Mesh Signatures

How to Leak a Secret with Unwitting and Unwilling Participants

Xavier Boyen

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

Abstract. We define the mesh signature primitive as an anonymous signature similar in spirit to ring signatures, but with 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 complete certificate chains. Because withholding one's public key from view is no longer a shield against being named as a possible cosignatory, mesh signatures may be used as a ring signature with compulsory enrollment.

We give an efficient construction based on bilinear maps in the common random string model. Our signatures have linear size, achieve everlasting perfect anonymity, and reduce to very efficient ring signatures without random oracles as a special case. 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 [?], but overcome some of their crucial limitations.

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 thus 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 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 nodea are And, Or, and Threshold gates, and whose leaves are regular signatures. The access structure can be satisfied using different subsets of the regular signatures; once created, the mesh signature will not reveal what particular subset was used. The regular signatures at the leaves can be "static", and thus PKI certificates are eligible if the mesh signer does not have the CA's signing key. Since furthermore the monotone tree structure is powerful enough to express disjunctions of certificate chains, we are no longer dependent on individual ring members publishing their keys. As a toy example, suppose that *Alice* wants to implicate *Bob*, who may or may not have a verification key on record. *Alice* can still produce the following mesh signature,

$$\sigma = [VK_{Alice} \colon Msg_1] \text{ or } ([VK_{CertAuth} \colon VK_{Bob}] \text{ and } [VK_{Bob} \colon Msg_2]) \ ,$$

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, the mesh signature σ implicates him via the certificate $[VK_{CertAuth} \colon VK_{Bob}]$ that binds his name to the string VK_{Bob}; the certificate can be real or a fake. Conversely, *Bob* could have created σ himself, using the real certificate and his own private key, to implicate *Alice*; although in this case her public key would have to be available to the verifier since her certificate is not part of σ. Another feature of mesh signatures is that they provide threshold gates, which makes it easy to scale constructs like,

$$\sigma = \text{2-out-of-3 in } \{[\textsc{ceo} \colon \textit{skrt-memo}] \, , \, [\textsc{cfo} \colon \textit{skrt-memo}] \, , \, [\textsc{coo} \colon \textit{skrt-memo}]\} \ .$$

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 constructible non-interactively from reusable constituent atomic signatures (in our case, these are Boneh-Boyen short signatures [?]).

1.1 Related Work

The original ring signature primitive was defined in [?], to enable secret leaking that is at once authenticated (by a crowd) and anonymous (within the crowd). Whereas that construction [?] was based on trapdoor permutations, a number of alternatives have subsequently been proposed, based on bilinear pairings [?], discrete logarithms [?], factoring (Strong-RSA specifically) [?], or hybrids [?]; all these constructions are set in the random oracle model. Most have linear size in the ring membership count, except [?] 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 [?] 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 [?], that combines the anonymity of ring signatures with the non-transferability of deniable authentication [?], and supports threshold and access structures. Among specific constructions in the random oracle model, we note the distributed ring signatures of [?] which lets coalitions of users cooperate in an interactive signing protocol, and the hierarchical identity-based ring signatures of [?], which adds 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 [?], 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 [?], 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 [?]. There exists efficient constant-size group signature schemes, with random oracles [?], from interactive assumptions [?], and in the standard model [?].

Efficient ring signatures constructions without random oracles have also been proposed recently, such as [?], [?], and [?]. The construction of [?] uses bilinear groups and is efficient but relies on a cumbersome assumption stated without justification. The results of [?] include an impractical scheme from non-interactive Zaps [?], but also two efficient constructions (based on [?] or [?] 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 [?] which can efficiently creates linear-size ring signatures in the "trusted parameters" model; unforgeability is based on computational Diffie-Hellman, and anonymity on the decisional Subgroup [?] 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 [?] and base anonymity on the decisional Linear [?] 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 [?] 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 for the stated message and 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):

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

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 $\Upsilon(\bot, ..., \bot) = \bot$ 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.*

We emphasize that two distinct symbols L_i and L_j can express the same sentence and yet have opposite truth values, since the signer is free to use a valid atomic signature for one and not for the other. The current construction does not support cloning truth values without losing the original, just as it cannot express the negation of a truth value.

2.2 Anonymity Model

The strongest notion of anonymity defined in [**?**], "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 any long-term secret held by any party in the system. We exclude ephemeral randomness from the above requirement, 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.

2.3 Unforgeability Model

The strongest notion of unforgeability defined in [?], "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, this is overly restrictive, since it excludes forgeries such 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. We capture these circumstances by deeming admissible any forgery on a statement Υ if there exists a well-formed Υ' that involves only honest users and such that $\Upsilon \Rightarrow \Upsilon'$.

To see where this comes from, for all corrupted users let us set the corresponding literal $L_i \leftarrow \top$, which is the most that they can supposedly do. If Υ evaluates to $\top$, the forgery is inadmissible; otherwise, Υ reduces to some well-formed formula Υ' which involves honest users, exclusively. Hence, the condition 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.
- **Signature queries:** the adversary makes up to q mesh signature queries, one at a time, on specifications Υ_j whose satisfiability involves the challenger's users.

The adversary may also query q atomic signatures to each of the users controlled by the challenger (since atomic signatures should be usable instead of signing keys for mesh signing.)

The challenger processes each request before accepting the next one.

Signature forgery: the adversary produces a forged signature whose specification Υ contains no clause $[VK_i : Msg_i]$ from an atomic query, and is such that $\forall j, \Upsilon \neq \Upsilon_j$ and $\exists \Upsilon', \Upsilon(L_1, ..., L_\ell, ...) \Rightarrow \Upsilon'(L_1, ..., L_\ell)$ where Υ' is a well-formed formula with honest user clauses only.

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 [**?**]. 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 [**?**], 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 [**?**] 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 [**?**] 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 [**?**] 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 by checking,

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

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,

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

To decide whether a solution $((w_1,\ u_1),\ ...,\ (w_\ell,\ u_\ell))$ to the new problem is correct, one also 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 $1 \le i \le \ell$ and $1 \le j \le q$, 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 $\sum_i r_i = 1$, without seeing 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 a few justifications for this:

- the modified form results in a weaker assumption (by analogy to the implication from Original SDH to Modified SDH);
- the input/output symmetry simplifies the security reductions;
- its instances are more concisely stated when more than one iterator is needed;
- the modified problem form is impervious to a (benign) generic analysis described in [**?**], 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 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 [**?**] 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 [**?**], 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 $g^{\frac{\pi(\alpha_1,\ldots,\alpha_\ell)}{\Delta}}$, where $\pi \in \mathbb{F}_p[\alpha_1, ..., \alpha_\ell]_{q\ell+1}$ is any multivariate polynomial in $\alpha_1, ..., \alpha_\ell$ 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, ..., $\pi_\ell \in \mathbb{F}_p[\alpha_1, ..., \alpha_\ell]_{q\ell+1}$ 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 $i = 1, ..., \ell$ and $j = 1, ..., q$, 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 $\leq q\,\ell + 1$ in the variables $\alpha'_1, ..., \alpha'_\ell$, 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 $\alpha'_1, ..., \alpha'_\ell$ 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 $\mathbb{F}_p[\alpha'_1, ..., \alpha'_\ell]$ or $\mathbb{F}_p[\alpha_1, ..., \alpha_\ell]$, 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 takes $q_G = \Omega(\sqrt{\epsilon\, p / q\, \ell})$ operations to solve (q, ℓ)-Poly-SDH with probability ϵ in generic groups of order p.

4 Special Case : Ring Signatures

We first describe a ring signature based on the Poly-SDH assumption as a special case of our technique. It is more efficient than other provably secure ring signature schemes without random oracles, and is set in the common random string model without trusted parameters.

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: User $\#i$ draws a signing key $(a_i,\, b_i,\, c_i) \in (\mathbb{F}_p^\times)^3$, and publishes $(A_i,\, B_i,\, C_i,\, \hat{A}_i,\, \hat{B}_i,\, \hat{C}_i) = (g^{a_i},\, g^{b_i},\, g^{c_i},\, \hat{g}^{a_i},\, \hat{g}^{b_i},\, \hat{g}^{c_i}) \in \mathbb{G}^3 \times \hat{\mathbb{G}}^3$.
In case $\psi : \hat{\mathbb{G}} \to \mathbb{G}$ is easy to compute, users publish only $(\hat{A}_i,\, \hat{B}_i,\, \hat{C}_i)$.

Ring signature: To create a ring signature on a message $m \in \mathbb{F}_p$ 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 $\sigma = (S_0,\ ...,\ S_\ell,\ t_0,\ ...,\ t_\ell) \in \mathbb{G}^{\ell+1} \times \mathbb{F}_p^{\ell+1}$, given by,

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

with $m_1, ..., m_\ell$ 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 $\sigma = (S_1,\ ...,\ S_\ell,\ t_1,\ ...,\ t_\ell)$ with respect to a message m and a ring $R = (1, ..., \ell)$, it suffices to set $m_1 = ... = m_\ell = m$ and $m_0 = H((1, m_1), ..., (\ell, m_\ell))$, and test the equality,

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

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 traditional ring signatures.

The purpose of signing a hash of the message and ring composition under 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. It also helps in the security proof.

We emphasize that the public string K has no hidden structure, and can be drawn publicly at random as long as it is not chosen to grant anyone undue advantage to compute discrete logarithms in $\mathbf{G}$ or sign under the key "in the sky". The absence of secret coins is the main difference between a common random string (CRS) and the much more demanding trusted parameters (TP) model: in the former the parameters can be drawn in the open; in the latter they must be crafted in a special way from secret coins by some trusted setup authority, who must then voluntarily give up the secret knowledge it has (and convince everyone that it did not cheat). No trusted setup agent, either centralized or distributed, is needed in our system.

Furthermore, we have irrevocable, or everlasting, unconditional anonymity of the signatures (*i.e.*, with forward security against coerced disclosure of the long-term signing keys), as stated by the following theorem.

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

The second security theorem states that the scheme is existentially unforgeable in the model of Section **??**. The proofs will appear in the full version.

Theorem 2. *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.*

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 both requirements [**?**]. This is done as follows.

- 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: $\sigma = (\hat{S}_0, \ ..., \ S_\ell, \ t_0, \ ..., \ t_\ell) \in \hat{\mathbb{G}} \times \mathbb{G}^\ell \times \mathbb{F}_p^{\ell+1}$, given by,

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

- 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.

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: $\Upsilon ::= N$ and,

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

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 [**?**,**?**].

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

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

1. Initialize a counter $k_c \leftarrow 0$.
   The counter k_c is used for allocating new variables, so that each Z_{k+k_c} is always a "fresh" variable that is never used before or after in the algorithm.
2. Label the root node N_0 with the polynomial $\pi_{N_0} \leftarrow Z_0$.
3. Select a non-leaf node N with non-empty label $\pi_N \neq \emptyset$.
   (a) Denote by $N_1, ..., N_m$ the $m \geq 2$ children of N.
   (b) If N is $\vee\{N_1, ..., N_m\}$, then $\forall i = 1, ..., m$ let $\pi_{N_i} = \pi_N$.
   (c) If N is $\wedge\{N_1, ..., N_m\}$, then $\forall i = 1, ..., m$ let $\pi_{N_i} = \pi_N + \sum_{k=1}^{m-1} l_{i,k}\, Z_{k+k_c}$ where $l_{i,k} \in \mathbb{Z}$. The selection of $l_{i,k}$ is explained below.
   (d) If N is $\geq_t\{N_1, ..., N_m\}$, then $\forall i = 1, ..., m$ let $\pi_{N_i} = \pi_N + \sum_{k=1}^{t-1} l_{i,k}\, Z_{k+k_c}$ where $l_{i,k} \in \mathbb{Z}$.
   (e) Label each child N_i with the polynomial π_{N_i}.
   (f) Unlabel node N, *i.e.*, set $\pi_N \leftarrow \emptyset$.
   (g) Increment $k_c \leftarrow k_c + t - 1$ (using $t = 1$ and $t = m$ for $\vee$- and $\wedge$-gates).
   (h) Continue at Step **??** if an eligible node remains, otherwise skip to Step **??**.
4. Let $\vartheta \leftarrow k_c$ and output the polynomials $(\pi_0, ..., \pi_\ell)$ associated with the leaf nodes $L_0, ..., L_\ell$. Each polynomial π_i is represented as a vector of coefficients $(y_{i,0}, ..., y_{i,\vartheta}) \in \mathbb{F}_p^{\vartheta+1}$ such that $\pi_i = \sum_{k=0}^{\vartheta} y_{i,k}\, Z_k$ is the result of the sequence of operations in Steps **??**, **??** and **??**.

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

In Steps **??** and **??**, 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 [**?**] for Linear Secret Sharing Structures, and proven by induction on the structure of $\tilde{\Upsilon}$. We refer to the literature [**?**] for further details.

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

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

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 **??**: 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 perfect hiding.

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

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

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

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

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

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: On input the following mesh signature specification:

- ℓ atomic signature specifications $[VK_i\colon Msg_i]$, not necessarily all distinct, and ℓ boolean flags L_i, for $i = 1, ..., \ell$;
- a well-formed formula Υ with ℓ boolean inputs; 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 $\mathbf{G}$ on the statement $[VK_i\colon Msg_i]$, given as a pair,

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

where $w = Msg_i$ and (a, b, c) is the signing key for the clause $[VK_i\colon Msg_i]$.

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

1. Compute $Msg_0 = H([VK_1\colon Msg_1], ..., [VK_\ell\colon Msg_\ell], \Upsilon)$ by hashing the mesh specification, and associate the literal L_0 to the clause $[VK_0\colon Msg_0]$.
2. Construct $\tilde{\Upsilon} = L_0 \vee \Upsilon$, which is well-formed per the definition.
3. Extend χ so that $\chi(L_0) = \bot$, as we lack an atomic signature for L_0.

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:

4. Create a flattened representation of $\tilde{\Upsilon}$ and χ as discussed in Section **??**. 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 $\boldsymbol{s} = (s_1, ..., s_\ell) \in \mathbb{F}_p^\ell$ as in Section **??**.
6. $\forall i \in \{0, ..., \ell\} : \chi(L_i) = \bot$, pick $t_i \in \mathbb{F}_p$ and fix $u_i = g^0 = 1 \in \mathbb{G}$.

7. For all $j = 0, ..., \ell$ and $k = 0, ..., \vartheta$, let $m_j = \mathit{Msg}_j$ and calculate,

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

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

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

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

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:
- $\ell + 1$ propositions $[\mathit{VK}_0 : \mathit{Msg}_0], ..., [\mathit{VK}_\ell : \mathit{Msg}_\ell]$ viewed as inputs to,
- an arborescent monotone threshold circuit $\tilde{\Upsilon} : \{\bot, \top\}^{\ell+1} \to \{\bot, \top\}$,
- 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 subcircuit $\Upsilon(L_1, ..., L_\ell)$ such that $\tilde{\Upsilon} = \Upsilon \vee L_0$, and verify that $\mathit{Msg}_0 = H([\mathit{VK}_1 : \mathit{Msg}_1], ..., [\mathit{VK}_\ell : \mathit{Msg}_\ell], \Upsilon)$.
2. Compute the representation $(\pi_0, ..., \pi_\ell)$ of the formula $\tilde{\Upsilon}$ by reproducing the deterministic conversion of Section **??**.
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, let $m_i = \mathit{Msg}_i$, and calculate,

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

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 if and only if all $\vartheta + 1$ 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, replace Step **??** onward by the following:

??′. Set $d_0 = 1$, pick random $d_1, ..., d_\vartheta \in \mathbb{F}_p$, and verify the single equality,

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

??$'$. Accept the signature as valid if and only if the equality holds in $\mathbb{G}_t$.

Theorem 3. *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 **??** 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 4. *The mesh signature has everlasting perfect anonymity.*

Theorem 5. *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.*

Optimization. We note that the user keys and the key "in the sky" can be shortened significantly. It turns out that in the proofs the b_i are always known to the simulators and are thus superfluous: we can set $b_i = 1$ and omit the $\hat{B}_{i,k} = \hat{g}_k^{b_i} = \hat{g}_k$ from the keys. This holds in the ring scheme, too.

We can independently 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 **??** always gives $\pi_0 = Z_0$, *i.e.*, $y_{0,0} = 1$ and $y_{0,k} = 0$ for $k \neq 0$, meaning that the tuples $(\hat{A}_{0,k}, ..., \hat{C}_{0,k})$ for $k \neq 0$ are in fact never used. Furthermore, it is safe to set $\hat{B}_{0,0} = \hat{g}$, which leaves just the 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 an efficient ring signature without random oracles as a special case of our construction.

Acknowledgements

The author wishes to express his gratitude to Anna Lysyanskaya 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/`.