as a tree, whose interior nodes are And, Or, and Threshold gates, and whose leaves are regular "atomic" signatures. The access structure can be satisfied using different combinations of atomic signatures; once created, the mesh signature will not reveal what particular subset was used. The atomic signatures may be "static" and reusable, as opposed to fresh; hence PKI certificates are eligible even if the mesh signer lacks the CA's signing key. Since furthermore the access structure is powerful enough to express disjunctions of certificate chains, we are no longer beholden to the prior publication of all the ring keys. As a toy example, suppose that *Alice* wants to implicate *Bob*,

[0] `xb@boyen.org` — Voltage Inc., Palo Alto

who may or may not have a verification key on record. *Alice* can still produce the following mesh signature,

$$\sigma = [\mathit{VK}_{\mathit{Alice}} \colon \mathit{Msg}_1] \ \text{ or } \ ([\mathit{VK}_{\mathit{CertAuth}} \colon \mathit{VK}_{\mathit{Bob}}] \ \text{ and } \ [\mathit{VK}_{\mathit{Bob}} \colon \mathit{Msg}_2]) \ .$$

All that *Alice* needs to create σ is her own private key and *CertAuth*'s certificate verification key. Even if *Bob* has published no verification key of his own, the mesh signature σ implicates him via the certificate $[\mathit{VK}_{\mathit{CertAuth}} \colon \mathit{VK}_{\mathit{Bob}}]$ that binds his name to the string $\mathit{VK}_{\mathit{Bob}}$. The certificate may be real or fake; no one can tell. Conversely, *Bob* could have created σ himself to implicate *Alice*, using his own private key and a valid certificate; although in this case her public key would have to be known to the verifier since it is not explicitly certified by σ. Another feature of mesh signatures is that they provide threshold gates, which makes it easy to scale such constructs as,

$$\sigma = \text{ 2-out-of-3 in } \{[\mathit{CEO} \colon \mathit{secret\text{-}memo}] \, , \, [\mathit{CFO} \colon \mathit{secret\text{-}memo}] \, , \, [\mathit{COO} \colon \mathit{secret\text{-}memo}]\} \ .$$

Threshold gates like this can feed or be fed from other gates as in the earlier example. The unconditional anonymity of mesh signatures guarantees that, as long as the signature σ is valid, there is no way to tell the true and false clauses apart in the formula expressed by σ.

We can immediately see how much more practical mesh signatures are than ring signatures: instead of requiring that each and everyone generate and publish their public key in a ring scheme, here we just need one trustworthy certificate authority (or preferably a few) to publish their keys in the mesh scheme—a natural demand to place on certificate authorities, though not on individuals.

To make the use of certificate chains truly believable, it is important that mesh signatures be "modular", or constructible non-interactively from reusable constituent atomic signatures, *i.e.*, without necessarily requiring the private keys. In our case, the atomic signatures are Boneh-Boyen short signatures [4] with independent keys in a common random bilinear context.

1.1 Related Work

The original ring signature primitive was defined in [27], to enable secret leaking that is at once authenticated (by a crowd) and anonymous (within the crowd). Whereas that construction [27] was based on trapdoor permutations, a number of alternatives have subsequently been proposed, based on bilinear pairings [6], discrete logarithms [21], factoring (Strong-RSA specifically) [16], or hybrids [1]; all these constructions are set in the random oracle model. Most have linear size in the ring membership count, except [16] which squeezes it all in constant size using accumulators in the random oracle model.

A number of general protocols bear similarities with our new primitive. Perhaps the first such scheme is an anonymous authentication protocol of [15] that supports access structures and can be turned into a signature using the Fiat-Shamir heuristic. Another is an interactive anonymous authentication protocol, called deniable ring authentication [26], that combines the anonymity of ring signatures with the non-transferability of deniable authentication [18], and supports threshold and access structures. Among specific constructions in the random oracle model, we note the distributed ring signatures of [22] which let coalitions of users cooperate in an interactive signing protocol, and the hierarchical identity-based ring signatures of [32], which add signer ambiguity to the notion of hierarchical identity-based signature. Additionally, we mention that mesh signatures could in principle be realized using signatures of knowledge [11], which allow the knowledge of a witness to an NP statement to serve as a signing key, in the common random string model.

Another related notion that has received much attention is that of group signatures, originally introduced in [12], and which also provides for the anonymous creation of signatures on behalf of a crowd. The main difference is that group signatures require the anonymity to be revocable by a group manager, who also controls enrollment into the group. Group membership is often immutable although this restriction has been relaxed in [9]. There exist efficient constant-size group signature schemes, with random oracles [5], from interactive assumptions [2], and in the standard model [8].

Efficient ring signatures constructions without random oracles have also been proposed recently, such as [14], [3], and [28]. The construction of [14] uses bilinear groups and is efficient but relies on a cumbersome assumption stated without justification. The results of [3] include an impractical scheme from non-interactive Zaps [17], but also two efficient constructions (based on [10] or [31] signatures) for rings of size two, and a discussion of security models for ring signatures.

Probably the most closely related to our work is the very recent ring scheme of [28] which can efficiently create linear-size ring signatures in the "trusted parameters" model; unforgeability is based on computational Diffie-Hellman, and anonymity on the decisional Subgroup [7] assumption. Because of the latter, the scheme requires a bilinear map in a group of composite order with a hidden factorization; such a group is set up explicitly by a central authority, which afterwards must erase the factorization to ensure anonymity. It may be possible to use ideas from [20] and base anonymity on the decisional Linear [5] assumption, which would no longer require secret-coin *trusted parameters* (TP) but only a public-coin *common random string* (CRS), as in our scheme; however anonymity would still remain computational. The main advantage of [28] over our ring scheme is that unforgeability rests on a weaker assumption.

2 Definitions and Security Models

Intuitively, a mesh signature is a non-interactive witness-indistinguishable proof that some monotone boolean expression Υ is true, where each input of Υ is notionally labeled with a key & message pair and is true only if the mesh signer is in possession of a valid atomic signature on the stated message under the stated key.

A mesh signature scheme should satisfy two security properties. First, it should be anonymous (ideally, unconditionally so), *i.e.*, it should not reveal what assignment to the inputs of Υ caused it to be satisfied. Second, it should be unforgeable, *i.e.*, the creation of a valid mesh signature must be predicated on the possession of a set of valid atomic signatures sufficient to satisfy Υ.

2.1 Recursive Mesh Signature Specification

We use ℓ to denote the number of atomic clauses in a mesh structure (in a ring signature, this would be equal to the number of users in the ring). Let Υ be the expression generated by the following grammar, with propositional-logic semantics, under the restriction that, for each $i = 1, ..., \ell$, the production $\textsc{expr} ::= L_i$ corresponding to the symbol L_i be used at most once (in other words, no L_i may appear more than once in the written expression of Υ):

$\textsc{expr} ::= L_1 \mid ... \mid L_\ell$	input symbols (these productions are single-use each)
$\mid\ \geq_t\{\textsc{expr}_1, ..., \textsc{expr}_m\}$	t-out-of-m threshold, with $1 < t < m$
$\mid\ \wedge\{\textsc{expr}_1, ..., \textsc{expr}_m\}$	m-wise conjunction, with $1 < m$
$\mid\ \vee\{\textsc{expr}_1, ..., \textsc{expr}_m\}$	m-wise disjunction, with $1 < m$

Equivalently, we call Υ an "arborescent monotone threshold circuit" with ℓ Boolean inputs $L_1, ..., L_\ell$ and one Boolean output denoted $\Upsilon(L_1, ..., L_\ell)$. It is apparent by induction that Υ is always a non-trivial monotone function of its inputs, and in particular $\Upsilon(\perp, ..., \perp) = \perp$ and $\Upsilon(\top, ..., \top) = \top$.

We use expressions of this form to state the meaning of mesh signatures. The signer specifies the circuit Υ, and assigns to each symbol L_j an atomic proposition $[\mathit{VK}\colon \mathit{Msg}]$ to convey the meaning: "This is *Msg* signed under *VK*." The mesh signature then simply expresses that $\Upsilon(L_1, ..., L_\ell) = \top$ holds for the stated interpretation of the L_i (without revealing their individual truth values). For the example in the introduction, $\Upsilon = L_1 \vee (L_2 \wedge L_3)$ where L_1 denotes $[\mathit{VK}_{\mathit{Alice}}\colon \mathit{Msg}_1]$, *etc.*

Multiplicity of Clauses. For simplicity, we require that the clauses of Υ share no public key, *i.e.*, for any two distinct $L_i = [\mathit{VK}_i\colon \mathit{Msg}_i]$ and $L_j = \left[\mathit{VK}_j\colon \mathit{Msg}_j\right]$ appearing in Υ, we have $\mathit{VK}_i \neq \mathit{VK}_j$. To mitigate this technicality, we expressly allow users to own multiple keys, which means that expressions Υ with a plurality of clauses involving the same user can easily be constructed.

By the same token, certificate authorities should be willing to translate any certificate issued under one of their keys into an equivalent certificate signed with any other of their own keys (or, perhaps, issue all the equivalent certificates at once).

2.2 Anonymity Model

The strongest notion of anonymity defined in [3], "anonymity against full key exposure", in the context of ring signatures, requires that the signer remain anonymous following full exposure of all the private keys, after their use. It is however a constrained notion of anonymity because the keys are not chosen by the adversary, and are only revealed *a posteriori*. We contend that, since the motivating application of ring and mesh schemes is to leak secrets, it is crucial that anonymity be *unconditional* and *everlasting*, subsequently to the exposure of all secrets, for the long-term peace of mind of the signer. We thus insist on perfect (*i.e.*, information theoretic) anonymity, even upon prior disclosure of the signer's and every user's secret keys.

Precisely, we require that the identity of the signer be statistically independent, conditionally on all public keys and the mesh formula, of all long-term secrets held by all parties in the system (and the random bits used to create those keys, though it makes no difference in our construction).

We exclude from the above requirement any ephemeral randomness used for signing, for the reason that there is no way to prevent the signer to prove willingly that she herself created a particular signature: revealing the ephemerals used to create a signature is but one way to do this. By contrast, the signer must be protected againt coerced disclosure, which is why independence from her long-term keys is crucial (and should not rely on the common random string's cleanliness).

Anonymity, Unlinkability, and Randomization. A mesh signature σ is most generally constructeed from a set of atomic signatures σ_i for the selected clauses $[\mathit{VK}_i\colon \mathit{Msg}_i]$ assigned the truth value $\top$. As we will see, mesh signatures are not unique, in part because of the "extrinsic" randomization due to the mesh signing process, but also because the atomic signatures themselves bring to the table their own "intrinsic" randomization (which will have to be faked for all clauses set to $\perp$). Therefore it is a legitimate concern to wonder how the intrinsic randomization associated with the atomic signatures interacts with the requirements of unconditional anonymity.

To fix ideas, suppose that a mesh signature σ contains a (valid or invalid) atomic signature on a clause $[\mathit{VK}_1\colon \mathit{Msg}_1]$ with intrinsic randomization t_1. Later, someone exposes an atomic signature σ_1 on the same clause and with the same randomization t_1. Is this evidence that σ_1 was used to construct σ, thereby putting the anonymity of the mesh signer in jeopardy? The answer is 'no', as long as the owner of VK_1 could have created σ_1 after seeing t_1. Conversely, if σ had been revealed subsequently to σ_1, one could not infer that σ_1 was used in the creation of σ, as long as σ could plausibly have been created *ex post facto* to match the exposed randomization.

In general, if the intrinsic randomization of the atomic signature can be chosen freely and is conveyed in the clear, then matching randomization merely implies awareness and not linkability, *i.e.*, it shows that the second signature was created with knowledge of the (randomization of the) first, but not that the two were actually created from each other, or by the same signer.

2.3 Unforgeability Model

The strongest notion of unforgeability defined in [3], "unforgeability with respect to insider corruption", for ring signatures, gives the adversary the ability to corrupt users dynamically, and include its own public keys when making ring signature queries. Since the point of mesh signatures is to implicate uncooperative users, it is judicious to allow them to choose their keys maliciously.

However, as a compromise for unconditional anonymity, we relax the fully dynamic corruption model into an enhanced static one, in which the honest users are static and created ahead of time by a challenger, and the corrupted users are under the full control of an adversary who can bring them to life dynamically. We also need to specify what constitutes a valid forgery. For ring signatures, a forgery is any signature by a ring without adversarially controlled users. For mesh signatures, however, this would be overly restrictive, since it would exclude such forgeries as,

$$\Upsilon = ([U_1 : m_1] \wedge [U_3 : m_3]) \vee ([U_2 : m_2] \wedge [U_4 : m_4]) \ ,$$

where U_1 and U_2 are honest users, and U_3 and U_4 are corrupted. Since Υ nominally entails $\Upsilon' = [U_1 : m_1] \vee [U_2 : m_2]$, a forger who signs Υ lacking the imprimatur of both U_1 and U_2 should be deemed successful. The same reasoning would continue to apply if the forger legitimately obtained an atomic signature on $[U_3 : m_3]$ even though U_3 were honest. We capture these circumstances by deeming admissible any forgery on a statement Υ if there exists a well-formed (and thus non-trivial) formula Υ' that contains no clause under the forger's control and such that $\Upsilon \Rightarrow \Upsilon'$.

To see where this comes from, for all corrupted users and all issued atomic signatures, let us set the corresponding literal $L_i \leftarrow \top$, which is the most that the adversary can do in legitimacy. If Υ then evaluates to $\top$, the forgery is inadmissible; otherwise, Υ will reduce to some well-formed formula Υ' that contains non-adversarial clauses exclusively. Hence, the existence of Υ' simply demands that Υ be unsatisfiable by the volition of the adversarial users alone. We distill all of this into the following existential unforgeability game, and define the adversary's advantage as the probability of outputting an admissible valid forgery.

Challenger setup: the challenger designates a number ℓ of public keys, corresponding to the honest target users under the challenger's control.

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

Adversary setup: the adversary reveals polynomially many public keys, one at a time, corresponding to the users under the adversary's control.

Mesh signature queries: the adversary makes up to q mesh signature queries on specifications Υ_j that involve at least one honest user.

Atomic signature queries: The adversary also makes up to q atomic signatures queries on clauses $\left[VK_i : Msg_j\right]$ for every honest user.

The challenger accepts or responds to each request before accepting the next one. The q mesh queries and the $q\,\ell$ atomic queries may be interleaved arbitrarily.

Signature forgery: the adversary produces a forged signature whose specification Υ satisfies $\forall j, \Upsilon \neq \Upsilon_j$ and implies a well-formed formula Υ' on the honest users, *i.e.*, $\Upsilon(L_1, ..., L_\ell, ...) \Rightarrow \Upsilon'(L_1, ..., L_\ell)$, obtained by setting to "true" ($\top$) every literal L_i whose clause $[VK_i\colon Msg_i]$ involves an adversarial key or matches an atomic query.

We define the adversary's advantage at mesh unforgeability as its probability of winning the game.

In the adversary setup, one must recognize that the adversary might try to claim some of the challenger's keys as its own (perhaps re-randomized to make it less obvious). Since the same is possible in the real world, and is readily detectable by the challenger, we take no step to forbid it.

Mesh Unforgeability *vs.* Ring Unforgeability. The mesh security model allows the forger to make arbitrary atomic signature queries on behalf of the honest users: this is because mesh signatures must be constructible from any satisfying set of atomic signatures (such as PKI certificates) without requiring the private keys.

For ring signatures, atomic signature queries are superfluous, and we can obtain a tighter proof of security without them, mainly because we reduce the number of queries from $(\ell + 1)\, q$ to just q. Hence, we define existential unforgeability for ring signatures as for mesh signatures, but without atomic signature queries (also, regular signatures can always be emulated using rings of size one).

3 Framework and Computational Assumption

We write $\mathbb{F}_p$ for the finite field of prime order p, and $\mathbb{F}_p^\times = \mathbb{F}_p \setminus \{0\}$ for its multiplicative group of order $p-1$. Let a bilinear context $\mathbf{G} = (p, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_t, g, \hat{g}, \mathbf{e})$, where $\mathbf{e} : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_t$ is a pairing [25]. We use the "hat-notation" (as in $\hat{g}$) to indicate that an element belongs to $\hat{\mathbb{G}}$ rather than $\mathbb{G}$.

3.1 Review of the SDH Assumption

The complexity assumption we shall need is inspired from the Strong Diffie-Hellman assumption proposed in [4], which we now review. The q-SDH problem in a (bilinear) group $\mathbb{G}$ is stated:

(Original SDH) Given elements $g,\ g^\alpha,\ g^{\alpha^2},\ ...,\ g^{\alpha^q} \in \mathbb{G}$, choose $w \in \mathbb{F}_p$ and output $(w,\ g^{1/(\alpha+w)})$.

The SDH assumption then posits that the q-SDH problem above is intractable for $q = \mathrm{O}(\mathrm{poly}(\kappa))$. What makes this assumption special is that the problem admits not one but exponentially many "independent" solutions, which are all equally hard to find. Hence the modified q-SDH problem:

(Modified SDH) Given $g,\ g^\alpha \in \mathbb{G}$ and $q-1$ pairs $(w_j,\ g^{1/(\alpha+w_j)})$, output another $(w,\ g^{1/(\alpha+w)})$.

It is known from [4] that if the original q-SDH problem is hard, then so is the modified problem.

Although the SDH problem statement does not require a bilinear group, it is because the bilinear map provides an efficient Decision Diffie-Hellman procedure [23] that the correctness of an SDH solution can be decided openly. Specifically, given g and g^α, deciding whether $(w,\ u) = (c,\ g^{1/(\alpha+w)})$ amounts to checking the equality $\mathbf{e}(u,\ \hat{g}^\alpha\, \hat{g}^w) = \mathbf{e}(g,\ \hat{g})$, basically a DDH a test that anyone can perform from public information. The short signature scheme of [4] relies on this.

3.2 Poly-SDH : for Better Use of the Pairing

The verifiability of SDH solutions with a simple DDH test suggests that more general assumptions could be made, based on the observation that the pairing is a powerful tool that can be used to decide more complex relations that are not efficiently reducible to DDH. For example, a natural generalization of the SDH problem is that of finding ℓ pairs $(w_i,\ u_i = g^{r_i/(\alpha+w_i)})$ for $i = 1, ..., \ell$, such that $\sum_{i=1}^{\ell} r_i = 1 \pmod p$. Purported solutions can then be verified using the equation,

$$\prod_{i=1}^{\ell} \mathbf{e}(u_i,\ \hat{g}^{\alpha}\, \hat{g}^{w_i}) = \mathbf{e}(g,\ \hat{g}) \ . \tag{1}$$

Clearly, when $\ell = 1$, this is identical to the SDH problem. For larger values of ℓ, the adversary is given to spread the exponent inversion task across multiple pairs, by means of linear combination.

Unfortunately, for $\ell > 1$, the problem is in fact trivial, because Equation (1) admits spurious solutions that do not require the solver to know the secret α and invert the exponent: for example, for $\ell = 2$ the solution $w_1 = 1,\ u_1 = g,\ w_2 = 0,\ u_2 = g^{-1}$ satisfies the equality regardless of α.

To remedy the preceding problem, we change the solver's task slightly, and ask that the ℓ pairs to be output involve ℓ independent secrets $\alpha_1, ..., \alpha_\ell$ that appear once each, *i.e.*, find,

$$\left(w_i,\ u_i = g^{\frac{r_i}{\alpha_i + w_i}} \right) \ : \ i = 1, ..., \ell \ , \qquad \text{s.t.} \quad \sum_{i=1}^{\ell} r_i = 1 \pmod p \ .$$

To decide whether a solution $((w_1,\ u_1),\ ...,\ (w_\ell,\ u_\ell))$ to the new problem is correct, one needs, besides the generators g and $\hat{g}$, the ℓ group elements $(\hat{g}_1,\ ...,\ \hat{g}_\ell) = (\hat{g}^{\alpha_1},\ ...,\ \hat{g}^{\alpha_\ell})$. The verification equation is then,

$$\prod_{i=1}^{\ell} \mathbf{e}(u_i,\ \hat{g}_i\, \hat{g}^{w_i}) = \mathbf{e}(g,\ \hat{g}) \ . \tag{2}$$

Notice that (1) is a special case of (2) where $\alpha_1 = ... = \alpha_\ell = \alpha$; however, for the security of the assumption it is important that the α_i be independently and uniformly distributed. Despite the added variables, we stress that Equation (2) is no more expensive to verify.

Based on the previous observations, the (q, ℓ)-Poly-SDH problem can be informally stated as:

(Poly-SDH) Given $g, g^{\alpha_1}, ..., g^{\alpha_\ell} \in \mathbb{G}$ and $q\,\ell$ pairs $(w_{i,j},\ g^{1/(\alpha_i+w_{i,j})})$ for $1 \leq i \leq \ell$ and $1 \leq j \leq q$, choose fresh $w_1,\ ...,\ w_\ell \in \mathbb{F}_p$ and output ℓ pairs $(w_i,\ g^{r_i/(\alpha_i+w_i)})$ such that $\sum_{i=1}^{\ell} r_i = 1$.

The α_i and $w_{i,j}$ in the instance are drawn from a uniform distribution. The w_i and r_i are chosen by the respondent. We require that $\forall i, \forall j, w_i \neq w_{i,j}$, lest the task be easy. The exponents r_i need not be revealed, since Equation (2) can establish that a solution is correct, and thus that $\sum_i r_i = 1$, without having to see the r_i.

We have chosen to state the (q, ℓ)-Poly-SDH problem in a form analogue to Modified SDH, rather than Original SDH. There are several justifications for this:

- the modified form results in a weaker assumption (as Original SDH implies Modified SDH);
- it has a clear input/output symmetry which simplifies the security reductions;
- its instances are more concisely stated when more than one iterator is needed (i and j);
- the modified problem form is impervious to a (benign) generic analysis described in [13], which relies on the availability of g, g^{α}, and g^{α^d} for certain d, as in Original SDH instances.

The reason why there are no undesirably easy solutions to the (q, ℓ)-Poly-SDH problem will become apparent as we prove generic hardness in Section 3.3. See also Appendix A.1 for formal definitions.

3.3 Generic Hardness of Poly-SDH

We now take some time to explain why the Poly-SDH assumption based on Equation (2) is plausible, unlike our first attempt from Equation (1) that was so easily broken. We give a heuristic argument based on the impossibility of efficient generic attacks. Specifically, we show that finding a solution to the (q,ℓ)-Poly-SDH problem will require, on expectation, $\Omega(\sqrt{p/q\,\ell})$ generic group operations.

The generic group model [29] assumes the lack of any structure beyond that of an (Abelian) cyclic group, restricting all manipulations on group elements to the group operation and its inverse (*i.e.*, multiplication and division if the group is written multiplicatively). In the bilinear version of the model [4], one can also compute a pairing $\mathbf{e}:\mathbb{G}\times\hat{\mathbb{G}}\to\mathbb{G}_t$, as well as an isomorphism $\psi:\hat{\mathbb{G}}\to\mathbb{G}$ (for "type-1" and "type-2" contexts) and its inverse $\psi^{-1}:\mathbb{G}\to\hat{\mathbb{G}}$ (for "type-1" only).

Let us assume that $\mathbb{G}=\hat{\mathbb{G}}$, which only makes the attack easier. Recall that the Poly-SDH instance furnishes g, g^{α_1}, ..., g^{α_ℓ}, and a large number of pairs $(w_{i,j},\ u_{i,j}=g^{1/(\alpha_i+w_{i,j})})$. Based on this information, the attacker must output ℓ pairs $(w_i,\ u_i=g^{r_i/(\alpha_i+w_i)})$ such that $\sum_i r_i=1$, and where w_i is distinct from all $w_{i,j}$ with the same index i.

First, notice that the pairing $\mathbf{e}$ is useful to verify a solution, but not really to find one. This is because $\mathbf{e}$ ranges into $\mathbb{G}_t$, and once we have landed in $\mathbb{G}_t$ we can never leave it. Also, ψ and ψ^{-1} just model the identity function since we have already assumed that $\mathbb{G}=\hat{\mathbb{G}}$. We can thus focus on multiplication and division in the multiplicative group $\mathbb{G}$ of prime order p.

Next, observe that all the group elements that can be created from g, $\{g^{\alpha_i}\}$, and $\{g^{1/(\alpha_i+w_{i,j})}\}$ are of the form $g^{\frac{\pi(\alpha_1,\dots,\alpha_\ell)}{\Delta}}$, where $\pi\in\mathbb{F}_p[\alpha_1,\dots,\alpha_\ell]_{q\ell+1}$ is any multivariate polynomial in $\alpha_1,\dots,\alpha_\ell$ of total degree at most $q\,\ell+1$, and where Δ is the common denominator $\Delta=\prod_{i=1}^{\ell}\prod_{j=1}^{q}(\alpha_i+w_{i,j})$. We need to produce ℓ elements $u_i=g^{r_i/(\alpha_i+w_i)}$ and the corresponding w_i. Our task is thus to find ℓ polynomials π_1, ..., $\pi_\ell\in\mathbb{F}_p[\alpha_1,\dots,\alpha_\ell]_{q\ell+1}$ such that $\pi_i/\Delta=r_i/(\alpha_i+w_i)$ for some $\sum_i r_i=1$, *i.e.*, such that,

$$\sum_{i=1}^{\ell}(\alpha_i+w_i)\,\pi_i=\Delta=\prod_{i=1}^{\ell}\prod_{j=1}^{q}(\alpha_i+w_{i,j})\ .$$

We show that there can be no such polynomials π_i using a linear change of variable. For all $i=1,\dots,\ell$ and $j=1,\dots,q$, we define $\alpha'_i=\alpha_i+w_i$ and $w'_{i,j}=w_{i,j}-w_i$. Notice that all $w'_{i,j}\neq 0$. Our new task becomes to find ℓ polynomials π'_1, ..., π'_ℓ of degree $\le q\,\ell+1$ in the variables $\alpha'_1,\dots,\alpha'_\ell$, such that,

$$\sum_{i=1}^{\ell}\alpha'_i\,\pi'_i=\Delta=\prod_{i=1}^{\ell}\prod_{j=1}^{q}(\alpha'_i+w'_{i,j})\ .$$

Clearly, all the monomials in the left-hand side have degree in $\alpha'_1,\dots,\alpha'_\ell$ at least 1. On the other hand, all $w'_{i,j}$ are non-zero, so the right-hand side yields a non-vanishing independent (degree-0) term equal to $\prod_i\prod_j w'_{i,j}=\prod_i\prod_j(w_{i,j}-w_i)\neq 0$, which is a contradiction.

The contradiction shows that the equations above cannot be satisfied identically in $\mathbb{F}_p[\alpha'_1,\dots,\alpha'_\ell]$ or $\mathbb{F}_p[\alpha_1,\dots,\alpha_\ell]$, which proves that the polynomials π'_i and thus π_i cannot exist. A standard argument then shows that the equations can only be satisfied in $\mathbb{F}_p$ for certain assignments of α_1, ..., $\alpha_\ell\in\mathbb{F}_p$: the polynomial roots. Since the α_i are chosen at random, we can bound the probability of hitting those roots. We find that, if $q\,\ell<\mathrm{O}(\sqrt[3]{p})$, it will take $q_G=\Omega(\sqrt{\epsilon\,p/q\,\ell})$ operations to solve (q,ℓ)-Poly-SDH with probability ϵ in generic groups of order p.

We give a precise theorem and a complete proof based on this argument in Appendix A.4.

3.4 Pluri-SDH : a Weaker Assumption

Although we will need the Poly-SDH assumption to prove security of mesh signatures, ring signatures can be based on a slightly weaker assumption, due to the lack of atomic signature queries. Recall that in the (q,ℓ)-Poly-SDH problem, we are given ℓ generators g^{α_i} as well as ℓ series of q solution pairs $(w_{i,j},\ u_{i,j} = g^{1/\alpha_i + w_{i,j}})$. Our weaker assumption is similar, except that we only give out a single series of solution pairs, conventionally for an extra generator of index $i = 0$.

We define the $(q,\ell,1)$-Pluri-SDH problem as a relaxed version of $(q,\ell+1)$-Poly-SDH:

(Pluri-SDH) Given generators $g,\ g^{\alpha_0},\ \ldots,\ g^{\alpha_\ell} \in \mathbb{G}$ and q pairs $(w_{0,j},\ g^{1/(\alpha_0+w_{0,j})})$ for $1 \le j \le q$, choose fresh $w_0,\ \ldots,\ w_\ell \in \mathbb{F}_p$ and output $\ell+1$ pairs $(w_i,\ g^{r_i/(\alpha_i+w_i)})$ such that $\sum_{i=0}^{\ell} r_i = 1$.

See also Appendix A.2 for formal definitions, including that of the (q,ℓ,ℓ')-Pluri-SDH problem which is stated in an obvious way for $\ell' \ge 1$.

Generic Complexity. Regarding generic complexity, we can show that for $q\,\ell < \mathrm{O}(\sqrt[3]{p})$, a generic algorithm can solve the $(q,\ell,1)$-Pluri-SDH problem with constant probability ϵ in a generic group of prime order p only by performing $q_G = \Omega(\sqrt{\epsilon\, p/q})$ generic group operations on expectation.

A precise theorem for $(q,\ell,1)$-Pluri-SDH and (q,ℓ,ℓ')-Pluri-SDH is given in Appendix A.5.

3.5 Comparing SDH with Pluri-SDH and Poly-SDH

An interesting fact about the $(q,\ell,1)$-Pluri-SDH problem in generic bilinear groups is that it is quantitatively as difficult as the (modified) q-SDH problem: in particular, the generic lower bounds are essentially the same as those found in [4], and do not strongly depend on ℓ. In other words, allowing the opponent to make ℓ-wise linear combinations has little adverse effect on generic security, provided that care has been taken to structure the problem to rule out all of the trivial solutions. A similar comparison can be made for the full (q,ℓ)-Poly-SDH problem, except that the relevant benchmark here is the $q\ell$-SDH problem. Although we appear to lose a factor ℓ in the number of allowed queries with respect to SDH, it will be a wash if the security reduction of interest allows ℓ times as many queries, which will be the case of our mesh unforgeability simulator.

The main difference between SDH and Pluri-SDH/Poly-SDH is thus not one of hardness. It is that the former is useful in any "Gap-DH" group where the Diffie-Hellman problem has a decision procedure, while Pluri-SDH and Poly-SDH require a group with an actually computable pairing (or at least an oracle for comparing products of pairings) in order to verify its solutions.

4 Special Case : Ring Signatures

We first describe a ring signature based on Pluri-SDH as a special case of our technique. It is more efficient than other provably secure ring signature schemes without random oracles, and the first of those schemes to offer unconditional anonymity. It is set in the "public-coin" common random string model, *i.e.*, without trusted parameters or setup authority.

Initialization: Given a security parameter κ and a public random string $K \in \{0,1\}^{\mathrm{poly}(\kappa)}$, the parties generate from K a common bilinear instance $\mathbf{G} = (p,\ \mathbb{G},\ \hat{\mathbb{G}},\ \mathbb{G}_t,\ g,\ \hat{g},\ \mathbf{e}) \leftarrow \mathcal{G}(1^\kappa; K)$

and a collision-resistant hash function $H : \{0,1\}^* \to \mathbb{F}_p$ shared by all. Since $\mathbf{G}$ has prime order and no hidden structure, it can safely be generated from public coins.

The string K is also used to generate three random elements $\hat{A}_0$, $\hat{B}_0$, and $\hat{C}_0$ in $\hat{\mathbb{G}}$. These elements define a public verification key "in the sky" whose matching signing key is undefined.

For notational convenience, we suppose for now that the isomorphism $\psi : \hat{\mathbb{G}} \to \mathbb{G}$ is efficiently computable in the instance $\mathbf{G}$, and we let $A_0 = \psi(\hat{A}_0)$, $B_0 = \psi(\hat{B}_0)$, and $C_0 = \psi(\hat{C}_0)$ in $\mathbb{G}$. This temporary restriction will be lifted later in this section.

Key generation: To create a key pair, User $\#i$ draws a triple $(a_i,\, b_i,\, c_i) \in (\mathbb{F}_p^\times)^3$ as signing key, and posts $(A_i,\, B_i,\, C_i,\, \hat{A}_i,\, \hat{B}_i,\, \hat{C}_i) = (g^{a_i},\, g^{b_i},\, g^{c_i},\, \hat{g}^{a_i},\, \hat{g}^{b_i},\, \hat{g}^{c_i}) \in \mathbb{G}^3 \times \hat{\mathbb{G}}^3$ as verification key.

In case $\psi : \hat{\mathbb{G}} \to \mathbb{G}$ is easy to compute, users publish only $(\hat{A}_i,\, \hat{B}_i,\, \hat{C}_i)$ to avoid redundancy.

Ring signature: To create a ring signature on message $m_1, ..., m_\ell \in \mathbb{F}_p$ attributed to a ring of ℓ users, any member of the ring would proceed as follows. W.l.o.g., suppose that the signer is User $\#\ell$ in the ring $R = (1, ..., \ell)$. The signer selects $2\,\ell + 1$ random integers $s_0, s_1, ..., s_{\ell-1}, t_0, t_1, ..., t_\ell \in \mathbb{F}_p$, and outputs the signature,

$$\sigma = \left(g^{s_0},\ ...,\ g^{s_{\ell-1}},\ \left(g \cdot \prod_{i=0}^{\ell-1} (A_i\, B_i^{m_i}\, C_i^{t_i})^{-s_i} \right)^{\frac{1}{a_\ell + b_\ell\, m_\ell + c_\ell\, t_\ell}},\ t_0,\ ...,\ t_\ell \right) \in \mathbb{G}^{\ell+1} \times \mathbb{F}_p^{\ell+1} ,$$

where $m_1, ..., m_\ell$ are the messages to be signed, and $m_0 = H((1, m_1), ..., (\ell, m_\ell))$, a collision-resistant hash of the statement expressed by the signature.

Ring verification: To verify a signature $\sigma = (S_1,\ ...,\ S_\ell,\ t_1,\ ...,\ t_\ell)$, test the equality,

$$\prod_{i=0}^{\ell} \mathbf{e}(S_i,\ \hat{A}_i\, \hat{B}_i^{m_i}\, \hat{C}_i^{t_i}) = \mathbf{e}(g,\ \hat{g}) ,$$

where $R = (1, ..., \ell)$ is the signature ring, $m_1, ..., m_\ell$ are the messages being signed, and $m_0 = H((1, m_1), ..., (\ell, m_\ell))$.

Consistency of the algorithms is readily verified. Note that the scheme is trivially modified to force all messages $m_1, ..., m_\ell$ to be the same, as in the traditional definition of ring signatures.

The purpose of including in the final signature a collision-resistant hash m_0 of the ring and all the messages, ostensibly binding m_0 to the public key "in the sky", is to prevent outsiders from appending new components to an existing signature, which would otherwise give an easy forgery (though perhaps a rather benign one). The second reason is that the key "in the sky" is useful in the security proof, and lets us rely on a weaker assumption.

4.1 No Trusted Setup

We emphasize that the public string K has no hidden structure whatsoever, and can be drawn by public coin flipping, or in any way that does not grant anyone undue advantage in the computation of discrete logarithms in $\mathbf{G}$ or of atomic signatures under the random key "in the sky" (either of which would defeat unforgeability but not anonymity).

The absence of secret coins marks the distinction between the common random string (CRS) model and the more stringent trusted parameters (TP) model. In the former, the CRS has no special structure and can be drawn in the open, or derived from public observations of natural phenomena. In the latter, the TP must be crafted in a special way from secret coins by some trusted setup authority, which must exist in the real world, and must voluntarily erase that privileged information once it has completed its business, while convincing the other players that it is not cheating.

To reiterate, no trusted setup agent, either centralized or distributed, is needed in our system.

4.2 Anonymity

Independently of setup assumptions, our ring signatures have irrevocable or everlasting, perfect, unconditional anonymity (*i.e.*, with forward security against coerced disclosure of the long-term signing keys, and the randomness that created them, of all users in the system).

Theorem 1. *The ring signature has everlasting perfect anonymity.*

Proof. See Appendix B.1 □

4.3 Unforgeability

We then have existential unforgeability in the common random string model based on our computational assumption. More precisely, we can give two alternative reductions: one establishes security in the *ring* forgery game provided that the $(q, \ell, 1)$-Pluri-SDH problem is hard; the other proves security in the more demanding *mesh* forgery game from the hardness of $(q, \ell+1)$-Poly-SDH. Here, we recall from Section 2.3 that a mesh forger can also make atomic signature queries to the honest users in addition to mesh (or ring) queries, whereas a ring forger makes no atomic queries.

We now state the ring result, which is the most appropriate in the context of ring signatures. In Appendix B.2, however, we shall state and prove the stronger result instead, because parts of that proof will be reused when proving security of the full mesh scheme of Section 5.

Theorem 2. *The ring signature is existentially unforgeable under an adaptive attack, against a static adversary that makes no more than q adaptive ring signature queries, provided that the $(q, \ell, 1)$-Pluri-SDH assumption holds in $\mathbf{G}$, in the common random string model.*

Proof. See Appendix B.2 □

4.4 Withholding the Isomorphism

Since the most general types of bilinear instance $\mathbf{G}$ may fail to provide both an efficient isomorphism $\psi : \hat{\mathbb{G}} \to \mathbb{G}$ and an efficient sampling procedure in $\hat{\mathbb{G}}$, it is useful to modify the ring scheme in order to relax these requirements. Although we can typically rely on one or the other [19], it is easy to eliminate both requirements at once in the following way.

- First, we redefine the random key "in the sky" to consist just of A_0, B_0, and C_0, to be sampled directly in $\mathbb{G}$ from the common random seed K (skipping $\hat{\mathbb{G}}$ altogether).
- Next, we modify the group element of index 0 in the signature, $\hat{g}^{s_0} \in \hat{\mathbb{G}}$ replacing $g^{s_0} \in \mathbb{G}$. The signature becomes, *e.g.*, with User $\#\ell$ as the signer: $\sigma = (\hat{S}_0, ..., S_\ell, t_0, ..., t_\ell) =$

$$\left(\hat{g}^{s_0},\ g^{s_1},\ ...,\ g^{s_{\ell-1}},\ \left(g \cdot \prod_{i=0}^{\ell-1} (A_i\, B_i^{m_i}\, C_i^{t_i})^{-s_i} \right)^{\frac{1}{a_\ell + b_\ell\, m_\ell + c_\ell\, t_\ell}},\ t_0,\ ...,\ t_\ell \right) \in \hat{\mathbb{G}} \times \mathbb{G}^\ell \times \mathbb{F}_p^{\ell+1} ,$$

- Last, we exchange the arguments under the pairing of index 0 and amend the verification equation into,

$$\mathbf{e}(A_0\, B_0^{m_0}\, C_0^{t_0},\ \hat{S}_0) \cdot \prod_{i=1}^{\ell} \mathbf{e}(S_i,\ \hat{A}_i\, \hat{B}_i^{m_i}\, \hat{C}_i^{t_i}) = \mathbf{e}(g,\ \hat{g}) \ .$$

It is easy to see that the security theorems continue to hold in the modified ring signature scheme. On the one hand, anonymity is unconditional and thus insensitive to the existence of some efficient algorithm for ψ or for sampling in $\hat{\mathbb{G}}$. On the other hand, unforgeability relies no more on the presence of such algorithms than on their absence, as an inspection of the proof would show.

4.5 Removing the Key "In the Sky"

A (tenuous) argument can be made that having a public key "in the sky" entails a stronger flavor of CRS than the mere sharing of a bilinear instance $\mathbf{G}$ and a collision-resistant hash function H.

The crux of the argument is that, for someone who controls the CRS, it is much easier to implant a trapdoor into the public key VK_0 than to prepare $\mathbf{G}$ for the subsequent efficient computation of discrete logarithms: the former can be done by constructing VK_0 from an explicit signing key (as the simulator does in the unforgeability proof), whereas the latter might involve the infeasible pre-computation of an exponential-size lookup table for the baby-step giant-step algorithm in $\mathbf{G}$. A counterargument is that, if the CRS is truly random, then all of this is equally hard for everyone. Either way, both flavors of the CRS model seem much more palatable than the TP model.

Notwithstanding, the key "in the sky" can be eliminated altogether if we remove from the ring signature the component $[VK_0 \colon h]$ where $h = H((VK_1, m_1), ..., (VK_\ell, m_\ell))$, change one of the remaining clauses from $[VK_i \colon m_i]$ into $[VK_i \colon 0\|h]$, and change all the other remaining clauses from $[VK_i \colon m_i]$ into $[VK_i \colon 1\|m_i]$; it does not matter which clause is selected to host h. The signing mechanism is unchanged, except for the removal of S_0 and t_0, and the fact that one of the m_i cedes to $0\|h$ and the others to $1\|m_i$. Unforgeability then follows from hardness of (q, ℓ)-Poly-SDH.

5 General Case : Mesh Signatures

We now describe our mesh signature scheme, based on the Poly-SDH assumption. We proceed in stages: we first define a few useful notions, which we then use to describe the actual system.

5.1 Flattened Mesh Representation

Recall that a mesh signature is characterized by an expression Υ generated by the grammar,

$$\begin{aligned} \Upsilon &::= N \\ N &::= L_1 \mid ... \mid L_\ell \mid \geq_t\{N_1, ..., N_m\} \mid \wedge\{N_1, ..., N_m\} \mid \vee\{N_1, ..., N_m\} \ . \end{aligned}$$

To harmonize the notation with the scheme description, we need to consider an extra literal L_0 whose meaning is unimportant for now, and let $\tilde{\Upsilon}$ be as above with $\ell + 1$ input literals $L_0, ..., L_\ell$.

We show how to convert the recursive expression of $\tilde{\Upsilon}$ into a representation as a list of $\ell + 1$ polynomials in $\ell + 1$ variables (or fewer, depending on the structure of $\tilde{\Upsilon}$), akin to Linear Secret Sharing Structures [24, 30].

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 coefficients $y_{i,j}$ are elements of $\mathbb{F}_p$. The polynomials are such that, if the formula $\tilde{\Upsilon}$ is satisfied by setting some subset of symbols to $\top$, 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 $k_\mathrm{c} \leftarrow 0$.

   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 $\pi_{N_i} = \pi_N + \sum_{k=1}^{m-1} l_{i,k}\, Z_{k+k_\mathrm{c}}$ where $l_{i,k} \in \mathbb{F}_p$. 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 $\pi_{N_i} = \pi_N + \sum_{k=1}^{t-1} l_{i,k}\, Z_{k+k_\mathrm{c}}$ where $l_{i,k} \in \mathbb{F}_p$.
   (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 $k_\mathrm{c} \leftarrow k_\mathrm{c} + t - 1$ (using $t = 1$ for an $\vee$-gate, and $t = m$ for an $\wedge$-gate).
   (h) Continue at Step 3 if an eligible node remains, otherwise skip to Step 4.

4. Let $\vartheta \leftarrow k_\mathrm{c}$ 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 $\vartheta = k_\mathrm{c}$ 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 shows the equivalence between the recursive specification of $\tilde{\Upsilon}$ and its flattened representation. It is adapted from a classic result [24] for Linear Secret Sharing Structures, and proven by induction on the structure of $\tilde{\Upsilon}$. We refer to the literature [30] for further details.

Lemma 3. [24] *Let $\tilde{\Upsilon}$ be an arborescent monotone threshold circuit, 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 in $\mathbb{F}_p$ coefficients $\nu_0, ..., \nu_\ell$ such that,*

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

5.2 Information-Theoretic Blinding

In the 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 3: 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. In the actual signature, these coefficients appear not as integers but as exponents of elements of $\mathbb{G}$, and are thus already computationally hidden; however, this is not enough and we need to take an extra step to ensure information-theoretic hiding.

By Lemma 3 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 $\vec{s} = (s_1, ..., s_\ell) \in \mathbb{F}_p^\ell$ 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 $\vec{s}$ 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 $\vec{s}$ and compute the blinding terms h_i "in the exponent".

5.3 Construction

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

Initialization: Given a security parameter κ and a public random string $K \in \{0,1\}^{\mathrm{poly}(\kappa)}$, all participants generate the common bilinear instance $\mathbf{G} = (p, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_t, g, \hat{g}, \mathbf{e}) \leftarrow \mathcal{G}(1^\kappa; K)$. Here, we require that the accompanying isomorphism $\psi : \hat{\mathbb{G}} \to \mathbb{G}$ be efficiently computable.

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

Given a mesh size parameter λ, the string K then specifies $\lambda + 1$ elements $\hat{g}_0, \hat{g}_1, ..., \hat{g}_\lambda$ in $\hat{\mathbb{G}}$, on which the efficient algorithm for ψ can be applied to obtain the images $g_0, g_1, ..., g_\lambda$ in $\mathbb{G}$.

Additionally, K defines $\lambda+1$ random triples $(\hat{A}_{0,k},\ \hat{B}_{0,k},\ \hat{C}_{0,k}) \in \hat{\mathbb{G}}^3$ for $k \in \{0,...,\lambda\}$; these elements together constitute a public verification key "in the sky" with no known signing key. We define $A_{0,k} = \psi(\hat{A}_{0,k})$, $B_{0,k} = \psi(\hat{B}_{0,k})$, $C_{0,k} = \psi(\hat{C}_{0,k})$, in $\mathbb{G}$, again easy to compute.

Key generation: To create a key pair, User $\#i$ draws a triple $(a_i,\ b_i,\ c_i) \in (\mathbb{F}_p^\times)^3$ as signing key. User $\#i$ computes for each $k \in \{0,...,\lambda\}$ the triple $(\hat{A}_{i,k},\ \hat{B}_{i,k},\ \hat{C}_{i,k}) = (\hat{g}_k^{a_i},\ \hat{g}_k^{b_i},\ \hat{g}_k^{c_i}) \in \hat{\mathbb{G}}^3$, and lets these $3\,(\lambda+1)$ group elements constitute his or her verification key.

For simplicity, we write $(A_{i,k},\ B_{i,k},\ C_{i,k}) = (\psi(\hat{A}_{i,k}),\ \psi(\hat{B}_{i,k}),\ \psi(\hat{C}_{i,k})) = (g_k^{a_i},\ g_k^{b_i},\ g_k^{c_i}) \in \mathbb{G}^3$, which anyone can compute from the verification key of User $\#i$ thanks to ψ.

Mesh signature: Consider the following mesh signature prototype information:

- ℓ statements $[\mathit{VK}_i\colon \mathit{Msg}_i]$, assumed w.l.o.g. to involve the public keys of Users $\#1,...,\ell$, and whose propositional truth values are denoted by the literals L_i for $i = 1,...,\ell$.
- an arborescent monotone threshold circuit Υ where each literal $L_1,...,L_\ell$ is an input leaf; and an assignment $\chi : \{L_1,...,L_\ell\} \to \{\bot,\top\}$ that satifies $\Upsilon(L_1,...,L_\ell) = \top$;
- $\forall i = 1,...,\ell$ such that $\chi(L_i) = \top$, a valid Boneh-Boyen signature in $\mathbf{G}$, given as a pair,

$$\left(\ u_i = g^{\frac{1}{a+b\,w+c\,t_i}},\ t_i\ \right)\ , \qquad\qquad \text{for some} \quad t_i \in \mathbb{F}_p\ ,$$

  where $w = \mathit{Msg}_i$ and (a,b,c) is the signing key for the statement $[\mathit{VK}_i\colon \mathit{Msg}_i]$.
- Optionally, a prescribed value $t_i \in \mathbb{F}_p$ for any or all indices i such that $\chi(L_i) = \bot$.

To create a mesh signature based on the preceding data, the signer firsts extends Υ into a new specification that involves the verification key "in the sky":

1. Hash the public mesh specification to get $\mathit{Msg}_0 = H([\mathit{VK}_1\colon \mathit{Msg}_1],...,[\mathit{VK}_\ell\colon \mathit{Msg}_\ell],\Upsilon)$, and implicitly associate the literal L_0 to the clause $[\mathit{VK}_0\colon \mathit{Msg}_0]$.
2. Construct $\tilde{\Upsilon} = L_0 \vee \Upsilon$, a well-formed arborescent monotone threshold circuit.
3. Extend χ so that $\chi(L_0) = \bot$, as we lack the corresponding atomic signature.

The signer then builds the mesh signature from the circuit $\tilde{\Upsilon}$, the 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 5.1. Accordingly, let $\pi_0,...,\pi_\ell \in \mathbb{F}_p[Z_0,...,Z_\vartheta]$ be public degree-1 multivariate polynomials that encode $\tilde{\Upsilon}$, and $\nu_0,...,\nu_\ell \in \mathbb{F}_p$ the secret scalar coefficients of a linear combination that expresses χ. Explicitly determine all the coefficients $y_{j,k} \in \mathbb{F}_p$ in all polynomials $\pi_j = \sum_{k=0}^{\vartheta} y_{j,k}\, Z_k$.
5. Create a random blinding vector $\vec{s} = (s_1,...,s_\ell) \in \mathbb{F}_p^\ell$ as in Section 5.2.
6. $\forall i \in \{0,...,\ell\} : \chi(L_i) = \bot$, randomly draw $t_i \in \mathbb{F}_p$ or use the optionally prescribed value, and arbitrarily fix $u_i = g^0 = 1 \in \mathbb{G}$.
   (Recall that for $\chi(L_i) = \top$, the t_i and u_i are supplied with the atomic signatures.)
7. For all $j = 0,...,\ell$ and $k = 0,...,\vartheta$, calculate,

$$v_{j,k} = \left(A_{j,k}\, B_{j,k}^{m_j}\, C_{j,k}^{t_j}\right)^{y_{j,k}}\ , \qquad v_j = \prod_{k=0}^{\vartheta} v_{j,k}\ , \qquad \text{setting} \quad m_j = \mathit{Msg}_j\ .$$

   (Notice that if we instantiate $Z_k = \mathrm{dlog}_g(g_k)$, then we get $v_j = g^{(a_j+b_j\,m_j+c_j\,t_j)\,\pi_j}$.)

8. Compute, for $i = 1, ..., \ell$, and $k = 0, ..., \vartheta$, respectively,

$$S_i = {u_i}^{\nu_i} \, {v_0}^{-s_i} \; , \qquad\qquad P_k = \prod_{j=1}^{\ell} {v_{j,k}}^{s_j} \; .$$

   (The value of any intervening u_i such that $\chi(L_i) = \perp$ is unimportant since then $\nu_i = 0$; this is true in particular for the user "in the sky" of index 0.)

9. Output the mesh signature, consisting of the statement Υ and the tuple,

$$\sigma = (\; t_0, \; ..., \; t_\ell, \; S_1, \; ..., \; S_\ell, \; P_0, \; ..., \; P_\vartheta \;) \;\in\; \mathbb{F}_p^{\ell+1} \times \mathbb{G}^{\ell+\vartheta+1} \; .$$

Mesh verification: A fully qualified mesh signature package consists of:

- a list of $\ell + 1$ propositions $[\mathit{VK}_0 \colon \mathit{Msg}_0], ..., [\mathit{VK}_\ell \colon \mathit{Msg}_\ell]$ viewed as inputs to,
- an arborescent monotone threshold circuit $\tilde{\Upsilon} : \{\perp, \top\}^{\ell+1} \to \{\perp, \top\}$,
- a mesh signature $\sigma = (t_0, ..., t_\ell, S_1, ..., S_\ell, P_0, ..., P_\vartheta) \in \mathbb{F}_p^{\ell+1} \times \mathbb{G}^{\ell+\vartheta+1}$.

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 sub-circuit $\Upsilon(L_1, ..., L_\ell)$ such that $\tilde{\Upsilon} = \Upsilon \vee L_0$, and verify that $\mathit{Msg}_0 = H([\mathit{VK}_1 \colon \mathit{Msg}_1], ..., [\mathit{VK}_\ell \colon \mathit{Msg}_\ell], \Upsilon)$.
2. Recompute the polynomials $(\pi_0, ..., \pi_\ell)$ representating the formula $\tilde{\Upsilon}$ by reproducing the deterministic conversion of Section 5.1.
3. For $i = 0, ..., \ell$, determine the coefficients $y_{i,k} \in \mathbb{F}_p$ of the polynomials $\pi_i = \sum_{k=0}^{\vartheta} y_{i,k} \, Z_k$.
4. For $i = 0, ..., \ell$ and $k = 0, ..., \vartheta$, retrieve $(\hat{A}_{i,k}, \hat{B}_{i,k}, \hat{C}_{i,k})$ from the key VK_i, and calculate,

$$\hat{v}_{i,k} = \left(\hat{A}_{i,k} \, \hat{B}_{i,k}^{m_i} \, \hat{C}_{i,k}^{t_i} \right)^{y_{i,k}} \; , \qquad \hat{v}_i = \prod_{k=0}^{\vartheta} \hat{v}_{i,k} \; , \qquad \text{setting} \quad m_i = \mathit{Msg}_i \; .$$

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

$$\mathbf{e}\,(P_k, \; \hat{v}_0) \cdot \prod_{i=1}^{\ell} \mathbf{e}\,(S_i, \; \hat{v}_{i,k}) = \begin{cases} \mathbf{e}(g, \; \hat{g}_0) & \text{for } k = 0 \\ 1 & \text{otherwise} \end{cases} \; .$$

6. Accept the signature as valid if and only if all $\vartheta + 1$ preceding equalities hold in $\mathbb{G}_t$.

(Optional) **Probabilistic check:** Mesh signatures can be verified using fewer total pairings, at the cost of some additional random bits and exponentiations. In the same setting as above, it suffices to replace the end of the verification algorithm from Step 5 onward by the following:

5′. Using the pairing, for $d_0 = 1$ and random $d_1, ..., d_\vartheta \in \mathbb{F}_p$, verify the single equality,

$$\mathbf{e}\left(\prod_{k=0}^{\vartheta} P_k^{d_k}, \; \hat{v}_0 \right) \cdot \prod_{i=1}^{\ell} \mathbf{e}\left(S_i, \; \prod_{k=0}^{\vartheta} \hat{v}_{i,k}^{d_k} \right) = \mathbf{e}(g, \; \hat{g}_0) \; .$$

6′. Accept the signature as valid if and only if the preceding equality holds in $\mathbb{G}_t$.

5.4 Security

We state the correctness, anonymity, and unforgeability theorems for the mesh scheme. A corollary to the latter is also given, based on a weaker assumption, for the case where only a subset of the honest users are willing to answer atomic signature queries (*e.g.*, certificate authorities).

Theorem 4. *The mesh signature is consistent.*

Proof. For any list of public polynomials $\pi_0, ..., \pi_\ell$ and secret coefficients $\nu_0, ..., \nu_\ell$ that respectively encode per Lemma 3 a well-formed mesh specification $\tilde{\Upsilon}$ and an assignment χ that satisfies it, we need to show that a signature created by the above algorithm will be accepted by the same. A straightforward sequence of substitutions in the scheme description shows this to be the case. □

Theorem 5. *The mesh signature has everlasting perfect anonymity.*

Proof. See Appendix C.1 □

Theorem 6. *The mesh signature is existentially unforgeable under an adaptive attack, against a static adversary that makes no more than q mesh signature queries, and no more than q atomic signature queries to each of the ℓ honest users, adaptively, provided that the $(q, \ell + 1)$-Poly-SDH assumption holds in $\mathbf{G}$, in the common random string model.*

Proof. See Appendix C.2 □

Corollary 7. *The mesh signature is existentially unforgeable under an adaptive attack, against a static adversary that makes no more than q mesh signature queries, and no more than q atomic signature queries to each of ℓ' among a total of $\ell + \ell'$ honest users, adaptively, provided that the $(q, \ell, \ell' + 1)$-Pluri-SDH assumption holds in $\mathbf{G}$, in the common random string model.*

5.5 Optimizations for Shorter Keys and CRS

All verification keys (including the one "in the sky") can be shortened to 2/3 their original length, because the private keys' b_i are always known to the simulators and do not contribute to security: we can set $b_i = 1$ and omit the publication of $\hat{B}_{i,k} = \hat{g}_k^{b_i} = \hat{g}_k$. This holds in the ring scheme, too.

We can furthermore compress the key "in the sky" to just two elements of $\hat{\mathbb{G}}$, if we observe that for $\tilde{\Upsilon} = \Upsilon \vee L_0$ the encoding algorithm of Section 5.1 always gives $\pi_0 = Z_0$, *i.e.*, $y_{0,0} = 1$ and $y_{0,k} = 0$ for $k \neq 0$. This means that the tuples $(\hat{A}_{0,k}, \hat{B}_{0,k}, \hat{C}_{0,k})$ for $k \neq 0$ are in fact never used. Since it is safe to set $\hat{B}_{0,0} = \hat{g}$, the key "in the sky" can shrink to a mere random pair $(\hat{A}_{0,0}, \hat{C}_{0,0})$.

6 Conclusion

We have introduced mesh signatures as a generalization of ring signatures with a richer language for expressing signer ambiguity. Mesh signatures scale to large crowds with many co-signers and independent certificate authorities; they can even implicate unwilling individuals who, by withholding their ring public key, would have otherwise remained out of reach. Because in principle mesh signatures require neither trusted setup nor centralized authorities, they provide a credible answer to the question of how to leak a secret authoritatively.

We have constructed a simple and practical mesh signature scheme in prime order bilinear groups, that achieves everlasting unconditional anonymity, and existential unforgeability in the common random string model, without trusted setup authority. To obtain this result, we introduced a new complexity assumption, which we prove sound in the generic model; it is in the spirit of the SDH assumption, but better exploits the group structure of the values computed by pairing. Incidentally, we obtain a very efficient and the first unconditionally anonymous ring signature without random oracles as a special case of our construction.

Acknowledgements

The author wishes to express his gratitude to Anna Lysyanskaya, Steven Galbraith, and the anonymous referees of Eurocrypt 2007 for many valuable comments.

References

[1] Masayuki Abe, Miyako Ohkubo, and Koutarou Suzuki. 1-out-of-n signatures from a variety of keys. In *Proceedings of AsiaCrypt 2002*, volume 2501 of *LNCS*, pages 415–32. Springer, 2002.

[2] 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/`.

[3] Adam Bender, Jonathan Katz, and Ruggero Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. In *Proceedings of TCC 2006*, LNCS. Springer, 2006.

[4] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In *Advances in Cryptology—EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 56–73. Springer, 2004.

[5] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *Advances in Cryptology—CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, 2004.

[6] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Advances in Cryptology—EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 416–32. Springer, 2003.

[7] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In *Proceedings of TCC 2005*, Lecture Notes in Computer Science. Springer-Verlag, 2005.

[8] Xavier Boyen and Brent Waters. Full-domain subgroup hiding and constant-size group signatures. In *Public Key Cryptography—PKC 2007*, volume 4450 of *LNCS*, pages 1–15. Springer, 2007.

[9] Jan Camenisch and Anna Lysyanskaya. Signature schemes with efficient protocols. In *Proceedings of SCN 2002*, LNCS. Springer, 2002.

[10] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *Advances in Cryptology—CRYPTO 2004*, volume 3152 of *LNCS*. Springer, 2004.

[11] Melissa Chase and Anna Lysyanskaya. Signature of knowledge. In *Advances in Cryptology—CRYPTO 2006*, volume 4117 of *LNCS*. Springer, 2006.

[12] David Chaum and Eugène van Heyst. Group signatures. In *Advances in Cryptology—EUROCRYPT 1991*, volume 547 of *LNCS*, pages 257–65. Springer, 1991.

[13] Jung Hee Cheon. Security analysis of the strong Diffie-Hellman problem. In *Advances in Cryptology—EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 1–13. Springer, 2006.

[14] Sherman S. M. Chow, Victor K.-W. Wei, Joseph K. Liu, and Tsz Hon Yuen. Ring signatures without random oracles. In *Proceedings of AsiaCCS 2006*, pages 297–302. ACM Press, 2006.

[15] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Advances in Cryptology—CRYPTO 1994*, volume 839 of *LNCS*, pages 174–87. Springer, 1994.

[16] Yevgeniy Dodis, A. Kiayias, Antonio Nicolosi, and Victor Shoup. Anonymous identification in ad hoc groups. In *Advances in Cryptology—EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 609–26. Springer, 2004.

[17] Cynthia Dwork and Moni Naor. Zaps and their applications. In *Proceedings of FOCS 2000*, pages 542–552. IEEE Press, 2000.

[18] Cynthia Dwork, Moni Naor, and Amit Sahai. Concurrent zero-knowledge or the timing model for designing concurrent protocols. *Journal of the ACM*, 51(6):851–98, 2004.

[19] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. Cryptology ePrint Archive, Report 2006/165, 2006. `http://eprint.iacr.org/2006/165/`.

[20] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Non-interactive Zaps and new techniques for NIZK. In *Advances in Cryptology—CRYPTO 2006*, LNCS. Springer, 2006.

[21] Javier Herranz and Germán Sáez. Forking lemmas for ring signature schemes. In *Proceedings of IndoCrypt 2003*, volume 2904 of *LNCS*, pages 266–79. Springer, 2003.

[22] Javier Herranz and Germán Sáez. New distributed ring signatures for general families of signing subsets. Cryptology ePrint Archive, Report 2004/377, 2004. `http://eprint.iacr.org/`.

[23] Antoine Joux and Kim Nguyen. Separating decision Diffie-Hellman from computational Diffie-Hellman in cryptographic groups. *Journal of Cryptology*, 16(4), 2003.

[24] Mauricio Karchmer and Avi Wigderson. On span programs. In *Annual Conference on Structure in Complexity Theory*, 1993.

[25] Victor Miller. The Weil pairing, and its efficient calculation. *Journal of Cryptology*, 17(4), 2004.

[26] Moni Naor. Deniable ring authentication. In *Advances in Cryptology—CRYPTO 2002*, volume 2442 of *LNCS*, pages 481–98. Springer, 2002.

[27] Ron Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In *Proceedings of AsiaCrypt 2001*, volume 2248 of *LNCS*, pages 552–65. Springer, 2001.

[28] Hovav Shacham and Brent Waters. Efficient ring signatures without random oracles. In *Public Key Cryptography—PKC 2007*, volume 4450 of *LNCS*. Springer, 2007.

[29] Victor Shoup. Lower bounds for discrete logarithms and related problems. In *Advances in Cryptology—EUROCRYPT 1997*, volume 1233 of *LNCS*. Springer, 1997.

[30] Marten van Dijk. A linear construction of secret sharing schemes. *Designs, Codes and Cryptography*, 12(2):161–201, 1997.

[31] Brent Waters. Efficient identity-based encryption without random oracles. In *Advances in Cryptology—EUROCRYPT 2005*, volume 3494 of *LNCS*. Springer, 2005.

[32] Victor K. Wei and Tsz Hon Yuen. (Hierarchical identity-based) threshold ring signatures. Cryptology ePrint Archive, Report 2006/193, 2006. `http://eprint.iacr.org/`.

A The Poly-SDH and the Pluri-SDH Assumptions

We gave informal definitions of the Poly-SDH and Pluri-SDH assumptions in Section 3.2 and 3.4. Next, we give concrete and asymptotic definitions of the Poly-SDH and Pluri-SDH assumptions, whose generic hardness we prove in Appendices A.4 and A.5.

A.1 Formal Poly-SDH Definitions

Our formal statement of the (q, ℓ)-Poly-SDH problem that applies to canonical bilinear groups of all types, whether $\mathbb{G} = \hat{\mathbb{G}}$ or $\mathbb{G} \neq \hat{\mathbb{G}}$. In general, most of the group elements that appear in the problem instance will belong in $\mathbb{G}$, and the solution will also be asked in that group. Several elements must also be given in $\hat{\mathbb{G}}$, mainly to enable the pairing-based verification of the solutions; but rather than require the isomorphism $\psi : \hat{\mathbb{G}} \to \mathbb{G}$ to be efficiently computable and let the user do the translation, we shall give these elements in both groups explicitly.

Definition 8. In a bilinear context $\mathbf{G} = (p, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_t, g, \hat{g}, \mathbf{e})$, the (q, ℓ)-Poly-SDH problem is:

> Given, $\forall i = 1, ..., \ell$, $\forall j = 1, ..., q$, $\quad g^{\alpha_i} \in \mathbb{G}$, $\hat{g}^{\alpha_i} \in \hat{\mathbb{G}}$, and $(w_{i,j},\ g^{\frac{1}{\alpha_i + w_{i,j}}}) \in \mathbb{F}_p \times \mathbb{G}$,
> output $\forall i$, $(w_i,\ g^{\frac{r_i}{\alpha_i + w_i}})$, subject to: $\sum_{i=1}^{\ell} r_i = 1 \pmod p$, $\forall i, \forall j,\ w_i \neq w_{i,j}$.

The advantage of an algorithm $\mathcal{A}$ in solving the (q, ℓ)-Poly-SDH problem is defined as,

$$\mathbf{Adv}_{\mathcal{A}}^{\text{PolySDH}} = \Pr\left[\mathcal{A}\begin{pmatrix} g,\ (g^{\alpha_i})_{1\le i\le \ell},\ (\ w_{i,j},\ g^{\frac{1}{\alpha_i + w_{i,j}}}\ )_{\substack{1\le i\le \ell\\ 1\le j\le q}} \\ \hat{g},\ (\hat{g}^{\alpha_i})_{1\le i\le \ell} \end{pmatrix} = \Big(\ w_i,\ g^{\frac{r_i}{\alpha_i + w_i}}\ \Big)_{1\le i\le \ell} : \sum_{i=1}^{\ell} r_i = 1\right]$$

The probability is over the random choice of generators $g \in \mathbb{G} \setminus \{1\}$ and $\hat{g} \in \hat{\mathbb{G}} \setminus \{1\}$, of exponents α_1, ..., $\alpha_\ell \in \mathbb{F}_p^\times$, of integers $w_{i,j} \in \mathbb{F}_p \setminus \{-\alpha_i\}$, and the random bits consumed by $\mathcal{A}$. (Nevertheless, we allow g and/or $\hat{g}$ to be given externally, as it is convenient in type-1 and type-2 bilinear contexts to take $g = \psi(\hat{g})$ under a fixed isomorphism ψ.)

Definition 9. We say that the (q, ℓ, t, ϵ)-Poly-SDH assumption holds in $\mathbf{G} = (p, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_t, g, \hat{g}, \mathbf{e})$ if no t-time and ϵ-advantagerandomized algorithm solves the (q, ℓ)-Poly-SDH problem in $\mathbf{G}$.

A.2 Formal Pluri-SDH Definitions

As in the previous case, we formally define the $(q, \ell, 1)$-Pluri-SDH problem for bilinear groups of all types, whether symmetric ($\mathbb{G} = \hat{\mathbb{G}}$) or asymmetric ($\mathbb{G} \neq \hat{\mathbb{G}}$), with or without efficient isomorphism ($\psi : \hat{\mathbb{G}} \to \mathbb{G}$), at the cost of some possible redundancy in the instance statement in some cases.

Definition 10. In a bilinear context $\mathbf{G} = (p, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_t, g, \hat{g}, \mathbf{e})$, the $(q, \ell, 1)$-Pluri-SDH problem is:

Given: $\forall i = 0, ..., \ell,\ (g^{\alpha_i}, \hat{g}^{\alpha_i}) \in \mathbb{G} \times \hat{\mathbb{G}}$, and: $\forall j = 1, ..., q,\ (w_{0,j},\ g^{\frac{1}{\alpha_0 + w_{0,j}}}) \in \mathbb{F}_p \times \mathbb{G}$,

output: $\forall i,\ (w_i,\ g^{\frac{r_i}{\alpha_i + w_i}})$, subject to: $\sum_{i=0}^{\ell} r_i = 1 \pmod p$, and: $\forall j,\ w_0 \neq w_{0,j}$.

The advantage of an algorithm $\mathcal{A}$ in solving the $(q, \ell, 1)$-Pluri-SDH problem is defined as,

$$\mathbf{Adv}_{\mathcal{A}}^{\text{PluriSDH}} = \Pr\left[\mathcal{A}\begin{pmatrix} g,\ (g^{\alpha_i})_{0 \le i \le \ell},\ (\ w_{0,j},\ g^{\frac{1}{\alpha_0 + w_{0,j}}}\ )_{1 \le j \le q} \\ \hat{g},\ (\hat{g}^{\alpha_i})_{0 \le i \le \ell} \end{pmatrix} = \Big(\ w_i,\ g^{\frac{r_i}{\alpha_i + w_i}}\ \Big)_{0 \le i \le \ell} : \sum_{i=0}^{\ell} r_i = 1\right]$$

The probability is over the random choice of $g \in \mathbb{G} \setminus \{1\}$ and $\hat{g} \in \hat{\mathbb{G}} \setminus \{1\}$, of α_0, ..., $\alpha_\ell \in \mathbb{F}_p^\times$, of $w_{0,j} \in \mathbb{F}_p \setminus \{-\alpha_0\}$, and over the random bits consumed by $\mathcal{A}$. (We allow g and/or $\hat{g}$ to be externally given, as it is convenient in some bilinear contexts to take $g = \psi(\hat{g})$ under a fixed isomorphism ψ.)

Definition 11. We say the $(q, \ell, 1, t, \epsilon)$-Pluri-SDH assumption to hold in $\mathbf{G} = (p, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_t, g, \hat{g}, \mathbf{e})$ if no t-time and ϵ-advantage randomized algorithm solves the $(q, \ell, 1)$-Pluri-SDH problem in $\mathbf{G}$.

A Spectrum of Assumptions from Pluri-SDH to Poly-SDH. Given the clear similarity between the $(q, \ell+1)$-Poly-SDH and $(q, \ell, 1)$-Pluri-SDH problems, it is natural to consider a generalized notion of Pluri-SDH with $\ell + \ell'$ random generators in total, only ℓ' of which would be given with accompanying series of solution pairs. (In the basic Pluri-SDH problem, we had $\ell' = 1$.)

Definition 12. In a bilinear context $\mathbf{G} = (p, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_t, g, \hat{g}, \mathbf{e})$, the (q, ℓ, ℓ')-Pluri-SDH problem is (using positive and negative indices $i \neq 0$ to differentiate the two types of generators):

Given: $\forall i' = 1, ..., \ell,\ (g^{\alpha_{i'}}, \hat{g}^{\alpha_{i'}}) \in \mathbb{G} \times \hat{\mathbb{G}}$,

and: $\forall i'' = -\ell', ..., -1, \forall j = 1, ..., q,\ (g^{\alpha_{i''}}, \hat{g}^{\alpha_{i''}}) \in \mathbb{G} \times \hat{\mathbb{G}},\ (w_{i'',j},\ g^{\frac{1}{\alpha_{i''} + w_{i'',j}}}) \in \mathbb{F}_p \times \mathbb{G}$,

output: $\forall i = -\ell', ..., -1, 1, ..., \ell,\ (w_i,\ g^{\frac{r_i}{\alpha_i + w_i}})$,

subject to: $\sum_{i \in \{-\ell', ..., \ell\} \setminus \{0\}} r_i = 1 \pmod p$, and also: $\forall i < 0, \forall j,\ w_i \neq w_{i,j}$.

The advantage of an algorithm $\mathcal{A}$ in solving the (q, ℓ, ℓ')-Pluri-SDH problem is defined as,

$$\mathbf{Adv}_{\mathcal{A}}^{\text{PluriSDH}} = \Pr\left[\mathcal{A}\begin{pmatrix} g,\ (g^{\alpha_i})_{\substack{i=-\ell',...,\\ -1,1,...,\ell}},\ (\ w_{0,j},\ g^{\frac{1}{\alpha_i + w_{i,j}}}\ )_{\substack{1 \le j \le q \\ i=-\ell',..., \\ -1,1,...,\ell}} \\ \\ \hat{g},\ (\hat{g}^{\alpha_i})_{\substack{i=-\ell',...,\\ -1,1,...,\ell}} \end{pmatrix} = \Big(\ w_i,\ g^{\frac{r_i}{\alpha_i + w_i}}\ \Big)_i : \sum_i r_i = 1\right]$$

The probability is over the random choice of $g \in \mathbb{G} \setminus \{1\}$ and $\hat{g} \in \hat{\mathbb{G}} \setminus \{1\}$, of all $\alpha_i \in \mathbb{F}_p^\times$, of all $w_{i,j} \in \mathbb{F}_p \setminus \{-\alpha_i\}$, and over the random bits consumed by $\mathcal{A}$. (We allow g and/or $\hat{g}$ to be externally given, as it is convenient in some bilinear contexts to take $g = \psi(\hat{g})$ under a fixed isomorphism ψ.)

Definition 13. We say the $(q, \ell, \ell', t, \epsilon)$-Pluri-SDH assumption to hold in $\mathbf{G} = (p, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_t, g, \hat{g}, \mathbf{e})$ if no t-time and ϵ-advantage randomized algorithm solves the (q, ℓ, ℓ')-Pluri-SDH problem in $\mathbf{G}$.

The generalized (q, ℓ, ℓ')-Pluri-SDH problem is useful to prove security of the mesh scheme in the case where there are $\ell + \ell'$ honest users, but only ℓ' of them are willing to answer queries for atomic signatures (*e.g.*, because there are so many certificate authorities).

A.3 Asymptotic Definitions

For completeness, we also give a complexity-theoretic statement of either assumption, based on the asymptotic limit in an infinite family of bilinear groups. All our security theorems are easy to restate in terms of the asymptotic definition.

Definition 14. We say that the Poly-SDH assumption holds for a bilinear instance generaror $\mathcal{G}$ if, for any polynomial functions $q(\cdot)$, $\ell(\cdot)$, $t(\cdot)$, $\epsilon(\cdot)$, and any algorithm $\mathcal{A}$, there exists a threshold κ^*, beyond which for all values of the security parameter $\kappa > \kappa^*$, the $(q(\kappa), \ell(\kappa), t(\kappa), \epsilon(\kappa))$-Poly-SDH assumption holds against $\mathcal{A}$ in all bilinear instances $\mathbf{G}$ generated by $\mathcal{G}(1^\kappa)$.

Definition 15. We say that the Pluri-SDH assumption holds for a bilinear instance generaror $\mathcal{G}$ if, for any polynomial functions $q(\cdot)$, $\ell(\cdot)$, $\ell'(\cdot)$, $t(\cdot)$, $\epsilon(\cdot)$, and any algorithm $\mathcal{A}$, there exists a threshold κ^*, beyond which for all values of the security parameter $\kappa > \kappa^*$, the $(q(\kappa), \ell(\kappa), \ell'(\kappa), t(\kappa), \epsilon(\kappa))$-Pluri-SDH assumption holds against $\mathcal{A}$ in all bilinear instances $\mathbf{G}$ generated by $\mathcal{G}(1^\kappa)$.

A.4 Generic-Group Complexity of Poly-SDH

We now give a complete proof that the Poly-SDH assumption (and hence, also the weaker Pluri-SDH assumption) is sound in the generic group model in the sense of Shoup [29]. The proof is based on the argument given in Section 3.3.

In this model, slightly extended to incorporate bilinearity, the groups $\mathbb{G}$, $\hat{\mathbb{G}}$, and $\mathbb{G}_t$ are assumed to have a generic presentation, *i.e.*, only canonical operations may be performed on their elements (the group operations in each of $\mathbb{G}$, $\hat{\mathbb{G}}$, and $\mathbb{G}_t$, the isomorphism from $\hat{\mathbb{G}}$ to $\mathbb{G}$ and its inverse, and of course the pairing). Specifically, the solver $\mathcal{A}$ performs the canonical operations by interacting with an oracle $\mathcal{O}$, in such a way that $\mathcal{A}$ only sees arbitrary representations of the group elements. This is modeled using encoding functions f, $\hat{f}$, and f_t, one for each group, and by representing any group element $h \in \mathbb{G}$ as the string $f(h)$ when interacting with $\mathcal{A}$ (and similarly for elements of the other groups). $\mathcal{A}$ is otherwise computationally unbounded.

The following theorem gives an upper bound on the success probability of any generic Poly-SDH attacker $\mathcal{A}$, or equivalently, a lower bound on the complexity of solving Poly-SDH generically.

Theorem 16. *Let $\mathcal{A}$ be a computationally unbounded algorithm for the (q, ℓ)-Poly-SDH problem in generic bilinear groups of prime order p, that makes at most q_G generic group oracle queries in total (i.e., for $\mathbf{e}$, ψ, ψ^{-1}, and $*$ in $\mathbb{G}$, $\hat{\mathbb{G}}$, and $\mathbb{G}_t$). Then,*

$$\Pr\left[\mathcal{A}^{\mathcal{O}} \left(\begin{array}{l} f(g),\ f(g^{\alpha_i})_{1\le i\le \ell}\ ,\ (\, w_{i,j},\ f(g^{\frac{1}{\alpha_i + w_{i,j}}})\,)_{\substack{1\le i\le \ell \\ 1\le j\le q}} \\ \hat{f}(\hat{g}),\ \hat{f}(\hat{g}^{\alpha_i})_{1\le i\le \ell} \end{array} \right) = \Big(\, w_i,\ f(g^{\frac{r_i}{\alpha_i + w_i}})\,\Big)_{1\le i\le \ell}\ :\ \sum_{i=1}^{\ell} r_i = 1 \right]$$
$$= \mathbf{Adv}_{\mathcal{A}}^{\text{PolySDH}} \le \frac{(q_G + q\,\ell + 2\,\ell + 2)^2\,(q\,\ell)}{p-1} = \mathrm{O}(\frac{q_G^2\,q\,\ell + (q\,\ell)^3}{p})\ .$$

In order to be as general as possible, we state and prove the theorem assuming the solver is given access to both ψ and ψ^{-1}, as in type-1 bilinear groups. Since the withholding of one or both of these capabilities can only hurt $\mathcal{A}$, the same theorem gives us valid bounds for type-2 and type-3 groups without needing another proof (albeit, at the cost of a small slack factor).

Proof. We merely sketch the oracle simulation, which follows a similar routine as in [4]; the subsequent analysis will be the core of the argument.

We construct an algorithm $\mathcal{B}$ that simulates the generic group oracle $\mathcal{O}$ without committing to values for the α_i, and analyze what $\mathcal{A}$ can extract from it. Internally, $\mathcal{B}$ keeps track of the group elements by their discrete logarithms to the generators $g \in \mathbb{G}$, $\hat{g} \in \hat{\mathbb{G}}$, and $g_t = \mathbf{e}(g, \hat{g}) \in \mathbb{G}_t$. Since the variables $\alpha_1, ..., \alpha_\ell$ are left undetermined, in all generality these discrete logs will be multivariate expressions in $\mathbb{F}_p[\alpha_1, ..., \alpha_\ell]$, which we denote by ρ_k, $\hat{\rho}_k$, or ρ'_k, depending on the group. Externally, $\mathcal{B}$ maps the corresponding group elements to random strings it gives to $\mathcal{A}$: in the group $\mathbb{G}$ it associates ρ_i to $f_i = f(g^{\rho_i})$, in $\hat{\mathbb{G}}$ it maps $\hat{\rho}_i$ to $\hat{f}_i = \hat{f}(\hat{g}^{\hat{\rho}_i})$, and in $\mathbb{G}_t$ it maps ρ'_i to $f'_i = f_t(g_t^{\rho'_i})$.

To initiate the interaction, $\mathcal{B}$ must provide $\mathcal{A}$ with an instance of the (q, ℓ)-Poly-SDH problem. To do so, $\mathcal{B}$ picks random $w_{i,j} \in \mathbb{F}_p$ for $i = 1, ..., \ell$ and $j = 1, ..., q$, and creates:

- two strings, f_0 and $\hat{f}_0$, which it binds to the constants $\rho_0 = 1$ and $\hat{\rho}_0 = 1$ respectively;
- $2\,\ell$ strings, f_i and $\hat{f}_i$ for $i = 1, ..., \ell$, bound to the expressions $\rho_i = \alpha_i$ and $\hat{\rho}_i = \alpha_i$ respectively;
- $q\,\ell$ strings, f_k for $k = i + j\,\ell$ where $i = 1, ..., \ell$ and $j = 1, ..., q$, bound to the terms $\rho_k = \frac{1}{\alpha_i + w_{i,j}}$.

For simplicity, and to avoid dealing with ratios, we reduce all the expressions to the common denominator $\Delta = \prod_{i=1}^{\ell} \prod_{j=1}^{q} (\alpha_i + w_{i,j})$: for all k, we define $\pi_k = \rho_k\,\Delta$, $\hat{\pi}_k = \hat{\rho}_k\,\Delta$, and $\pi'_k = \rho'_k\,\Delta$. Observe that all of these are polynomials in $\mathbb{F}_p[\alpha_1, ..., \alpha_\ell]$ of degree $\leq q\,\ell + 1$.

$\mathcal{B}$ gives to $\mathcal{A}$ all the strings f_k and $\hat{f}_k$ created above (but not the corresponding polynomials). $\mathcal{B}$ also initializes three counters: $n \leftarrow (q\,\ell + \ell + 1)$, $\hat{n} \leftarrow (\ell + 1)$, and $n_t \leftarrow 0$.

$\mathcal{A}$ then makes a total of q_G adaptive queries to the generic group oracle, of the following kinds:

Group operations: Suppose $\mathcal{A}$ wants to compute the product of two operands in the group $\mathbb{G}$ represented as f_i and f_j. To answer, $\mathcal{B}$ retrieves the polynomials π_i and π_j that these strings correspond to (the same string may appear under multiple indices i, but in all cases the associated polynomial will be the same, so there is no inconsistency). It calculates the polynomial sum $\pi_n = \pi_i + \pi_j \in \mathbb{F}_p[\alpha]$. If the result π_n is already present in the $\mathbb{G}$-mapping, the corresponding string is copied into f_n, otherwise a new string is created; then the entry (π_n, f_n) is added to the mapping. Finally, $\mathcal{B}$ gives f_n to $\mathcal{A}$, and then increments $n \leftarrow n + 1$.

If $\mathcal{A}$ had wanted the ratio in lieu of the product, $\mathcal{B}$ would have let $\pi_n = \pi_i - \pi_j \in \mathbb{F}_p[\alpha]$.

Group operation queries in $\hat{\mathbb{G}}$ or $\mathbb{G}_t$ are serviced analogously, using the relevant mappings.

Isomorphisms: Suppose $\mathcal{A}$ wants to map an operand $\hat{f}_i$ in $\hat{\mathbb{G}}$ to its image in $\mathbb{G}$ by the isomorphism. To answer, $\mathcal{B}$ proceeds as above to retrieve the polynomial $\hat{\pi}_i$, assigns $\pi_n = \hat{\pi}_i$, and lets f_n be the copy of an existing string or a new string depending on whether π_n was already present in the $\mathbb{G}$-mapping. It adds (π_n, f_n) to the mapping, gives f_n to $\mathcal{A}$, then increments $n \leftarrow n + 1$.

Inverse isomorphism queries are answered similarly, after exchanging the roles of $\mathbb{G}$ and $\hat{\mathbb{G}}$.

Pairing: Suppose $\mathcal{A}$ wants to compute the bilinear map between f_i in $\mathbb{G}$ and $\hat{f}_j$ in $\hat{\mathbb{G}}$. To answer, $\mathcal{B}$ retrieves the polynomials π_i and $\hat{\pi}_j$, computes the polynomial product $\pi_{n_t} = \pi_i \cdot \pi_j \in \mathbb{F}_p[\alpha]$, determines whether π_{n_t} already exists in the $\mathbb{G}_t$-mapping, and accordingly lets f_{n_t} be a clone of the corresponding string or a new string. It adds (π_{n_t}, f_{n_t}) to the mapping, gives f_{n_t} to $\mathcal{A}$, then increments $n_t \leftarrow n_t + 1$.

W.l.o.g., we have assumed that $\mathcal{A}$ only queried $\mathcal{B}$ on legitimate strings that were previously revealed, and similarly assume that it outputs its solution in terms of such strings exclusively.

After q_G queries, $\mathcal{A}$ returns a solution to the Poly-SDH instance, in the form of ℓ pairs $(w_1,\, f_{k_1})$, ..., $(w_\ell,\, f_{k_\ell})$ where $0 \leq k_i < n$. We denote by π_{k_i} the formal polynomials that correspond to the generic representations f_{k_i}.

To verify the solution, $\mathcal{B}$ randomly selects $\alpha_i^* \in \mathbb{F}_p^\times$ for each i, and evaluates the formal polynomials under the assignment $(\alpha_1, ..., \alpha_\ell) = (\alpha_1^*, ..., \alpha_\ell^*)$. The validation equation is,

$$\sum_{i=1}^{\ell} (\alpha_i + w_i)\, \rho_{k_i}(\alpha_1, ..., \alpha_\ell) = 1 \ , \qquad i.e., \qquad \sum_{i=1}^{\ell} (\alpha_i + w_i)\, \pi_{k_i}(\alpha_1, ..., \alpha_\ell) = \Delta \ . \tag{$\star$}$$

$\mathcal{A}$ wins the game outright if Equation ($\star$) holds in $\mathbb{F}_p$ under the random assignment.

$\mathcal{A}$ wins by default if any two distinct polynomials representing elements of the same group (*e.g.*, $\pi_i \neq \pi_j$, or $\hat{\pi}_i \neq \hat{\pi}_j$, or $\pi'_i \neq \pi'_j$) evaluate to the same value in $\mathbb{F}_p$ under the assignment (*viz.*, $\pi_i(\alpha_1^*, ..., \alpha_\ell^*) = \pi_j(\alpha_1^*, ..., \alpha_\ell^*)$, *etc.*): if this is the case, $\mathcal{B}$'s simulation is flawed since it portrayed as distinct two instances of the same group element.

$\mathcal{A}$ loses the game barring the two scenarios above. We shall bound the probability of either event, below.

Notice that all polynomials π used by $\mathcal{B}$ to represent an element in $\mathbb{G}$ or $\hat{\mathbb{G}}$ have degree $\leq q\,\ell+1$, since they are constructed using sums and differences from an initial set of polynomials with this property (the polynomials of highest degrees are $\pi_i = \rho_i\, \Delta = \alpha_i\, \Delta$ for $i = 1, ..., \ell$, of degree $q\,\ell + 1$). In the target group $\mathbb{G}_t$, the polynomials can be of degree $\leq 2\,q\,\ell + 2$, which is the highest degree that the product of two polynomials of degree $\leq q\,\ell + 1$ can attain.

To bound the probability that $\mathcal{A}$ makes a correct answer, we bound the probability that Equation ($\star$) will be satisfied. First, we show that it cannot hold identically in $\mathbb{F}_p[\alpha_1, ..., \alpha_\ell]$. Consider the following change of variables: for all $i = 1, ..., \ell$ and $j = 1, ..., q$, we define $\alpha'_i = \alpha_i + w_i$ and $w'_{i,j} = w_{i,j} - w_i$, where we note that $w'_{i,j} \neq 0$ because of the constraints on the w_i. The change of variable is well-defined and unambiguous, and lets us rewrite Equation ($\star$) as,

$$\sum_{i=1}^{\ell} \alpha'_i\, \pi_{k_i}(\alpha'_1 - w_1, ..., \alpha'_\ell - w'_1) = \Delta = \prod_{i=1}^{\ell} \prod_{j=1}^{q} (\alpha_i + w_{i,j}) = \prod_{i=1}^{\ell} \prod_{j=1}^{q} (\alpha'_i + w'_{i,j}) \ . \tag{$\star\star$}$$

Equation ($\star\star$) cannot hold identically in $\mathbb{F}_p[\alpha'_1, ..., \alpha'_\ell]$ since the left-hand side has no independent term while the right-hand side contains the independent term $\prod_i \prod_j w'_{i,j} = \prod_i \prod_j (w_{i,j} - w_i) \neq 0$. Equation ($\star$) then does not hold identically either, in $\mathbb{F}_p[\alpha_1, ..., \alpha_\ell]$, which means that it is a non-trivial polynomial equation of degree $\leq q\,\ell + 2$. For a random assignment of $(\alpha_1,\ ...,\ \alpha_\ell) \in (\mathbb{F}_p^\times)^\ell$, Equation ($\star$) will thus be satisfied in $\mathbb{F}_p$ with probability $\leq \frac{q\,\ell+2}{p-1}$.

To bound the probability that $\mathcal{A}$ wins by default, recall that all the polynomials representing elements in $\mathbb{G}$ and $\hat{\mathbb{G}}$ have degree $\leq q\,\ell + 1$, and those in $\mathbb{G}_t$ have degree $\leq 2\,q\,\ell + 2$. Again using the standard argument on the probability of satisfying polynomial equations, and combining all of these with the union bound, we find that the probability that there is at least one flaw in $\mathcal{B}$'s simulation is $\leq \binom{n}{2} \frac{q\,\ell+1}{p-1} + \binom{\hat{n}}{2} \frac{q\,\ell+1}{p-1} + \binom{n_t}{2} \frac{2\,q\,\ell+2}{p-1}$.

(The denominators are $p-1$ rather than p to reflect that the α_i are random in $\mathbb{F}_p^\times$ and not in $\mathbb{F}_p$: conceivably, the adversary could exploit this by forcing all polynomial solutions to be non-zero.) Now, if we take the union bound over the two events, and bring into account the loop invariant, $(n + \hat{n} + n_t) = (q\,\ell + 2\,\ell + 2 + q_G)$, we thus establish,

$\mathbf{Adv}_{\mathcal{A}}^{\mathrm{PolySDH}} \leq \binom{n}{2} \frac{q\,\ell+1}{p-1} + \binom{\hat{n}}{2} \frac{q\,\ell+1}{p-1} + \binom{n_t}{2} \frac{2\,q\,\ell+2}{p-1} + \frac{q\,\ell+1+2}{p-1} \leq (q_G + q\,\ell + 2\,\ell + 2)^2\, (q\,\ell)/(p-1)$, *i.e.*, $\mathbf{Adv}_{\mathcal{A}}^{\mathrm{PolySDH}} = \mathrm{O}(q_G^2\, q\,\ell/p + (q\,\ell)^3/p)$. □

We can state the generic hardness of the asymptotic Poly-SDH assumption much more simply, as follows.

Corollary 17. *All algorithms that solve (q, ℓ)-Poly-SDH problems with constant probability $\epsilon > 0$ in generic bilinear groups of order p such that $q\,\ell < \mathrm{O}(\sqrt[3]{p})$ require $\Omega(\sqrt{\epsilon\, p/q\,\ell})$ generic operations.*

A.5 Generic-Group Complexity of Pluri-SDH

Essentially the same proof as in Appendix A.4 gives the following generic-group complexity lower bounds for the Pluri-SDH assumption. We give the bounds for the general (q, ℓ, ℓ')-Pluri-SDH problem, and note that they also apply if we set $\ell' = 1$, which is the relevant assumption for the ring signature proof.

Theorem 18. *Let $\mathcal{A}$ be a computationally unbounded algorithm for the (q, ℓ, ℓ')-Pluri-SDH problem in generic bilinear groups of prime order p, that makes q_G queries to a generic group oracle. Then,*

$$\Pr\left[\mathcal{A}^{\mathcal{O}}\left(\begin{array}{l} f(g),\ f(g^{\alpha_i})_{\substack{i=-\ell',\ldots,\\ -1,1,\ldots,\ell}}\ ,\ (\,w_{i,j},\ f(g^{\frac{1}{\alpha_i+w_{i,j}}})\,)_{\substack{1\le j\le q\\ i=-\ell',\ldots,\\ -1,1,\ldots,\ell}} \\[2ex] \hat{f}(\hat{g}),\ \hat{f}(\hat{g}^{\alpha_i})_{\substack{i=-\ell',\ldots,\\ -1,1,\ldots,\ell}} \end{array}\right) = \Big(\,w_i,\ f(g^{\frac{r_i}{\alpha_i+w_i}})\,\Big)_i\ :\ \sum_i r_i = 1\right]$$
$$= \mathbf{Adv}_{\mathcal{A}}^{\mathrm{PluriSDH}} \le \frac{(q_G + q\,\ell' + 2\,\ell' + 2\,\ell)^2\,(q\,\ell' + \ell)}{p-1} = \mathrm{O}\Big(\frac{q_G^2\,(q\,\ell' + \ell) + (q\,\ell' + \ell)^3}{p}\Big)\ .$$

Proof. The proof is analogous to that of Theorem 16. □

We can restate the generic hardness of the asymptotic Pluri-SDH assumption more concisely:

Corollary 19. *Any algorithm that solves the (q, ℓ, ℓ')-Pluri-SDH problem with constant probability $\epsilon > 0$ in generic bilinear groups of prime order p such that $\ell \le q\,\ell' < \mathrm{O}(\sqrt[3]{p})$ requires $\Omega(\sqrt{\epsilon\, p/q\,\ell'})$ generic group operations.*

A.6 On Selecting the Group Order

A recent analysis of SDH shows that it may be helpful (but not required) to ensure that $p-1$ and $p+1$ have large prime factors [13]. It is shown how to recover α generically from an Original SDH instance, in time as low as $\Theta(\log p\,\sqrt{p/q})$ for certain values of q provided that $p \pm 1$ is smooth. Conversely, it is also conjectured that for "safe" p the generic time complexity should be brought back up to $\Theta(\sqrt{p})$, on a par with the discrete logarithm bound.

Parameterizing the system based on this conjecture (rather than the generic bound from [4]) is enticing because it entails smaller group sizes for a prescribed security level. However, requiring multiple primality constraints to be satisfied at once when constructing a pairing-friendly curve will undoubtedly complicate the construction. This may be a reason to disregard the conjecture, and instead use the generic bounds when parameterizing an SDH-based system.

These concerns do not carry over to the assumptions we made, for the main reason that the analysis from [13] does not seem to generalize to them. Specifically, the analysis from [13] was crucially based on Original SDH instances (given as a series of powers), rather than Modified SDH (given as a list of example solutions). Since we stated Poly-SDH and Pluri-SDH in the same "list of solutions" form as the latter, the recommendations on the choice of p should not be of great concern here.

We emphasize also that our generic complexity bounds are true for all primes p.

B Ring Scheme Security Proofs

This section focuses on the security properties of the special-case ring signature scheme of Section 4.

B.1 Anonymity of the Ring Scheme

We prove Theorem 1 using an information-theoretic argument.

Proof of Theorem 1. We need to show that it is impossible to determine which ring member produced a signature, even if all the secret keys are revealed. Observe that the distribution of σ is uniform over the $(2\,\ell+1)$-dimensional variety of $\mathbb{G}^{\ell+1} \times \mathbb{F}_p^{\ell+1}$ defined by the ring verification equation, *i.e.*,

$$\mathbf{P}(\sigma) = \mathbf{U}\left((S_0,\ ...,\ S_\ell,\ t_0,\ ...,\ t_\ell) \in \mathbb{G}^{\ell+1} \times \mathbb{F}_p^{\ell+1}\ :\ \prod_{i=0}^{\ell} \mathbf{e}(S_i,\ \hat{A}_i\, \hat{B}_i^{m_i}\, \hat{C}_i^{t_i}) = \mathbf{e}(g,\ \hat{g}) \right) \ .$$

Conditionally on the public information, the random variable σ is jointly independent of the signer identity and all the secret data (including the random coins used to generate the signing keys). The theorem follows immediately. □

The reader will notice that the independence does not extend conditionally on the ephemerals s_i that were used to create the signature. Even though it is in the interest of the signer to erase those immediately, a powerful adversary could possibly recompute them (by solving a discrete logarithm, as infeasible as it sounds), so one may ask how this affects anonymity. The answer is that is does not, because there are many possible sets of ephemerals, one for each member of the ring, and by Theorem 1 none of them can be preferred over any other based on public or secret information. [1]

B.2 Unforgeability of the Ring Scheme

We now prove a stronger version of Theorem 2 where the forger is given to make atomic signature queries, based on the stronger $(q, \ell+1)$-Poly-SDH assumption in $\mathbf{G}$. (The simpler Theorem 2 can be proven along the same lines.) The precise statement of the theorem we prove is as follows.

Theorem 20. *The ring signature is existentially unforgeable under an adaptive attack, against a static adversary that makes no more than q ring signature queries, and q atomic signature queries to each one of the ℓ honest users, adaptively, provided that the $(q, \ell+1)$-Poly-SDH assumption holds in $\mathbf{G}$, in the common random string model.*

[1] There remains the possibility that the signer could reveal the ephemerals *voluntarily* to revoke *her own* anonymity, but this falls outside of the purview of Theorem 1: the same outcome can be achieved generically in all ring signatures. To do so, the signer would append to the message being ring-signed, a one-time signature verification key, an encrypted signature of the same under her public key, and a signature of a hash of the decryption key under the one-time key; she could then provably de-anonymize herself by revealing the decryption key.

Proof. Let $\mathbf{G} = (p, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_t, g, \hat{g}, \mathbf{e})$ be some bilinear instance with a computable isomorphism $\psi : \hat{\mathbb{G}} \to \mathbb{G}$. Suppose that $H : \{0,1\}^* \to \mathbb{F}_p$ is a collision-resistant hash function.

We are given a random instance of the $(q, \ell+1)$-Poly-SDH problem in $\mathbf{G}$, stated as $\ell+1$ pairs $(g_i = g^{\alpha_i},\ \hat{g}_i = \hat{g}^{\alpha_i})$ for $i = 0, ..., \ell$, and $(\ell+1)\, q$ pairs $(w_{i,j},\ u_{i,j} = g^{1/\alpha_i + w_{i,j}})$ for $i = 0, ..., \ell$ and $j = 1, ..., q$. Our task is to output $\ell+1$ pairs $(w_i^*,\ u_i^* = g^{r_i^*/\alpha_i + w_i^*})$ for $i = 0, ..., \ell$, for some public choice of w_i^* such that $w_i^* \notin \{w_{i,1}, ..., w_{i,q}\}$ for all $i = 0, ..., \ell$, and some secret choice of r_i^* such that $\sum_{i=0}^{\ell} r_i^* = 1 \pmod p$.

We construct an algorithm $\mathcal{B}$ that solves such instances of the Poly-SDH problem by interacting with a black-box forger $\mathcal{A}$ for the ring scheme. For simplicity, we give sequential numbers in $\{1, 2, ...\}$ to all users (*i.e.*, potential ring members). Since the adversary is static, we suppose w.l.o.g. that the target users will consist of the set $\{1, ..., \ell\}$, and that the target ring will be $R^* = \{1, ..., \ell\}$.

Indeed, per the model of Section 2.3, any forgery would have to bear on a ring R^* that is a subset of $\{1, ..., \ell\}$: this is because, for ring signatures, a well-formed specification Υ is a flat disjunction of user/message clauses, and only with all users in the set $\{1, ..., \ell\}$ can it imply another disjunction Υ' that remains within the set (we can enlarge any subset to the whole set, but no further). We acknowledge that any target ring $R^* \subseteq \{1, ..., \ell\}$ would be fair game for the adversary, but since a larger ring only makes the forgery easier, we can take it as the whole set $R^* = \{1, ..., \ell\}$, w.l.o.g.

In this setting, we use the index $i = 0$ for the key "in the sky", and assume that $\mathcal{B}$ has ownership of the first ℓ users with $1 \le i \le \ell$, and that $\mathcal{A}$ controls all the others with $\ell+1 \le i \le I_{\max}$ for some polynomial bound $I_{\max}$. Each player reveals the public keys of the users in its custody, first $\mathcal{B}$, and then $\mathcal{A}$, without revealing the private keys.

For generality and homogeneity of notation, we will allow each ring signature component to bear on an arbitrary message in $\mathbb{F}_p$, *i.e.*, we do not require that the messages be the same.

Regular Simulation. We first describe a "regular" simulation that $\mathcal{B}$ can use when the adversary $\mathcal{A}$ does not behave in a pathological manner (to be made precise later). The reduction follows.

To start, the simulator $\mathcal{B}$ must fix the common random string from the distribution expected by $\mathcal{A}$. First, $\mathcal{B}$ publishes the bilinear instance $\mathbf{G}$ and the isomorphism ψ. Next, $\mathcal{B}$ chooses random $b_0,\ c_0 \in \mathbb{F}_p^\times$ and publishes $(\hat{A}_0,\ \hat{B}_0,\ \hat{C}_0) = (\hat{g}_0,\ \hat{g}^{b_0},\ \hat{g}^{c_0})$ as the public key "in the sky"; this implicitly reveals $(A_0,\ B_0,\ C_0) = (g_0,\ g^{b_0},\ g^{c_0})$. Last, $\mathcal{B}$ publishes the description of H.

$\mathcal{B}$ gives $\mathcal{A}$ the public keys of the first ℓ users. To do so, for each $i = 1, ..., \ell$, it draws random $b_i,\ c_i \in \mathbb{F}_p^\times$ and publishes the tuple $(A_i,\ B_i,\ C_i,\ \hat{A}_i,\ \hat{B}_i,\ \hat{C}_i) = (g_i,\ g^{b_i},\ g^{c_i},\ \hat{g}_i,\ \hat{g}^{b_i},\ \hat{g}^{c_i})$.

$\mathcal{A}$ gives $\mathcal{B}$ the public keys $(A_i,\ B_i,\ C_i,\ \hat{A}_i,\ \hat{B}_i,\ \hat{C}_i)$ of the users it controls, $i = (\ell+1), ..., I_{\max}$. Here, $I_{\max}$ is the total number of users, which must be polynomially bounded.

Per the unforgeability model, $\mathcal{A}$ is allowed to give those values to $\mathcal{B}$ interactively, interleaved with the signature queries described below. We only require that $\mathcal{A}$ reveal a public key before it makes any signature query that involves that key.

$\mathcal{A}$ makes q_S distinct ring signatures queries to $\mathcal{B}$, one at a time, proceeding adaptively. To exhaust the quota of queries that are available to $\mathcal{A}$, we assume w.l.o.g. that $q_S = q$.

For $j = 1, ..., q_S$, the j-th query is a pair $(M_j,\ R_j)$ where R_j is a ring and M_j is a vector of messages to be signed by that ring. Let n be the number of users in the ring. Let thus $R_j = \{i_1, ..., i_n\} \subseteq \{1, ..., I_{\max}\}$, and $M_j = (m_1, ..., m_n) \in \mathbb{F}_p^n$. We require that $R_j \cap \{1, ..., \ell\} \neq \emptyset$; otherwise the simulator is not supposed to respond and the query will be denied.

$\mathcal{B}$ responds to a well-formed j-th query as follows.

Select n random exponents $s_1,\ ...,\ s_n \in \mathbb{F}_p$, and n random integers $t_1,\ ...,\ t_n \in \mathbb{F}_p$. Compute $m_0 = H((i_1, m_1), ..., (i_n, m_n))$ and set $t_0 = (w_{0,j} - b_0\, m_0)/c_0$. Here, b_0 and c_0 are the secret exponents chosen during setup, and $w_{0,j}$ is taken from $(w_{0,j},\ u_{0,j})$ in the problem instance.

$\mathcal{B}$ replies to $\mathcal{A}$ by returning the signature $\sigma_j = (S_0,\ ...,\ S_n,\ t_0,\ ...,\ t_n)$, which is given by,

$$\sigma_j = \left(\begin{array}{c} \left(u_{0,j} \cdot \prod_{k=1}^{n} (A_{i_k}\, B_{i_k}^{m_k}\, B_{i_k}^{t_k})^{-s_k} \right),\ \left(A_0\, B_0^{m_0}\, C_0^{t_0}\right)^{s_1},\ ...,\ \left(A_0\, B_0^{m_0}\, C_0^{t_0}\right)^{s_n}, \\ t_0,\, t_1,\ ...,\ t_n \end{array} \right) .$$

Note that σ_j is properly randomized and passes the validity test: $\prod_{k=0}^{n} \mathbf{e}(S_k,\ \hat{A}_{i_k}\, \hat{B}_{i_k}^{m_k}\, \hat{C}_{i_k}^{t_k}) = \mathbf{e}\left(g^{\frac{1}{\alpha_0 + w_{0,j}}}\ \prod_{k=1}^{n} (A_{i_k}\, B_{i_k}^{m_k}\, C_{i_k}^{t_k})^{-s_k},\ \hat{A}_0\, \hat{B}_0^{m_0}\, \hat{C}_0^{t_0} \right) \cdot \prod_{k=1}^{n} \mathbf{e}\left((A_0\, B_0^{m_0}\, C_0^{t_0})^{s_k}, \hat{A}_{i_k}\, \hat{B}_{i_k}^{m_k}\, \hat{C}_{i_k}^{t_k} \right) = \mathbf{e}\left(g^{\frac{1}{\alpha_0 + w_{0,j}}},\ \hat{A}_0\, \hat{B}_0^{m_0}\, \hat{C}_0^{t_0} \right) = \mathbf{e}\left(g^{\frac{1}{\alpha_0 + w_{0,j}}},\ \hat{g}^{\alpha_0 + w0,j} \right) = \mathbf{e}(g,\ \hat{g})$, as required.

$\mathcal{A}$ can also make $q_{S,i}$ distinct atomic signature queries for each user $i = 1, ..., \ell$ controlled by $\mathcal{B}$. These queries can be adaptative and interleaved with the ring signature queries. Again to exhaust the quota of queries that are available to $\mathcal{A}$, we assume w.l.o.g. that $q_{S,i} = q$ for $i = 1, ..., \ell$.

$\mathcal{B}$ responds to the j-th query (i, m) to the i-th user, by retrieving the pair $(w_{i,j},\ u_{i,j})$ from the Poly-SDH instance, computing $t = (w_{i,j} - b_i\, m)/c_i$, and returning the Boneh-Boyen signature $(u_{i,j},\ t)$ to $\mathcal{A}$.

$\mathcal{A}$ eventually outputs a forgery σ^* on some list of messages $M^* = (m_1^*, ..., m_\ell^*)$ attributed to the target ring $R^* = \{1, ..., \ell\}$. To preclude trivial forgeries, we demand that $(M^* \neq M_j) \vee (R^* \neq R_j)$ for all queries (M_j, R_j) previously made by $\mathcal{A}$.

The forgery is a signature of the form $\sigma^* = (S_0^*,\ ...,\ S_\ell^*,\ t_0^*,\ ...,\ t_\ell^*)$ and must necessarily satisfy, for $m_0^* = H((1, m_1^*), ..., (\ell, m_\ell^*))$,

$$\prod_{i=0}^{\ell} \mathbf{e}\left(S_i^*,\ \hat{A}_i\, \hat{B}_i^{m_i^*}\, \hat{C}_i^{t_i^*} \right) = \mathbf{e}(g,\ \hat{g}) .$$

It follows that there exist ℓ unknown exponents $r_1^*,\ ...,\ r_\ell^* \in \mathbb{F}_p$ such that,

$$S_0 = \left(g^{\frac{1}{\alpha_0 + b_0\, m_0^* + c_0\, t_0^*}} \right)^{1 - \sum_{k=1}^{\ell} r_k^*}, \qquad \text{and} \quad S_i = \left(g^{\frac{1}{\alpha_i + b_i\, m_i^* + c_i\, t_i^*}} \right)^{r_i^*} \quad \text{for } i = 1, ..., \ell .$$

Thus, if we compute $w_i^* = b_i\, m_i^* + c_i\, t_i^*$ for $i = 0, ..., \ell$, and formally define $r_0^* = 1 - \sum_{k=1}^{\ell} r_k^*$, we obtain the following solution to the Poly-SDH problem instance,

$$(\ (w_0^*,\ S_0^*),\ ...,\ (w_\ell^*,\ S_\ell^*)\ ) = \left(\left(w_0^*,\ u_0^* = g^{\frac{r_0^*}{\alpha_0 + w_0^*}} \right),\ ...,\ \left(w_\ell^*,\ u_\ell^* = g^{\frac{r_\ell^*}{\alpha_\ell + w_\ell^*}} \right) \right) .$$

$\mathcal{B}$ can compute all the w_i^* and u_i^*, even though it does not know any of the r_i^*.

The preceding reduction will succeed to produce a solution to the given Poly-SDH instance, when the adversary's forgery is such that $w_i^* \notin \{w_{i,1}, ..., w_{i,q}\}$ for all $i = 0, ..., \ell$.

However, $\mathcal{A}$ could produce an anomalous forgery where $w_{i^*}^* \in \{w_{i^*,1}, ..., w_{i^*,q}\}$ for some $i^* \leq \ell$, either by luck or by design. The preceding simulation is useless in this case, since it would produce an easy solution to the Poly-SDH problem that is explicitly forbidden. We use an "alternative" simulation to deal with this anomaly.

Alternative Simulation. We construct an "alternative" reduction for the case where $\mathcal{A}$'s forgery causes the parameter $w_{i^*}^*$ to land in the set $\{w_{i^*,1}, ..., w_{i^*,q}\}$ of parameters explicitly listed in the problem instance given to $\mathcal{B}$. The simulation is as follows.

To start, the simulator $\mathcal{B}$ posts a common random string from the distribution expected by $\mathcal{A}$. To do so, $\mathcal{B}$ publishes the bilinear instance $\mathbf{G}$ and the isomorphism ψ; it chooses random $a_0,\ b_0 \in \mathbb{F}_p^\times$ and publishes $(\hat{A}_0,\ \hat{B}_0,\ \hat{C}_0) = (\hat{g}^{a_0},\ \hat{g}^{b_0},\ \hat{g}_0)$; it also publishes a description of H.

$\mathcal{B}$ gives $\mathcal{A}$ the public keys of the first ℓ users. To do so, for each $i = 1, ..., \ell$, it draws random $a_i,\ b_i \in \mathbb{F}_p^\times$ and publishes the tuple $(A_i,\ B_i,\ C_i,\ \hat{A}_i,\ \hat{B}_i,\ \hat{C}_i) = (g^{a_i},\ g^{b_i},\ g_i,\ \hat{g}^{a_i},\ \hat{g}^{b_i},\ \hat{g}_i)$.

$\mathcal{A}$ gives $\mathcal{B}$ the public keys $(A_i,\ B_i,\ C_i,\ \hat{A}_i,\ \hat{B}_i,\ \hat{C}_i)$ of the users it controls, $i = (\ell+1), ..., I_{\max}$. Again, $I_{\max}$ is the total number of users, which must be polynomially bounded.

As before, $\mathcal{A}$ may keep introducing new keys after it has started making signature queries, as long as public keys are revealed before queries that depend on them.

$\mathcal{A}$ makes q distinct ring signatures queries to $\mathcal{B}$, proceeding adaptively. As before, the j-th query is given as a pair $(M_j,\ R_j)$ with $M_j = (m_1, ..., m_n) \in (\mathbb{F}_p)^n$ for some n and $R_j = \{i_1, ..., i_n\} \subseteq \{1, ..., I_{\max}\}$ such that $R_j \cap \{1, ..., \ell\} \neq \emptyset$. Upon receiving this query, $\mathcal{B}$ responds as follows.

$\mathcal{B}$ computes $m_0 = H((i_1, m_1), ..., (i_n, m_n))$ and sets $t_0 = (a_0 + b_0\, m_0)/w_{0,j}$ using the j-th pair $(w_{0,j}, u_{0,j})$ from the problem instance. $\mathcal{B}$ also computes $V_j = B_0^{m_0}\, C_0^{t_0}$ and stores the tuple $(0,\ V_j,\ m_0,\ t_0,\ M_j,\ R_j)$ in some searchable database for future use.

$\mathcal{B}$ then chooses n random exponents $s_1,\ ...,\ s_n \in \mathbb{F}_p$, and n random integers $t_1,\ ...,\ t_n \in \mathbb{F}_p$, and answers the query by giving to $\mathcal{A}$ the signature,

$$\sigma_j = \left(\begin{array}{c} \left(u_{0,j}^{1/t_0} \cdot \prod_{k=1}^{n} (A_{i_k}\, B_{i_k}^{m_k}\, C_{i_k}^{t_k})^{-s_k} \right),\ \left(A_0\, B_0^{m_0}\, C_0^{t_0}\right)^{s_1},\ ...,\ \left(A_0\, B_0^{m_0}\, C_0^{t_0}\right)^{s_n}, \\ t_0,\ t_1,\ ...,\ t_n \end{array} \right).$$

We show that the signature $\sigma_j = (S_0,\ ...,\ S_n,\ t_0,\ ...,\ t_n)$ has the correct distribution. First, it has the requisite $2\,n+1$ degrees of freedom, and so is adequately randomized. Second, in the verification equation, all the factors under the pairings cancel out, except for $u_{0,j}^{1/t_0}$. Therefore, writing $i_0 = 0$, we find, $\prod_{k=0}^{n} \mathbf{e}\left(S_k,\ \hat{A}_{i_k}\, \hat{B}_{i_k}^{m_k}\, \hat{C}_{i_k}^{t_k}\right) = \mathbf{e}\left((u_{0,j}^{1/t_0}),\ \hat{A}_0\, \hat{B}_0^{m_0}\, \hat{C}_0^{t_0}\right) = \mathbf{e}\left(g^{\frac{w_{0,j}/(a_0+b_0\, m_0)}{\alpha_0+w_{0,j}}},\ \hat{g}^{a_0+b_0\, m_0}\, \hat{g}_0^{t_0}\right) = \mathbf{e}\left(g^{\frac{w_{0,j}}{\alpha_0+w_{0,j}}},\ \hat{g}\right) \mathbf{e}\left(g^{\frac{1}{\alpha_0+w_{0,j}}},\ \hat{g}^{\alpha_0}\right) = \mathbf{e}(g,\ \hat{g})$, as required.

$\mathcal{A}$ also makes q distinct atomic signature queries for each of the users $i = 1, ..., \ell$ controlled by $\mathcal{B}$. These queries are adaptative and interleaved with the ring signature queries.

$\mathcal{B}$ responds to the j-th query (i, m) on behalf of the i-th user, as follows.

It retrieves the fresh pair $(w_{i,j},\ u_{i,j})$ from the Poly-SDH instance, and sets $t = (a_i + b_i\, m)/w_{i,j}$. It also computes $\mathcal{B}$ also computes $V = B_i^m\, C_i^t$ and stores the tuple $(i,\ V,\ m,\ t,\ \emptyset,\ \emptyset)$ the database for future use. It then returns the Boneh-Boyen signature $(u_{i,j}^{1/t},\ t)$ to $\mathcal{A}$.

$\mathcal{A}$ finally outputs a forgery σ^* bearing on a message vector $M^* = (m_1^*, ..., m_\ell^*)$ and the target ring $R^* = \{1, ..., \ell\}$, provided that $(M^* \neq M_j) \vee (R^* \neq R_j)$ for all queries (M_j, R_j) made earlier. We let $m_0^* = H((1, m_1^*), ..., (\ell, m_\ell^*))$.

The forgery $\sigma^* = (S_0^*,\ ...,\ S_\ell^*,\ t_0^*,\ ...,\ t_\ell^*)$ is not valid unless $\prod_{i=0}^{\ell} \mathbf{e}(S_i,\ \hat{A}_i\, \hat{B}_i^{m_i^*}\, \hat{C}_i^{t_i^*}) = \mathbf{e}(g,\ \hat{g})$, *i.e.*, there must exist $r_1,\ ...,\ r_\ell \in \mathbb{F}_p$ and $r_0 = 1 - \sum_{k=1}^{\ell} r_k$ such that,

$$S_i = \left(g^{\frac{1}{a_i + b_i\, m_i^* + \alpha_i\, t_i^*}} \right)^{r_i} \qquad \text{for } i = 0, ..., \ell\ .$$

For $i = 0, ..., \ell$, let us define $w_i^* = b_i\, m_i^* + c_i\, t_i^*$ for the value $c_i = \mathrm{dlog}_g(C_i) = \alpha_i$ that would have given $\mathcal{A}$ the same view in the regular simulation. Let us also define $w_i^{**} = (a_i + b_i\, m_i^*)/t_i^*$ with a_i and b_i as chosen in the present simulation. Remark that $\mathcal{B}$ is unable to compute any of the w_i^*, but it can and does compute all the w_i^{**}.

$\mathcal{B}$ exploits all of this as follows. It computes $V_i^* = (B_i)^{m_i^*}\, (C_i)^{t_i^*} = g^{w_i^*}$ for $i = 0, ..., \ell$, and then searches the database for an entry $(i, V_j,\ m_i^{(j)},\ t_i^{(j)},\ M_j,\ R_j)$ such that $V_j = V_i^*$, or equivalently, such that $b_i\, m_i^{(j)} + \alpha_i\, t_i^{(j)} = w_i^*$.

There are three (possibly overlapping) possibilities:

1. An entry was found with $V_j = V_i^*$ and $m_i^{(j)} \neq m_i^*$ for some i: In this case, $\mathcal{B}$ can resolve the Poly-SDH instance explicitly by recovering the secret exponent $\alpha_i \in \mathbb{F}_p$,

$$\alpha_i = \frac{(t_i^* - t_i^{(j)})}{b_i\, (m_i^{(j)} - m_i^*)}\ .$$

2. An entry was found with $V_j = V_0^*$ and $m_0^{(j)} = m_0^*$: In this case, $\mathcal{B}$ has found a collision in the supposedly collision-resistant hash function H, since we have,

$$\overbrace{H(\quad \underbrace{f(M_j, R_j)}_{((i_1, m_1^{(j)}), ..., (i_n, m_n^{(j)}))} \quad)}^{m_0^{(j)}} \quad \overset{=}{\underset{\neq}{}} \quad \overbrace{H(\quad \underbrace{f(M^*, R^*)}_{((1, m_1^*), ..., (\ell, m_\ell^*))} \quad)}^{m_0^*}\ .$$

3. An entry was found with $V_j = V_i^*$ and $m_i^{(j)} = m_i^*$ for $i \neq 0$: In this case, the ring signature forgery includes a clause $[VK_i\colon m_i^*]$ on which $\mathcal{A}$ had previously made an atomic signature query (it being the j-th query to the i-th user). The forgery is therefore inadmissible and this case may be discounted (regardless of whether $V_j = V_i^*$ or $V_j \neq V_i^*$).
4. No entry that matches the conditions was found in the database: This corresponds to the event that $\forall i, \forall j, (w_i^* \neq b_i\, m_i^{(j)} + c_i\, t_i^{(j)})$ where $c_i = \mathrm{dlog}_g(C_i)$. In this case $\mathcal{B}$ is stuck, but from the point of view of $\mathcal{A}$ this is precisely when the "regular" simulation will be able to proceed to completion.

Fair Prior Apportionment. Even though $\mathcal{B}$ will not know in advance which of the "regular" and "alternative" simulation should be used, we can see that they all appear identical to the adversary, so by a standard argument, if $\mathcal{B}$ makes a fair random choice at the onset, the final reduction will be successful with probability $\frac{1}{2}$. $\square$

For completeness, we mention that the alternative simulation may succeed not by solving the Poly-SDH instance but by finding a hash collision. Since it is possible to build (keyed) collision-resistant hash function families from CDH, and thus from SDH or Poly-SDH, the theorem still holds without the need for a hashing assumption.

C Mesh Scheme Security Proofs

This section focuses on the security properties of the general mesh signature scheme of Section 5.

C.1 Anonymity of the Mesh Scheme

We prove Theorem 5 using an information-theoretic argument.

Proof of Theorem 5. We need to show that it is impossible to determine which assignment caused $\tilde{\Upsilon}$ to be satisfied in the signature σ. By design of the blinding factors, the distribution of σ is the uniform distribution $\mathbf{U}$ over the $(2\,\ell+1)$-dimensional variety of $\mathbb{F}_p^{\ell+1} \times \mathbb{G}^{\ell+\vartheta+1}$ defined by the mesh verification equation, *i.e.*,

$$\mathbf{P}(\sigma) = \mathbf{U}\left(\begin{array}{l} (t_0,\ ...,\ t_\ell,\ S_1,\ ...,\ S_\ell,\ P_0,\ ...,\ P_\vartheta) \in \mathbb{F}_p^{\ell+1} \times \mathbb{G}^{\ell+\vartheta+1}\ : \\ \quad \displaystyle\prod_{k=0}^{\vartheta} \mathbf{e}\left(P_0,\ \hat{A}_{0,k}\,\hat{B}_{0,k}^{m_0}\,\hat{C}_{0,k}^{t_0}\right)^{y_{0,k}} \cdot \prod_{i=1}^{\ell} \mathbf{e}\left(S_i,\ \hat{A}_{i,0}\,\hat{B}_{i,0}^{m_i}\,\hat{C}_{i,0}^{t_i}\right)^{y_{i,0}} = \mathbf{e}(g,\ \hat{g}_0) \\ \wedge \displaystyle\prod_{k=0}^{\vartheta} \mathbf{e}\left(P_1,\ \hat{A}_{0,k}\,\hat{B}_{0,k}^{m_0}\,\hat{C}_{0,k}^{t_0}\right)^{y_{0,k}} \cdot \prod_{i=1}^{\ell} \mathbf{e}\left(S_i,\ \hat{A}_{i,1}\,\hat{B}_{i,1}^{m_i}\,\hat{C}_{i,1}^{t_i}\right)^{y_{i,1}} = 1 \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vdots \\ \wedge \displaystyle\prod_{k=0}^{\vartheta} \mathbf{e}\left(P_\vartheta,\ \hat{A}_{0,k}\,\hat{B}_{0,k}^{m_0}\,\hat{C}_{0,k}^{t_0}\right)^{y_{0,k}} \cdot \prod_{i=1}^{\ell} \mathbf{e}\left(S_i,\ \hat{A}_{i,\vartheta}\,\hat{B}_{i,\vartheta}^{m_i}\,\hat{C}_{i,\vartheta}^{t_i}\right)^{y_{i,\vartheta}} = 1 \end{array}\right),$$

where $\forall i = 0, ..., \ell$ the exponents $y_{i,k}$ come from the polynomial coefficients of $\pi_i = \sum_{k=0}^{\vartheta} y_{i,k}\, Z_k$, whereas $\forall k = 0, ..., \vartheta$ the elements $\hat{A}_{i,k}, \hat{B}_{i,k}, \hat{C}_{i,k}$ come from the public verification keys VK_i.

First, we observe that $\mathbf{P}(\sigma)$ is indeed uniform over the stated $(2\,\ell+1)$-dimensional variety. Indeed, σ lives in a $(2\,\ell+\vartheta+2)$-dimensional space and is subject to $\vartheta+1$ independent linear constraints in the verification algorithm, so it has at most $2\,\ell+1$ degrees of freedom. Conversely, for each joint random assigment to $t_0, ..., t_\ell$ and $s_1, ..., s_\ell$ there exists a distinct valid signature based on that assignment, and so σ has at least $2\,\ell+1$ degrees of freedom, which are then as stated.

Next, we verify that $\mathbf{P}(\sigma)$ is indeed independent of the true signer, and more generally of the secret linear combination $\nu_0, ..., \nu_\ell$ that was used in the creation of σ (conditionally on Υ being satisfied). This is immediate since the polynomials π_i and their coefficients $y_{i,k}$ form a public encoding of Υ determined independently of the truth value assignment χ from which the ν_i derive.

Last, we note that the hash value $Msg_0 = H([VK_1 : Msg_1], ..., [VK_\ell : Msg_\ell], \Upsilon)$ and all ancillary information adjoined to σ is itself a function of public information only. The theorem follows. $\square$

C.2 Unforgeability of the Mesh Scheme

We prove Theorem 6 from the $(q, \ell+1)$-Poly-SDH assumption in $\mathbf{G}$.

Proof of Theorem 6. Let $\mathbf{G} = (p, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_t, g, \hat{g}, \mathbf{e})$ be a bilinear instance with a computable isomorphism $\psi : \hat{\mathbb{G}} \to \mathbb{G}$, and suppose that $H : \{0,1\}^* \to \mathbb{F}_p$ is a collision-resistant hash function.

As before, we are given a random instance of the $(q, \ell+1)$-Poly-SDH problem in $\mathbf{G}$, stated as $\ell+1$ pairs $(g^{\alpha_i}, \hat{g}^{\alpha_i})$ for $i = 0, ..., \ell$, and $(\ell+1)\,q$ pairs $(w_{i,j},\, u_{i,j} = g^{1/\alpha_i + w_{i,j}})$ for $i = 0, ..., \ell$ and $j = 1, ..., q$. Our task is to output $\ell+1$ pairs $(w_i^*,\, u_i^* = g^{r_i^*/\alpha_i + w_i^*})$ for $i = 0, ..., \ell$, for some public choice of w_i^* such that $w_0^* \notin \{w_{0,1}, ..., w_{0,q}\}$, and some secret choice of r_i^* such that $\sum_{i=0}^{\ell} r_i^* = 1 \pmod{p}$.

We construct an algorithm $\mathcal{B}$ that solves such instances of the Poly-SDH problem by interacting with a black-box forger $\mathcal{A}$ for the mesh scheme. For simplicity, we give sequential numbers in $\{1, 2, ...\}$ to all users (that is, to all individuals that can be potentially named as mesh members).

Since the adversary is static, we suppose w.l.o.g. that the target users will consist of the set $\{1, ..., \ell\}$. Per the mesh security model, a successful forgery must bear on a mesh expression Υ over an arbitrary set of users, as long as it logically implies a weaker expression Υ' that involves only the users in $\{1, ..., \ell\}$, and no component that matches an earlier atomic query. Recall that this is to ensure that Υ remains falsifiable even if the adversary sets all the literals under its control to $\top$, lest we accept a trivial forgery. Remark that a maximally weak such Υ' representable in the language is a disjunction over all target users in $\{1, ..., \ell\}$, *i.e.*, the ring signature disjunction $\Upsilon'' = \vee_{i=1}^{\ell} [VK_i : Msg_i]$. Here, the clauses $[VK_i : Msg_i]$ will be the same as in the original forgery, except for those that were the object of earlier atomic queries in which case they are replaced by $[VK_i : Msg'_i]$ for some arbitrary Msg'_i. Observe that Υ'' specifies a ring signature forgery analogous to that in the proof of Theorem 20.

Our strategy will thus be, first, to accept from $\mathcal{A}$ a forgery σ with formula Υ on some arbitrary set of users. Next, we transform σ into a pseudo-signature σ' with a weaker formula Υ' defined as a disjunction of the clauses $[VK_i : Msg_i]$ from Υ that involve no adversarial user and match no prior atomic signature query, *i.e.*, only clauses under the simulator's control are permitted in Υ'; the result σ' is a ring signature that is technically invalid because the clause $[VK_0 : m_0]$ for the "key in the sky" pertains to the original hash value $m_0 = H(..., \Upsilon)$. Then, we transform σ' into an even weaker ring pseudo-signature σ'' whose formula Υ'' is a disjunction over the entire ring $R^* = \{1, ..., \ell\}$ of all honest users: for the latter step we simply add a clause $[VK_i : Msg'_i]$ for each user of index $i \le \ell$ that is missing from Υ'; the messages Msg'_i may be arbitrary and match Msg_i from the forgery, as long as we steer clear from all messages used in atomic queries made to VK_i. From σ'', which still contains the original hash m_0 but would otherwise be a valid ring signature, $\mathcal{B}$ can compute a solution to the Poly-SDH instance exactly as in Theorem 20.

It suffices to show how to respond to well-formed mesh signatures and atomic signature queries during the adaptive query phase, and then how to effect the final transformation $\sigma \mapsto \sigma' \mapsto \sigma''$ corresponding to $\Upsilon \mapsto \Upsilon' \mapsto \Upsilon''$ at the end of the game.

Regular Simulation. We merely show how the regular mesh simulation generalizes that of the basic ring signature.

To start, $\mathcal{B}$ publishes the blinear instance $\mathbf{G}$, the isomorphism ψ, and the hash function H. Furthermore, $\mathcal{B}$ chooses $\lambda + 1$ random exponents $z_0, ..., z_\lambda \in \mathbb{F}_p^\times$, and for each k publishes the reference element $\hat{g}_k = (\hat{g}^{\alpha_0})^{z_k}$, based on the value of $\hat{g}^{\alpha_0}$ given in the Poly-SDH instance. This also induces $g_k = (g^{\alpha_0})^{z_k} = \psi(\hat{g}_k)$, for $k = 0, ..., \lambda$.

Additionally, $\mathcal{B}$ publishes the verification key "in the sky", consisting of $3\,(\lambda+1)$ elements $\hat{A}_{0,k} = (\hat{g}^{\alpha_0})^{a_{0,k}}$, $\hat{B}_{0,k} = \hat{g}^{b_{0,k}}$, $\hat{C}_{0,k} = \hat{g}^{c_{0,k}}$ with random $a_{0,k}, b_{0,k}, c_{0,k} \in \mathbb{F}_p^\times$ for $k = 0, ..., \lambda$. (Notice that the $\hat{A}_{0,k}$ are powers of $\hat{g}^{\alpha_0}$ given in the Poly-SDH instance, while the $\hat{B}_{0,k}$ and $\hat{C}_{0,k}$ are mere powers of $\hat{g}$.)

$\mathcal{B}$ gives $\mathcal{A}$ the public keys of the first ℓ users. To do so, for each $i = 1, ..., \ell$, it draws random exponents $b_i, c_i \in \mathbb{F}_p^\times$ and publishes $\hat{A}_{i,k} = (\hat{g}^{\alpha_i})^{z_k}$, $\hat{B}_{i,k} = \hat{g}^{b_i\, z_k}$, $\hat{C}_{i,k} = \hat{g}^{c_i\, z_k}$ for $k = 0, ..., \lambda$. (Here, the $\hat{A}_{i,k}$ derive from the $\hat{g}^{\alpha_i}$ from the Poly-SDH instance, whereas the $\hat{B}_{i,k}$ and $\hat{C}_{i,k}$ are known powers of $\hat{g}$.)

$\mathcal{A}$ gives $\mathcal{B}$ the public keys of the users it controls, of indices $i = (\ell+1), ..., I_{\max}$, during the course of queries. For each such i, a key consists of $3\,(\lambda+1)$ elements $\hat{A}_{i,k}$, $\hat{B}_{i,k}$, $\hat{C}_{i,k}$, for $k = 0, ..., \lambda$.

It must be the case that $\mathrm{dlog}_{\hat{g}_0}(\hat{A}_{i,0}) = ... = \mathrm{dlog}_{\hat{g}_\lambda}(\hat{A}_{i,\lambda})$, and similarly for the $\hat{B}_{i,k}$ and the $\hat{C}_{i,k}$, which the simulator can easily verify using the pairing.

$\mathcal{A}$ makes $q_S = q$ distinct mesh signatures queries to $\mathcal{B}$, one at a time, proceeding adaptively.

Each query is a well-formed mesh statement Υ to be signed by any coalition of users that can satisfy Υ. Let n be the number of mesh literals $L_1, ..., L_n$, and let the corresponding clauses be $[VK_{i_1}\colon m_1], ..., [VKi_n\colon m_n]$. We require that Υ be unsatisfiable using only clauses with user indices $i_j > \ell$, otherwise the adversary should be able to create a signature without having to query for it. (We remark that the simulator is still able to respond even in this case, using the signing key "in the sky", but chooses not to do so.)

The adversary may also specify, as part of the query, an atomic signature (u_j, t_j) on m_j for any number of users $i_j > \ell$ in its control. This is to simulate the scenario where the signer wishes to make use of atomic signatures on messages it knows, *e.g.*, PKI certificates on designated users' keys, but lacks the ability to create different signatures on those messages. (In the simulation, $\mathcal{B}$ simply ignores the given u_j and uses the t_j instead of random values.)

$\mathcal{B}$ responds to a well-formed j-th query very much as in the ring signature simulation of Theorem 20. The differences are as follows.

First, $\mathcal{B}$ creates $\tilde{\Upsilon} = \Upsilon \vee L_0$ where L_0 corresponds to the proposition $[VK_0\colon m_0]$ with $m_0 = H([VK_1\colon Msg_1], ..., [VK_n\colon Msg_n], \Upsilon)$, per the mesh scheme. $\mathcal{B}$ computes a representation of $\tilde{\Upsilon}$ as a list of degree-1 polynomials $\pi_0, ..., \pi_n \in \mathbb{F}_p[Z_0, ..., Z_\vartheta]$ with coefficients $y_{i,k} \in \mathbb{F}_p$.

Next, $\mathcal{B}$ satisfies $\tilde{\Upsilon}$ with a truth assignment such that $\chi(L_0) = \top$ and $\chi(L_i) = \bot$ everywhere else. Accordingly, $\mathcal{B}$ uses the $(0,j)$-th pair $(w_{0,j},\, u_{0,j})$ from the Poly-SDH instance to obtain an atomic signature on m_0 under a combination of the keys "in the sky" expressed by the polynomial π_0. Precisely, it defines $a_0 = \sum_{k=0}^{\vartheta} a_{0,k}\, y_{0,k}$, $b_0 = \sum_{k=0}^{\vartheta} b_{0,k}\, y_{0,k}$, $c_0 = \sum_{k=0}^{\vartheta} c_{0,k}\, y_{0,k}$, and builds the signature as: $(u_0 = {u_{0,j}}^{1/a_0},\ t_0 = \frac{a_0\, w_{0,j} - b_0\, m_0}{c_0})$. [2] [3]

[2]Notice that, unlike the user keys which have a lot of internal structure (exhibited by the many obvious discrete log relations), the various components of the key "in the sky" are independently distributed and so a signature that will verify under one triple $(\hat{A}_{0,k_1}, \hat{B}_{0,k_1}, \hat{B}_{0,k_1})$ will not verify under another $(\hat{A}_{0,k_2}, \hat{B}_{0,k_2}, \hat{B}_{0,k_2})$. What we need is a signature that will verify for the particular combination of such triples given by the coefficients of π_0.

[3]It can also be shown that for $\tilde{\Upsilon} = \Upsilon \vee L_0$, the algorithm of Section 5.1 always gives $\pi_0 = Z_0$, and so we have

Then, $\mathcal{B}$ runs the mesh signing algorithm as in the real scheme, based on the π_i and ν_i it has. Specifically, $\mathcal{B}$ takes t_0, chooses $t_1, ..., t_n$ at random (or uses the $\mathcal{A}$-specified values for them), and computes $S_{i_1}, ..., S_{i_n}$ and $P_0, ..., P_\vartheta$ as in the real scheme. It can do so without knowing any signing key because $\nu_i = 0$ for all users $i \neq 0$; however, the mesh prototype it obtains is invalid precisely for that reason. (The only non-zero coefficient in the linear combination is the coefficient ν_0 that applies to the polynomial π_0, but the signing algorithm of Section 5.3 purposely ignores it since in real life it is always zero. For $\tilde{\Upsilon} = \Upsilon \vee L_0$, the algorithm of Section 5.1 gives $\pi_0 = Z_0$, and thus $\nu_0 = 1$ per Lemma 3 since all other coefficients are zero. The simulator needs to incorporate ν_0 manually into the prototype signature.)

Thus, $\mathcal{B}$ has to amend the prototype to turn it into a valid mesh signature. This is done by multiplying u_0 into the component P_0. Indeed, because the prototype so far contains only blinding factors and no actual signature, the left-hand sides of all the verification equations resolve to $1 \in \mathbb{G}_t$, including the main equation (the one involving P_0), which should equate to $\mathbf{e}(g, \hat{g}_0) \in \mathbb{G}_t$ instead. A substitution of $u_0^{\nu_0} P_0 = u_0 P_0$ for the prototype's P_0 effects the desired correction without disturbing the other equations.

The resulting mesh signature is given to $\mathcal{A}$. It is valid, and uniformly distributed over the correct space, as we count $2n + 1$ degrees of freedom among its $2n + \vartheta + 1$ components.

$\mathcal{A}$ also makes $q_{S,i} = q$ distinct atomic signature queries for each user $i = 1, ..., \ell$ controlled by $\mathcal{B}$; these can be arbitrarily interleaved with the ring signature queries.

$\mathcal{B}$ responds to the j-th query (i, m) to the i-th user, by retrieving the (i, j)-th pair $(w_{i,j},\ u_{i,j})$ from the Poly-SDH instance, computing $t = (w_{i,j} - b_i\, m)/c_i$, and returning the Boneh-Boyen signature $(u_{i,j},\ t)$ to $\mathcal{A}$.

$\mathcal{A}$ finally outputs a forgery $\sigma = (t_0, ..., t_n, S_1, ..., S_n, P_0, ..., P_\vartheta)$ bearing on a well-formed mesh statement Υ such that $\Upsilon \Rightarrow \Upsilon'$ for some other well-formed statement Υ' that involves only users of indices $i \leq \ell$ and no clause that matches an earlier atomic query. W.l.o.g., we can assume that Υ' is a disjunction, since every well-formed formula in the language can be weakened into a disjunction of its inputs.

$\mathcal{B}$ performs the first transformation $\Upsilon \mapsto \Upsilon'$ by eliminating from σ the components of user indices $i > \ell$, or those that match an atomic query, which will produce a ring pseudo-signature σ' on the disjunction Υ'. The process amounts to performing Gaussian elimination of every variable S_i that corresponds to an undesirable clause, in the linear system that lives "in the exponents" of the verification equations. Each step of the Gaussian elimination will "consume" exactly one of the ϑ supplemental verification equation (*i.e.*, those that involve some P_k for $k \neq 0$), with the aim of eliminating one of the remaining offending S_i, thus chosen as our "pivot". As in classical Gaussian elimination, only an S_i whose coefficient $y_{i,k} \neq 0$ at the end of a previous step can serve as pivot and hence be eliminated by consuming the next equation in P_k. Remark that we do not eliminate P_0. A straightforward argument shows that if $\Upsilon \Rightarrow \Upsilon'$ where Υ' represents a disjunction over $L_0, ..., L_\ell$, then a pseudo-signature corresponding to Υ' can be obtained by the Gaussian elimination process ("pseudo" because, once again, the hash value m_0 is not updated in the process). We now describe Gaussian elimination "in the exponent".

$y_{0,k} = 0$ for $k \neq 0$. It follows that the public key "in the sky" only needs one triple $(\hat{A}_{0,0}, \hat{B}_{0,0}, \hat{B}_{0,0})$ instead of $\lambda + 1 \geq \vartheta + 1$ of them. We omit this optimization from the present proof to avoid further complicating the argument, but since it greatly shortens the CRS we explicitly recommend its use in Section 5.5.

Suppose w.l.o.g. that $\mathcal{B}$ seeks to eliminate S_n from σ, where S_n appears with exponent $y_{n,\vartheta} \neq 0$ in the equation that involves P_ϑ, *i.e.*,

$$\mathbf{e}\left(P_\vartheta,\ \hat{v}_0\right) \cdot \prod_{i=1}^{n} \mathbf{e}\left(S_i,\ \left(\hat{A}_{i,\vartheta}\, \hat{B}_{i,\vartheta}^{m_i}\, \hat{C}_{i,\vartheta}^{t_i}\right)^{y_{i,\vartheta}}\right) = 1\ .$$

The idea is to find a linear combination that will cancel out S_n in the main equation, *i.e.*,

$$\mathbf{e}\left(P_0,\ \hat{v}_0\right) \cdot \prod_{i=1}^{n} \mathbf{e}\left(S_i,\ \left(\hat{A}_{i,0}\, \hat{B}_{i,0}^{m_i}\, \hat{C}_{i,0}^{t_i}\right)^{y_{i,0}}\right) = \mathbf{e}(g,\ \hat{g}_0)\ .$$

This is easy to do once we observe that $(\hat{A}_{i,\vartheta}\, \hat{B}_{i,\vartheta}^{m_i}\, \hat{C}_{i,\vartheta}^{t_i})^{z_0} = (\hat{A}_{i,0}\, \hat{B}_{i,0}^{m_i}\, \hat{C}_{i,0}^{t_i})^{z_\vartheta}$ for all $i = 1, ..., n$. We raise both sides of the P_ϑ equation (on top) to the power of $\rho = \frac{y_{n,0}\, z_0}{y_{n,\vartheta}\, z_\vartheta}$, and divide the result into the main equation, causing the pairing with S_n to vanish. We then consolidate the new terms into the existing ones to preserve the form of the equation. That is, we let $P_0' = P_0 / P_\vartheta^\rho$ replace the ratio of P_0 and P_ϑ^ρ, and for all $i = 1, ..., n-1$, we merge the ratio of $\mathbf{e}\left(S_i,\ (\hat{A}_{i,0}\, \hat{B}_{i,0}^{m_i}\, \hat{C}_{i,0}^{t_i})^{y_{i,0}}\right)$ and $\mathbf{e}\left(S_i,\ (\hat{A}_{i,\vartheta}\, \hat{B}_{i,\vartheta}^{m_i}\, \hat{C}_{i,\vartheta}^{t_i})^{y_{i,\vartheta}}\right)^\rho = \mathbf{e}(S_i,\ (\hat{A}_{i,0}\, \hat{B}_{i,0}^{m_i}\, \hat{C}_{i,0}^{t_i})^{\rho \frac{z_\vartheta}{z_0} y_{i,\vartheta}})$, into the one pairing $\mathbf{e}\left(S_i,\ (\hat{A}_{i,0}\, \hat{B}_{i,0}^{m_i}\, \hat{C}_{i,0}^{t_i})^{y'_{i,0}}\right)$ by letting $y'_{i,0} = y_{i,0} - \rho \frac{z_\vartheta}{z_0} y_{i,\vartheta} = y_{i,0} - \frac{y_{n,0}}{y_{n,\vartheta}} y_{i,\vartheta}$. Notice that $y'_{n,0} = 0$ is evidence that S_n has vanished from the equation.

We similarly eliminate S_n from each of the remaining verification equations, for $k = 1, ..., \vartheta - 1$, using the same technique. This produces new points P_k' to replace the old P_k, as well as new exponents $y'_{i,k}$ for all $i = 0, ..., n-1$ to replace the $y_{i,k}$.

The signature σ has been transformed into $(t_0, ..., t_{n-1}, S_1, ..., S_{n-1}, P_0', ..., P_{\vartheta-1}')$, which has three fewer components.

After ϑ iterations of this process, we obtain a signature $(t_0, ..., t_{n-\vartheta}, S_1, ..., S_{n-\vartheta}, P_0'')$, which corresponds to a mesh statement Υ' which is a flat disjunction of $n - \vartheta + 1$ inputs. We have thus obtained a ring signature over a ring of size $n - \vartheta$ plus the user "in the sky". We can rename the component P_0'' as S_0 and rearrange the signature into the familiar form $\sigma' = (S_0, ..., S_{n-\vartheta}, t_0, ..., t_{n-\vartheta})$, which represents a disjunction, and should only contain target users and honest clauses if the original forgery was admissible in the first place.

$\mathcal{B}$ then performs the second transformation $\Upsilon' \mapsto \Upsilon''$. Starting from σ', it expands the ring to cover all ℓ users, by adding dummy signatures by the missing users on arbitrary messages: this is done by adding S_j components for randomly chosen t_j. Since these dummy signature components need not (and must not) contribute to the final output, they consist only of a blinding factor that is easy to cancel in an existing S_i, using the usual reciprocity trick under the pairing. We finally obtain a ring pseudo-signature $\sigma'' = (t_0, ..., t_\ell, S_0, ..., S_\ell)$ for the full ring $R^* = \{1, ..., \ell\}$ of honest users plus the verification key "in the sky".

The final step of the reduction is to turn σ'' into a solution to the Poly-SDH problem. We omit the description of this step as it is exactly the same as in the regular simulation in Theorem 20, once $\mathcal{B}$ has removed whichever (known) exponent $y''_{i,0}$ still remains in each S_i.

We already mentioned that the ring pseudo-signature σ'' is technically invalid since the message m_0 beared by the key "in the sky" is the hash value $m_0 = H([VK_1 \colon Msg_1], ..., [VK_n \colon Msg_n], \Upsilon)$ from the original mesh specification Υ, and not Υ''. This does not affect the final reduction.

However, just as in the proof of Theorem 20, for some $i \in \{0, ..., \ell\}$ the forgery message m_i could induce a value $w_i^* \in \{w_{i,1}, ..., w_{i,q}\}$ that was already given in the Poly-SDH instance (using the w_i^* notation from Theorem 20). If the component S_i can be eliminated by the above process, then all is well. Otherwise, we need an alternative reduction from the same mold as in Theorem 20.

Alternative Simulation. The alternative reduction works only on a final forgery that "matches" one of the Poly-SDH pairs that was given to the simulator, in the sense that it corresponds to a value $w_0^* \in \{w_{0,1}, ..., w_{0,q}\}$. In this case, the simulator can either recover the Poly-SDH secret exponent α_0, or find a collision under the hash function H.

The simulation proceeds as in the ring signatures of Theorem 20, based on the same judicious choice of known and unknown discrete logarithms to let the final reduction go through. As in the regular simulation above, there are two main difficulties compared to the ring signature case:

1. We need to answer queries not for ring signatures but for more complicated mesh signatures. This is easy to do by using the signing key "in the sky" and the method described in the regular mesh simulation above, except for how the atomic signature of index 0 is created. Here, the simulator knows the discrete logarithm of A_0 and B_0 instead of B_0 and C_0, and so the atomic signature is constructed as in the altenative simulation for ring signatures.

   Queries on atomic signatures for the ℓ honest users are answered using the lists of pairs for values of $i = 1, ..., \ell$, as the altenative simulator in the ring proof.

2. We need to transform the final mesh forgery (provided it is admissible) into a ring forgery that exactly covers the target users $\{1, ..., \ell\}$. The transformation operates in two steps analogous to those of the regular mesh simulation, except for a minor modification: in order for the final reduction to work, the simulator must arrange to know the discrete logarithms of the private key components $\hat{A}_i$ and $\hat{B}_i$, instead of $\hat{B}_i$ and $\hat{C}_i$. However, this does not affect the transformation, which exploits a different set of discrete logarithms, namely the z_k.

Success Probability. The important point that justifies that either the regular or the alternative simulation will give the desired result, is that in both cases the final reduced ring signature σ'' still contains a component of index 0 that bears on a hash m_0 of the full unretouched mesh specification Υ given by the forger (and with the pristine original randomizer t_0, too). Since it is those values and the given Poly-SDH instance that determine which of the regular and alternative simulations will work, we have the desired perfect dichotomy.

We conclude that if $\mathcal{B}$ chooses one at random at the onset, the overall success probability of the reduction (conditionally on $\mathcal{A}$'s success) will be $\frac{1}{2}$. □