Attribute-Based Signatures

Hemanta K. Maji* Manoj Prabhakaran* Mike Rosulek†

Abstract

We introduce *Attribute-Based Signatures (ABS)*, a versatile primitive that allows a party to sign a message with fine-grained control over identifying information. In ABS, a signer, who possesses a set of attributes from the authority, can sign a message with a predicate that is satisfied by his attributes. The signature reveals no more than the fact that a single user with some set of attributes satisfying the predicate has attested to the message. In particular, the signature hides the attributes used to satisfy the predicate and any identifying information about the signer (that could link multiple signatures as being from the same signer). Furthermore, users cannot collude to pool their attributes together.

We give a general framework for constructing ABS schemes, then show several practical instantiations based on groups with bilinear pairing operations, under standard assumptions. We describe several practical problems that motivated this work, and how ABS can be used to solve them.

*Department of Computer Science, University of Illinois, Urbana-Champaign. {hmaji2,mmp}@uiuc.edu. Supported by NSF grants CNS 07-16626 and CNS 07-47027.

†Department of Computer Science, University of Montana. mikero@cs.umt.edu.

1 Introduction

Alice, a finance manager in a big corporation, while going through her company's financial records, has learned about a major international scandal. She decides to send these records to a major newspaper, retaining her anonymity, but with a proof that she indeed has access to the records in question. It turns out that several people, due to a combination of reasons, may have access to these records: those in the New York, London or Tokyo office who are either finance managers associated with project Skam, or internal auditors. Alice considers using a *ring signature* [28] to endorse her message anonymously, but realizes that it is infeasible not only because of the large number of people involved, but also because she does not know who these people are. She realizes she cannot use a *group signature* [15] either, because the set of people Alice needs to refer to here is idiosyncratic to her purposes, and may not have been already collected into a group.[1] She is also aware of *mesh signatures* [10], but mesh signatures provide no way to convince the newspaper that the financial record was endorsed by a single person, not, say, a programmer in the New York office colluding with an internal auditor in the Smalltown office.

Alice's needs in this story reflect the challenges in a system where the roles of the users depend on the *combination* of attributes they possess. In such systems, users obtain multiple attributes from one or more *attribute authorities*, and a user's capabilities in the system (e.g., sending or receiving messages, access to a resource) depend on their attributes. While offering several advantages, attribute-based systems also present fundamental cryptographic challenges. For instance, suppose Alice wants to simply send a message to the above group of people using an "attribute-based messaging" system; then to provide *end-to-end* secure communication, it must be possible for her to encrypt a message using attribute-keys (rather than individual users' keys). Recently cryptographic tools have emerged to tackle some of these challenges for encryption [31, 18, 3, 34]. In this work, we provide a solution for authentication, which among other things, will let Alice in the above example leak the financial records anonymously, but with the appropriate claim regarding her credentials.

Why attribute-based signatures?

The kind of authentication required in an attribute-based system differs from that offered by digital signatures, in much the same way public-key encryption does not fit the bill for attribute-based encryption. An attribute-based solution requires a richer semantics, including anonymity requirements, similar to signature variants like group signatures [15], ring signatures [28], and mesh signatures [10]. The common theme in all these signature primitives is that they provide a guarantees of *unforgeability* and *signer anonymity*. A valid signature can only be generated in particular ways, but the signature does not reveal any further information about which of those ways was actually used to generate it.

More specifically, group and ring signatures reveal only the fact that a message was endorsed by one of a list of possible signers. In a ring signature, the list is public, chosen by the signer *ad hoc*, and given explicitly. In a group signature, the group must be prepared in advance by a group manager, who can revoke the anonymity of any signer. In mesh signatures, a valid signature describes an access structure and a list of pairs (m_i, vk_i), where each vk_i is the verification key of a standard signature scheme. A valid mesh signature can only be generated by someone in posession of enough standard signatures σ_i, each valid under vk_i, to satisfy the given access structure.

In this work we introduce *attribute-based signatures (ABS)*. Signatures in an ABS scheme describe a message and a predicate over the universe of attributes. A valid ABS signature attests to the fact that "a single user, whose attributes satisfy the predicate, endorsed the message." We emphasize the word "single" in this informal security guarantee; ABS signatures, as in most attribute-based systems, require

[1] Even if a group exists, the group manager could identify Alice as the informant.

that colluding parties not be able to pool their attributes together.[2] Furthermore, attribute signatures do not reveal more than the claim being made regarding the attributes, even in the presence of other signatures.

Ring and group signatures are then comparable to special cases of ABS, in which the only allowed predicates are *disjunctions* over the universe of attributes (identities). Only one attribute is required to satisfy a disjunctive predicate, so in these cases collusion is not a concern. As in ring signatures, ABS signatures use *ad hoc* predicates. Mesh signatures allow more fine-grained predicates, but do not provide hiding of signature data that would be needed in an ABS scheme. A straight-forward application of mesh signatures as an ABS scheme would either allow collusion (as in the previous example, a New York programmer colluding with a Smalltown auditor to satisfy the "New York auditor" predicate) or allow signatures to be associated with a pseudonym of the signer (thus linking several signatures as originating from the same signer).

Applications

Attribute-based signatures have natural applications in many systems where users' capabilities depend on possibly complex combinations of attributes. ABS is a natural choice for simple authentication in such systems. One of our motivations for developing such schemes comes from the authentication requirements in an Attribute-Based Messaging (ABM) system. In addition to the "leaking secrets" application described above, in Section 7 we also identify applications in trust negotiation systems.

Overview of Our Results

We introduce the concept of Attribute-Based Signatures (ABS) as a powerful primitive with several applications and several efficient instantiations. Our main technical contributions in this work are the following:

• A formal security definition for ABS, that includes the guarantees of unforgeability (even in the presence of collusion) and privacy for the signer.

• A general framework for constructing ABS schemes. The ingredients that go into this framework are a "credential bundle scheme" and a non-interactive proof of credential ownership that can be bound to a message. The credential bundle must have the property that multiple users should not be able to collude and combine their credentials. The proof system must have some zero-knowledge-like guarantee so that the signature does not leak information about the signer's attributes. Based on this framework we present multiple constructions, which provide a spectrum of trade-offs between security and efficiency.

1. A credential bundle can be easily constructed from an arbitrary digital signature scheme, and the requisite proof system can be constructed from any non-interactive witness-indistinguishable (NIWI) argument scheme for proving membership in arbitrary NP-languages. Thus based on these standard primitives, our framework gives a conceptually simple ABS scheme. This yields the best *security guarantee* among the schemes we present, since the security can be based on any enhanced trapdoor permutation, a well-studied cryptographic primitive. However, in terms of efficiency and compactness this scheme is the worst.

2. Using certain digital signature schemes that use bilinear-pairings, we can use the NIWI scheme by Groth and Sahai [20] to carry out the proofs. We give two such instantiations, one using a signature scheme by Boneh and Boyen [7] and one using a signature scheme by Waters [33]. The two instantiations achieve different trade-offs in terms of sizes of the signature and the public-key. (Each of these instantiations have two further variants, depending on the computational assumption used to instantiate the Groth-Sahai NIWI argument system.) These two schemes are relatively practical, with the number of group

[2] Note that for attribute-based *encryption*, if collusion is allowed there are fairly easy solutions; but for ABS, even after allowing collusion (for instance by considering all users to have the same identity while generating keys), the residual primitive is essentially a mesh signature, which is already a non-trivial cryptographic problem.

elements in the signatures or public-key growing linearly in the security parameter. The security guarantee is reasonably strong, comparable to that of other recent attribute-based cryptographic schemes based on hardness assumptions related to bilinear-pairings.
3. By building a credential bundle scheme directly (instead of using a digital signature scheme) we can obtain further efficiency improvements. In particular, we obtain a scheme with a constant number of group elements in the signature and public-key (independent of the security parameter, and depending only on the size of the claim-predicate). This is a couple of orders of magnitude more efficient than the above schemes, but the security of the credential bundle scheme we use is based on non-standard assumptions.

- We present a practical ABS scheme suitable for high throughput systems. This construction deviates from our framework of credential bundles and proof of credential ownership. In this scheme we do employ a credential bundle scheme (same as the one in the last item above), but use a novel randomization technique to blind the actual attributes. This gives the best efficiency among our schemes. However, the security of this scheme is proven in the heuristic generic-group model (augmented to handle groups with bilinear pairings).
- One of the most striking features of our construction is that it is very easily amenable to natural multi-authority settings. We describe practical considerations related to such a deployment.
- We show how our techniques can be used to add *simulation-extractability* to the Groth-Sahai proof system, several orders of magnitude more efficiently than the only other comparable scheme, constucted by Groth in [19].

Which among the above schemes will suit an application will depend on the specific efficiency and security requirements in the system. In all these schemes, the privacy is unconditional, and it is only the unforgeability that depends on computational assumptions. Within a large enterprise setting (with pre-authenticated users) where the threat of forgery may be limited but the volume of signatures may be large, the final scheme may be the most suited. In more susceptible systems with a high security requirement, one of the schemes based on the Groth-Sahai proof systems maybe more suitable (at the expense of efficiency). The choice also depends on whether the application demands high-volume real-time performance (as in a messaging system) or involves only offline signing and verification (for instance when a secret is leaked to a newspaper).

All of our instantiations depend on expressing the attribute predicate as a monotone-span program, which is the state of the art for attribute-based cryptography [18, 3, 34]. We remark that unlike in many constructions of attribute-based encryption schemes, we achieve "full security" in all our constructions. That is, we do not weaken the definition in the manner of "selective-ID" security. Nor do we need to limit our construction to a small universe of attributes. Indeed, attributes can be arbitrary strings in all our instantiations: given a collision-resistant hash function, an *a priori* unbounded attribute universe can be used.

Further Related Work

Groups with bilinear pairings have been used recently to construct identity-based (e.g., [9]) and attribute-based encryption schemes [31, 18, 3]. Non-interactive zero-knowledge proofs (including identity-based proofs) have previously been used in the context of efficient constructions of signature primitives [1, 22, 11, 19].

A very useful tool in our instantiations is the Groth-Sahai non-interactive proof system [20]. In a precursor to that work, Groth [19] introduced a (much less efficient) zero-knowledge proof system, which enjoyed the stronger simulation-extractability property (which we improve upon in this work).

Khader [24, 23] proposes a notion called *attribute-based group signatures*. This primitive hides only the identity of the signer, but reveals which attributes the signer used to satisfy the predicate. It also allows a group manager to identify the signer of any signature (which is similar to the semantics of group signatures [15]); in contrast we require signer privacy to hold against everyone, including all authorities.

A preliminary (unpublished) version of this work [27] introduced the definition of ABS, and gave an efficient construction that is secure in the generic group model. Subsequently Li and Kim [25] gave an ABS scheme whose security relies on the computational Diffie-Hellman assumption in groups with bilinear maps. Their construction is limited to attribute-predicates which are solely conjunctions of attributes (hence privacy is required only for the identity of the signer and not for the attributes used in satisfying the predicate), is restricted to a "selective" unforgeability definition, and (unless random oracles are used) supports only a small universe of attributes. Guo and Zeng [21] also gave a construction of a signature scheme based on attribute policies, but their definition of security did not include any privacy for the signer.

Binding a non-interactive proof to a message, as we do, is also a feature of *identity-based* proofs [22], in which every proof is bound to some identity, and proofs under one identity cannot be used to forge proofs under a different identity (even of the same statement). Indeed, such ID-based proofs have been used to construct signature-like primitives; however the construction from [22] does not have all the properties we need.

Another related concept is the widely studied notion of anonymous credentials [14]. Anonymous credential systems are designed to allow users to prove that they possess a particular attribute (or more generally, a conjunction of attributes), whereas ABS considers more general predicates of attributes. However, anonymous credential systems are also incomparable to ABS schemes, because the goal of anonymous credential systems is to provide a user with anonymity against multiple colluding attribute authorities, typically having the side-effect of allowing users to pool their attributes. We leave it for future work to formulate a primitive reconciling the anonymity requirements of anonymous credentials with the guarantees of an ABS scheme.

A related primitive (but much simpler than ABS) is identity-based signatures (IBS) [32]. It is well-known that a simple scheme using traditional certificates realizes IBS, but dedicated schemes aimed at achieving better efficiency have been widely studied. We refer the reader to a comprehensive survey by Bellare et al. [2] for more details.

Supporting multiple attribute-authorities is crucial to many attribute-based systems. There has been much interest on this aspect for attribute-based *encryption* schemes Chase et al. [12, 13].

2 Preliminaries

2.1 Groups with Bilinear Pairings

Let $\mathbb{G}, \mathbb{H}, \mathbb{G}_T$ be cyclic (multiplicative) groups of order p, where p is a prime. Let g be a generator of $\mathbb{G}$, and h be a generator of $\mathbb{H}$. Then $e : \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$ is a *bilinear pairing* if $e(g, h)$ is a generator of $\mathbb{G}_T$, and $e(g^a, h^b) = e(g, h)^{ab}$ for all a, b. We review several standard cryptographic assumptions in such groups:

Definition 1 (q-SDH assumption [7])**.** *Let $\mathbb{G}$, $\mathbb{H}$, and $\mathbb{G}_T$ be as above. The* q-Strong Diffie-Hellman (q-SDH) assumption *holds in $(\mathbb{G}, \mathbb{H})$ if, given the elements $(g, g^x, g^{x^2}, \ldots, g^{x^q}, h, h^x) \in \mathbb{G}^{q+1} \times \mathbb{H}^2$, for random choice of $x \leftarrow \mathbb{Z}_p$ and random generators $g \in \mathbb{G}, h \in \mathbb{H}$, it is computationally infeasible to compute any pair of the form $\left(c, g^{\frac{1}{x+c}}\right) \in \mathbb{Z}_p \times \mathbb{G}$.*

Definition 2 (SXDH assumption [20])**.** *Let $\mathbb{G}$, $\mathbb{H}$, and $\mathbb{G}_T$ be as above. The* Symmetric External Diffie-Hellman (SXDH) assumption *holds in $(\mathbb{G}, \mathbb{H})$ if the standard Decisional Diffie-Hellman (DDH) assumption holds simultaneously in $\mathbb{G}$ and $\mathbb{H}$.*

Definition 3 (DLIN assumption [8])**.** *Let $\mathbb{G}$, $\mathbb{H}$, and $\mathbb{G}_T$ be as above, but with $\mathbb{G} = \mathbb{H}$. The* Decision-Linear (DLIN) assumption *holds in $\mathbb{G}$ if, given the elements $(g^x, g^y, g^{rx}, g^{sy}, g^t) \in \mathbb{G}^5$, for a random choice of $x, y, r, s \leftarrow \mathbb{Z}_p$, it is computationally infeasible to determine whether $t = r + s$ or t is random in $\mathbb{Z}_p$.*

2.2 Monotone Span Programs

Let $\Upsilon : \{0,1\}^n \to \{0,1\}$ be a monotone boolean function. A *monotone span program for* Υ over a field $\mathbb{F}$ is an $\ell \times t$ matrix $\mathbf{M}$ with entries in $\mathbb{F}$, along with a labeling function $a : [\ell] \to [n]$ that associates each row of $\mathbf{M}$ with an input variable of Υ, that, for every $(x_1, \ldots, x_n) \in \{0,1\}^n$, satisfies the following:

$$\Upsilon(x_1, \ldots, x_n) = 1 \iff \exists\, \vec{v} \in \mathbb{F}^{1\times\ell} : \vec{v}\mathbf{M} = [1, 0, 0, \ldots, 0] \\ \text{and } (\forall i : x_{a(i)} = 0 \Rightarrow v_i = 0)$$

In other words, $\Upsilon(x_1, \ldots, x_n) = 1$ if and only if the rows of $\mathbf{M}$ indexed by $\{i \mid x_{a(i)} = 1\}$ span the vector $[1, 0, 0, \ldots, 0]$.

We call ℓ the *length* and t the *width* of the span program, and $\ell + t$ the *size* of the span program. Every monotone boolean function can be represented by some monotone span program, and a large class do have compact monotone span programs. In particular, given a circuit expressed using *threshold gates*, with the i-th gate being an $\binom{\ell_i}{t_i}$ threshold gate, it is easy to recursively construct a monotone span program with length $\sum_i(\ell_i - 1) + 1$ and width $\sum_i(t_i - 1) + 1$.

2.3 Non-Interactive Proofs

We refer the reader to [20] for detailed definitions of non-interactive witness-indistinguishable (NIWI) proofs, but give a brief overview of the necessary definitions here. A NIWI scheme is comprised of the following main algorithms:

- NIWI.Setup: Outputs a reference string crs.
- NIWI.Prove: On input $(crs; \Phi; x)$, where Φ is a boolean formula and $\Phi(x) = 1$, outputs a proof π.
- NIWI.Verify: On input $(crs; \Phi; \pi)$, outputs a boolean.

The completeness requirement is that NIWI.Verify$(crs; \Phi;$ NIWI.Prove$(crs; \Phi; x)) = 1$, if $\Phi(x) = 1$ (i.e., x is a *witness* for Φ). The (perfect) witness indistinguishability requirement is that the distributions NIWI.Prove$(crs; \Phi; x_1)$ and NIWI.Prove$(crs; \Phi; x_2)$ are identical when x_1 and x_2 are witnesses for Φ. For the soundness/proof of knowledge requirement, we require the following additional algorithms:

- NIWI.SimSetup: Outputs a simulated reference string crs and trapdoor ψ.
- NIWI.Extract: On input $(crs, \psi; \Phi; \pi)$, outputs a witness x.

We require that the crs output by NIWI.SimSetup is indistinguishable to that of NIWI.Setup. Further, we require that for every $(crs, \psi) \leftarrow$ NIWI.SimSetup, if NIWI.Verify$(crs; \Phi; \pi) = 1$ then NIWI.Extract$(crs, \psi; \Phi; \pi)$ outputs a valid witness for Φ, with overwhelming probability.

3 Attribute-Based Signatures

We present the formal definitions of attribute-based signatures (ABS). An overview of how ABS can be used in an attribute-based system can be found in Appendix A

Let $\mathbb{A}$ be the universe of possible attributes. A *claim-predicate* over $\mathbb{A}$ is a monotone boolean function, whose inputs are associated with attributes of $\mathbb{A}$. We say that an attribute set $\mathcal{A} \subseteq \mathbb{A}$ *satisfies* a claim-predicate Υ if $\Upsilon(\mathcal{A}) = 1$ (where an input is set to be true if its corresponding attribute is present in $\mathcal{A}$).

Definition 4 (ABS)**.** *An* Attribute-Based Signature (ABS) scheme *is parameterized by a universe of possible attributes $\mathbb{A}$ and message space $\mathbb{M}$, and consists of the following four algorithms.*

- ABS.Setup*: Outputs public parameters PK and master key MK.*

- ABS.KeyGen*: On input* $(MK, \mathcal{A} \subseteq \mathbb{A})$*, outputs a signing key* $SK_{\mathcal{A}}$.[3]
- ABS.Sign*: On input* $(PK, SK_{\mathcal{A}}, m \in \mathbb{M}, \Upsilon)$*, where* $\Upsilon(\mathcal{A}) = 1$*, outputs a signature* σ.
- ABS.Ver*: On input* $(PK, m, \Upsilon, \sigma)$*, outputs a boolean value.*

Definition 5 (Correctness)**.** *We call an ABS scheme* correct *if for all* $(PK, MK) \leftarrow$ ABS.Setup*, all messages* m*, all attribute sets* $\mathcal{A}$*, all signing keys* $SK_{\mathcal{A}} \leftarrow$ ABS.KeyGen$(MK, \mathcal{A})$*, all claim-predicates* Υ *such that* $\Upsilon(\mathcal{A}) = 1$*, and all signatures* $\sigma \leftarrow$ ABS.Sign$\big(PK, SK_{\mathcal{A}}, m, \Upsilon\big)$*, we have* ABS.Ver$(PK, m, \Upsilon, \sigma) = 1$.

We present two formal definitions that together capture our desired notions of security. Slightly weaker security requirements may also be useful for most applications, but we use the stronger ones because our constructions satisfy them and because they are much easier to work with.

Definition 6 (Perfect Privacy)**.** *An ABS scheme is* perfectly private *if, for all* $(PK, MK) \leftarrow$ ABS.Setup*, all attribute sets* $\mathcal{A}_1, \mathcal{A}_2$*, all* $SK_1 \leftarrow$ ABS.KeyGen$(MK, \mathcal{A}_1)$*,* $SK_2 \leftarrow$ ABS.KeyGen$(MK, \mathcal{A}_2)$*, all messages* m*, and all claim-predicates* Υ *such that* $\Upsilon(\mathcal{A}_1) = \Upsilon(\mathcal{A}_2) = 1$*, the distributions* ABS.Sign(PK, SK_1, m, Υ) *and* ABS.Sign(PK, SK_2, m, Υ) *are equal.*

If the perfect privacy requirement holds, then a signature does not leak which set of attributes or signing key was used to generate it. This holds even if the adversary is unbounded and has access to the signer's private keys.

We slightly overload notation and write ABS.Sign(MK, m, Υ) (i.e., with the master key MK instead of PK and $SK_{\mathcal{A}}$) to denote the following procedure: first, run $SK_{\mathcal{A}} \leftarrow$ ABS.KeyGen$(MK, \mathcal{A})$ for any arbitrary $\mathcal{A}$ satisfying Υ; then output the result of ABS.Sign$(PK, SK_{\mathcal{A}}, m, \Upsilon)$. For convenience in the experiment below we use ABS.Sign$(MK, \cdot, \cdot)$ to generate signatures requested by the adversary. This is reasonable when the scheme satisfies perfect privacy, since any other way of letting the adversary obtain signatures will result in the same distribution.

Definition 7 (Unforgeability)**.** *An ABS scheme is* unforgeable *if the success probability of any polynomial-time adversary in the following experiment is negligible:*

1. *Run* $(PK, MK) \leftarrow$ ABS.Setup *and give* PK *to the adversary.*
2. *The adversary is given access to two oracles:* ABS.KeyGen$(MK, \cdot)$ *and* ABS.Sign$(MK, \cdot, \cdot)$.
3. *At the end the adversary outputs* $(m^*, \Upsilon^*, \sigma^*)$.

We say the adversary succeeds if (m^*, Υ^*) *was never queried to the* ABS.Sign *oracle, and* ABS.Ver$(PK, m^*, \Upsilon^*, \sigma^*) = 1$*, and* $\Upsilon^*(\mathcal{A}) = 0$ *for all* $\mathcal{A}$ *queried to the* ABS.KeyGen *oracle.*

Thus any signature which could not have been legitimately made by a *single* one of the adversary's signing keys is considered a forgery. Note that we do not consider it a forgery if the adversary can produce a *different* signature on (m, Υ) than the one he received from the signing oracle.

4 Constructing ABS Schemes

In this section we present our general framework for constructing an ABS scheme.

[3]For simplicity, we treat the signing key as a monolithic quantity. However, in our construction the signing key consists of separate components for each attribute in $\mathcal{A}$, and the ABS.Sign algorithm needs only as much of $SK_{\mathcal{A}}$ as is relevant to the claim-predicate.

4.1 Credential Bundles

We introduce a new generic primitive called *credential bundles*, which we use in our ABS constructions. Credential bundles model the intuitive requirements of publicly verifiable attributes that resist collusion.

Definition 8 (Credential bundle scheme)**.** *A* credential bundle scheme *is parameterized by a message space* $\mathbb{M}$*, and consists of the following three algorithms.*

- $\mathsf{CB.Setup}$*: Outputs a verification key* vk *and a secret key* sk*.*
- $\mathsf{CB.Gen}$*: On input* $(sk, \{m_1, \ldots, m_n\} \subseteq \mathbb{M})$*, outputs a tag* τ *and values* $\sigma_1, \ldots, \sigma_n$*.*
- $\mathsf{CB.Ver}$*: On input* $(vk, m, (\tau, \sigma))$*, outputs a boolean value.*

The scheme is correct *if, for all* $(\tau, \sigma_1, \ldots, \sigma_n) \leftarrow \mathsf{CB.Gen}(sk, m_1, \ldots, m_n)$*, we have* $\mathsf{CB.Ver}(vk, m_i, (\tau, \sigma_i)) = 1$ *for all* i*.*

Clearly by excluding some of the σ_i's from an existing bundle, one can generate a new bundle on a subset of attributes. Our main security definition requires that taking a subset of a *single* bundle is the only way to obtain a new bundle from existing bundles; in particular, attributes from several bundles cannot be combined.

Definition 9. *A credential bundle scheme is* secure *if the success probability of any polynomial-time adversary in the following experiment is negligible:*

1. *Run* $(vk, sk) \leftarrow \mathsf{CB.Setup}$*, and give* vk *to the adversary.*
2. *The adversary is given access to an oracle* $\mathsf{CB.Gen}(sk, \cdot)$*.*
3. *At the end the adversary outputs* $(\tau^*, (m_1^*, \sigma_1^*), \ldots, (m_n^*, \sigma_n^*))$*.*

We say the adversary succeeds *if* $\mathsf{CB.Ver}(vk, m_i^*, (\tau^*, \sigma_i^*)) = 1$ *for all* $i \leq n$*, and if no superset of* $\{m_1^*, \ldots, m_n^*\}$ *was ever queried to the* $\mathsf{CB.Gen}$ *oracle.*

From any unforgeable plain digital signature scheme we can easily construct a credential bundle scheme in which the bundle is a collection of signatures of messages "$\tau \| m_i$", where each m_i is the name of an attribute and τ is an identifier that is unique to each user. Conversely, when a credential bundle scheme is restricted to singleton sets of messages, its unforgeability definition is equivalent to normal digital signature unforgeability. Despite this equivalence under black-box reductions, the syntax of credential bundles more closely models our desired semantics for ABS.

4.2 A Framework for ABS

Our main construction is given in Figure 1. At a high level, to sign a message m with claim-predicate Υ, the signer proves that she possesses either a credential bundle containing either sufficient attributes to satisfy Υ, or a "pseudo-attribute" identified with the pair (m, Υ). Since $\mathsf{ABS.KeyGen}$ never gives out bundles involving these pseudo-attributes, the proof is convincing that the signer satisfied Υ. However, in the security reduction, the pseudo-attribute provides a mechanism to bind the NIWI proof to a message and give simulated signatures.

Theorem 1. *Given a NIWI argument of knowledge scheme and any secure credential bundle scheme (equivalently, any digital signature scheme), the construction in Figure 1 is a secure ABS scheme. Further, if the NIWI argument is perfectly hiding, the ABS scheme is perfectly private.*

Proof. Perfect privacy follows directly from the perfect witness hiding of the NIWI scheme, which our ABS scheme instantiates using the perfectly hiding setup.

Assuming that the NIWI scheme is sound, we show that any adversary A that violates ABS unforgeability can be converted into an adversary A^* that violates the security of the underlying credential bundle

Let $\mathbb{A}$ be the desired universe of ABS attributes. Let $\mathbb{A}'$ denote a space of *pseudo-attributes*, where $\mathbb{A} \cap \mathbb{A}' = \emptyset$. For every message m and claim-predicate Υ we associate a psuedo-attribute $a_{m,\Upsilon} \in \mathbb{A}'$. Let CB be a secure credential bundle scheme, with message space $\mathbb{A} \cup \mathbb{A}'$, and let NIWI be a perfect NIWI proof of knowledge scheme. Our ABS construction is as follows:

ABS.Setup: Run $crs \leftarrow$ NIWI.Setup and $(vk, sk) \leftarrow$ CB.Setup. Publish $PK = (crs, vk)$ and set $MK = sk$.

ABS.KeyGen$(MK, \mathcal{A})$**:** Ensure that $\mathcal{A}$ contains no pseudo-attributes. Then output the result of CB.Gen$(sk, \mathcal{A})$.

ABS.Sign$(PK, SK_{\mathcal{A}}, m, \Upsilon)$**:** Assume that $\Upsilon(\mathcal{A}) = 1$. Parse $SK_{\mathcal{A}}$ as $(\tau, \{\sigma_a \mid a \in \mathcal{A}\})$. Υ is a formula over formal variables $\mathbb{A}$. Define $\widetilde{\Upsilon} := \Upsilon \vee a_{m,\Upsilon}$, where $a_{m,\Upsilon} \in \mathbb{A}'$ is the pseudo-attribute associated with (m, Υ). Thus, we still have $\widetilde{\Upsilon}(\mathcal{A}) = 1$. Let $\{a_1, \ldots, a_n\}$ denote the attributes appearing in $\widetilde{\Upsilon}$ and let $\Phi(vk, m, \Upsilon)$ denote the following boolean expression:

$$\exists\, \tau, \sigma_1, \ldots, \sigma_n : \widetilde{\Upsilon}\Big(\big\{a_i \;\big|\; \text{CB.Ver}\big(vk, a_i, (\tau, \sigma_i)\big) = 1\big\}\Big) = 1 \tag{1}$$

For each i, set $\hat{\sigma}_i = \sigma_{a_i}$ from $SK_{\mathcal{A}}$ if it is present, and to any arbitrary value otherwise (since then its value does not matter). Compute $\pi \leftarrow$ NIWI.Prove$\big(crs; \Phi(vk, m, \Upsilon); (\tau, \hat{\sigma}_1, \ldots, \hat{\sigma}_n)\big)$. Output π as the ABS signature.

ABS.Ver(PK, m, Υ, π)**:** Output the result of NIWI.Verify$(crs; \Phi(vk, m, \Upsilon); \pi)$.

Figure 1: General framework for an ABS scheme.

scheme, with comparable advantage. Let A^* simulate a copy of A, and do the following in the bundle security experiment:

Receive from the experiment vk and run $(crs, \psi) \leftarrow$ NIWI.SimSetup. Give $PK = (crs, vk)$ to A. Whenever A makes a query $\mathcal{A} \subseteq \mathbb{A}$ to the ABS.KeyGen oracle, forward it to the CB.Gen oracle and return the result.

Whenever A makes a query (m, Υ) to the ABS.Sign oracle, request from the CB.Gen oracle a singleton bundle for the pseudo-attribute associated with (m, Υ). Use the result as a witness to generate a NIWI proof of $\Phi(vk, m, \Upsilon)$ to use as the simulated ABS signature.

Whenever A outputs a valid forgery (m^*, Υ^*, π^*), use NIWI.Extract with the trapdoor ψ to extract a witness for $\Phi(vk, m^*, \Upsilon^*)$. Extraction succeeds with overwhelming probability, thus we obtain a bundle that contains the pseudo-attribute associated with (m^*, Υ^*) or sufficient attributes to satisfy Υ^*. By the definition of ABS forgery, no ABS signature was ever simulated for (m^*, Υ^*), and no signing key was ever generated for an attribute set that satisfies Υ^*. Thus the extracted bundle constitutes a valid bundle forgery.

By applying the security of the NIWI scheme in a straight-forward series of hybrids (first replace legitimate signatures with simulated signatures, then replace NIWI.Setup with NIWI.SimSetup), we see that the advantage of A^* in the unforgeability game is comparable to that of A in the ABS forgery game. □

From this theorem and the constructions in [29, 16], we see that polynomial-time ABS schemes exist if enhanced trapdoor permutations exist. Of course, the bulky framework of zero-knowledge proofs of NP-statements makes such a construction impractical. In the next sections we describe specific instantiations that are quite practical.

4.3 Practical Instantiation 1

Our first practical instantiation uses Groth-Sahai proofs [20] as the NIWI component and Boneh-Boyen signatures [6] as the credential bundle component. One notable feature of this choice is that attributes in the scheme are simply Boneh-Boyen signatures on messages of the form "userid‖attr".

This instantiation requires cyclic groups of prime order equipped with bilinear pairings (Section 2.1). The Groth-Sahai system can prove satisfiability of *pairing-product equations* in such groups, and the main challenge in this instantiation is expressing the logic of the claim-predicate and the Boneh-Boyen signature verification in this limited vocabulary. We identify $\mathbb{Z}_p^*$ with the universe of attributes, where p is the size of the cyclic group used in the scheme.[4]

Boneh-Boyen signatures We briefly review the Boneh-Boyen digital signature scheme [7]. As before, we suppose there is a bilinear pairing $e : \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$, where $\mathbb{G}$ and $\mathbb{H}$ have prime order p, and where g is a generator of $\mathbb{G}$, and h is a generator of $\mathbb{H}$. The scheme, described below, is strongly unforgeable under the q-SDH assumpion (Definition 1).

DS.KeyGen**:** Choose random $b, c, d \leftarrow \mathbb{Z}_p$ and compute $B = g^b$, $C = g^c$, $D = g^d$. The verification key is $(B, C, D) \in \mathbb{G}^3$, and the signing key is $(b, c, d) \in (\mathbb{Z}_p)^3$.

DS.Sign$(sk, m \in \mathbb{Z}_p)$**:** Choose random $t \leftarrow \mathbb{Z}_p$; output $\sigma = \left(h^{1/(b+cm+dt)}, t\right) \in \mathbb{H} \times \mathbb{Z}_p$.

DS.Ver$(vk, m, \sigma = (S, t))$**:** Output 1 if $e(BC^m D^t, S) = e(g, h)$, and 0 otherwise.

Expressing the Non-Interactive Proof using Pairing Equations We use the notation introduced in Figure 1. We must show how the statement $\Phi(vk, m, \Upsilon)$ (equation 1) can be efficiently encoded in the Groth-Sahai system when the credential bundles use Boneh-Boyen signatures.

Groth-Sahai proofs work by first giving a commitment to the values of the witness, and then proving that the commited values satisfy given pairing equations. Suppose we commit to a group element Z (where the group $\mathbb{G}$ or $\mathbb{H}$ will be clear from context), then we will let $\langle Z \rangle$ denote the formal variable corresponding to that commitment. Thus, we express the statements to be proven as pairing equations whose formal variables we will write in the $\langle Z \rangle$ notation.

Suppose the modified predicate $\widetilde{\Upsilon}$ has a canonical monotone span program $\mathbf{M}$ of size $\ell \times t$, where the ith row corresponds to the $a(i)$-th attribute mentioned in $\widetilde{\Upsilon}$. To establish $\Phi(vk, m, \Upsilon)$, we prove the following equation, which implies it:

$$\begin{aligned} &\exists\, \tau, \sigma_1, \ldots, \sigma_n, v_1, \ldots, v_n : \vec{v}\mathbf{M} = [1, 0, \ldots, 0] \\ \wedge &\bigwedge_{i=1}^{\ell} \left[v_i \neq 0 \Rightarrow \mathsf{CB.Ver}(vk, a_{a(i)}, (\tau, \sigma_{a(i)})) = 1 \right] \end{aligned}$$

Then, in addition to $\tau, \{\sigma_i\}$, we will have the signer commit to the vector $\vec{v}$ which can be canonically computed from his satisfying assignment of $\widetilde{\Upsilon}$.

This new boolean expression is a conjunction of two kinds of clauses: The first has the form $\exists \vec{v} : \vec{v}\mathbf{M} = [1, \ldots, 0]$. To prove it, we commit to the values g^{v_i} and prove the following pairing equations (for each $j \in [t]$):

$$\prod_{i=1}^{\ell} e(\langle g^{v_i} \rangle, h^{\mathbf{M}_{i,j}}) = \begin{cases} e(g, h) & \text{if } j = 1 \\ e(g^0, h) & \text{otherwise} \end{cases}$$

[4] More precisely $\mathbb{A} \cup \mathbb{A}' \subseteq \mathbb{Z}_p^*$ where $\mathbb{A}'$ is the universe of pseudo-attributes. As is standard, the universe of (pseudo-)attributes can be extended to $\{0, 1\}^*$ by applying a collision-resistant hash with range $\mathbb{Z}_p^*$.

The other clauses have the form $\exists\, \tau, \sigma, v : \left[v \neq 0 \Rightarrow \mathsf{CB.Ver}(vk, m, (\tau, \sigma)) = 1\right]$. When we use Boneh-Boyen signatures as the instantiation of credential bundles, these clauses can be simplified to

$$\exists\, \tau, \sigma, v : \left[v \neq 0 \Rightarrow \mathsf{DS.Ver}(vk, \tau \| m, \sigma) = 1\right]$$

where $\mathsf{DS.Ver}$ is the Boneh-Boyen signature verification.

It is crucial that the proof is a proof *of knowledge*, so the simulator can extract the credential bundles. Thus we commit to τ and t *bitwise*, since they are elements of $\mathbb{Z}_p$ and could not otherwise be efficiently extracted in the Groth-Sahai scheme. In this way, the extractor can extract the bits and reconstruct the entire witness τ and t.[5] Let $(\tau, \sigma = (S, t), v)$ be a witness to the above expression. Express τ bitwise as $\tau = \sum_i \tau_i 2^i$. Then $\tau \| m$ may be identified with a number $m2^{|\tau|} + \sum_i \tau_i 2^i$. Similarly, interperet t bitwise as $t = \sum_i t_i 2^i$.

Using the same notation as before, we can prove satisfiability of the clause as follows. We commit to each t_i and τ_i in both groups, as $g^{t_i}, h^{t_i}, g^{\tau_i}, h^{\tau_i}$, and then first prove that each is indeed a single bit, using the following pairing equations for all i:

$$e(\langle g^{t_i} \rangle, h) = e(g, \langle h^{t_i} \rangle); \qquad e(\langle g^{\tau_i} \rangle, h) = e(g, \langle h^{\tau_i} \rangle);$$
$$e(\langle g^{t_i} \rangle, \langle h^{t_i} \rangle) = e(\langle g^{t_i} \rangle, h); \qquad e(\langle g^{\tau_i} \rangle, \langle h^{\tau_i} \rangle) = e(\langle g^{\tau_i} \rangle, h).$$

Next, observe that the pairing equation $e(BC^{\tau \| m} D^t, S^v) = e(g^v, h)$ is logically equivalent to the expression $v \neq 0 \Rightarrow \mathsf{DS.Ver}(vk, \tau \| m, (S, t)) = 1$, which we need to prove. However, the prover cannot directly compute $BC^{\tau \| m} D^t$ or S^v given the committed values. Thus the prover commits to some additional intermediate values $S^v \in \mathbb{H}$ and $C^\tau, D^t \in \mathbb{G}$, and proves the following equations:

$$e(\langle D^t \rangle, h) = \textstyle\prod_i e(D^{2^i}, \langle h^{t_i} \rangle); \quad e(\langle g^v \rangle, \langle S \rangle) = e(g, \langle S^v \rangle);$$
$$e(\langle C^\tau \rangle, h) = \textstyle\prod_i e(C^{2^i}, \langle h^{\tau_i} \rangle);$$
$$e(\langle g^v \rangle, h) = e(BC^{2^{|\tau|} m}, \langle S^v \rangle)\, e(\langle C^\tau \rangle, \langle S^v \rangle)\, e(\langle D^t \rangle, \langle S^v \rangle).$$

Note that since m and $|\tau|$ are public, all the coefficients in these equations can be publicly computed. This completes the description of how we encode the required logic into the Groth-Sahai proof system.

There are two instantiations of the Groth-Sahai proof system over prime order groups, based on the DLIN and SXDH assumptions, both of which are suitable for our purposes. Using these we obtain the following:

Theorem 2. *Under the q-SDH and either DLIN or SXDH assumptions, there is an ABS scheme supporting claim-predicates represented as monotone span programs, with signatures consisting of $O(ks)$ group elements,[6] where s is the size of the monotone span program.*

4.4 Practical Instantiation 2

We can also instantiate our framework using the same approach as above, but with the signature scheme of Waters [33]. Signatures in Waters' scheme do not include any elements of $\mathbb{Z}_p$. This fact allows us to avoid the inefficiency of committing to many components of the Boneh-Boyen signatures in a bitwise fashion. Furthermore, Waters signatures are secure under the much weaker BDH assumption, which is implied by the assumptions required for Groth-Sahai proofs. Thus this instantiation does not require the additional q-SDH assumption. However, as a tradeoff, the Waters instantiation requires larger public parameters: a linear (in the security parameter) number of group elements (Appendix B), not the constant number of group elements needed by the Boneh-Boyen instantiation.

[5] We remark that the proof need not be a proof of knowledge with respect to $\vec{v}$, so it was safe to use these values directly in $\mathbb{Z}_p$.

[6] A more detailed analysis of this instantiation's efficiency is given in Appendix B.

Waters signatures We briefly review the Waters digital signature scheme [33]. As before, we suppose there is a bilinear pairing $e : \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$, where $\mathbb{G}$ and $\mathbb{H}$ have prime order p, and where g is a random generator of $\mathbb{G}$, and h is a random generator of $\mathbb{H}$ (it is important that g and h are chosen independently at random, in the case where $\mathbb{G} = \mathbb{H}$).

$\mathsf{DS.KeyGen}$**:** Choose random $a, v_0, \ldots, v_n \leftarrow \mathbb{Z}_p$ and compute $A = h^a$, $V_i = g^{v_i}$. The verification key is $(A, V_0, \ldots, V_n)$, and the signing key is $g^a \in \mathbb{G}$.

$\mathsf{DS.Sign}(sk, m \in \mathbb{Z}_p)$**:** Choose random $r \leftarrow \mathbb{Z}_p$. Set

$$\sigma_1 = \left(V_0 \prod_{i=1}^{n} V_i^{m_i} \right)^r g^a; \qquad \sigma_2 = h^r$$

where m_i is the ith bit of m. Output $\sigma = (\sigma_1, \sigma_2) \in \mathbb{G} \times \mathbb{H}$.

$\mathsf{DS.Ver}(vk, m, \sigma = (\sigma_1, \sigma_2))$**:** Output 1 if

$$e\left(V_0 \prod_{i=1}^{n} V_i^{m_i},\ \sigma_2 \right) e(g, A) = e(\sigma_1, h)$$

and output 0 otherwise.

The Waters scheme is strongly unforgeable under the BDH assumpion, which is implied by either of the SXDH or DLIN assumptions (see [33]).

Expressing the Non-Interactive Proof using Pairing Equations We use the same approach as above to express the desired logic using pairing equations. The only significant difference is in how we encode clauses of the form

$$\exists\, \tau, \sigma, v : \left[v \neq 0 \Rightarrow \mathsf{DS.Ver}(vk, \tau \| m, \sigma) = 1 \right]$$

where $\mathsf{DS.Ver}$ is now the Waters signature verification.

Since the Waters scheme treats $\tau \| m$ bitwise, we must commit to τ bitwise, as before (m is an attribute name, and therefore public in all of our proof clauses). In this way, we ensure that the extractor can extract the bits and reconstruct the entire witness τ.

Let $(\tau, \sigma = (\sigma_1, \sigma_2), v)$ be a witness to the above expression. Express τ bitwise as $\tau = \tau_1 \cdots \tau_k$ and m as $m_1 \cdots m_k$. As before, we commit to τ_i in both groups, as g^{τ_i}, h^{τ_i}, and then first prove that each is indeed a single bit. This is done exactly as in the previous instantiation.

Next, observe that the pairing equation

$$e\left(V_0 \prod_{i=1}^{k} V_i^{\tau_i} \prod_{i=1}^{k} V_{k+i}^{m_i},\ \sigma_2 \right) e(g^v, A) = e(\sigma_1, h^v)$$

is logically equivalent to the desired expression $[v \neq 0 \Rightarrow \mathsf{DS.Ver}(vk, \tau \| m, (\sigma_1, \sigma_2)) = 1]$, provided that the prover sets $\sigma_2 = h^0$ when $v = 0$.

The prover cannot directly compute $\prod_i V_i^{\tau_i}$ given the committed values. Thus we let the prover commit

to this intermediate value, and prove consistency via the following equations:

$$e\left(\left\langle \prod_{i=1}^{k} V_i^{\tau_i} \right\rangle, h\right) = \prod_{i=1}^{k} e(V_i, \langle h^{\tau_i} \rangle);$$

$$e(\langle \sigma_1 \rangle, \langle h^v \rangle) = e\left(V_0 \prod_{i=1}^{k} V_{k+i}^{m_i}, \langle \sigma_2 \rangle\right) \cdot e\left(\left\langle \prod_{i=1}^{k} V_i^{\tau_k} \right\rangle, \langle \sigma_2 \rangle\right) e(\langle g^v \rangle, A).$$

Note that since m, A, B, V_i are public, all the coefficients in these equations can be publicly computed. Thus we have:

Theorem 3. *Under either the DLIN or SXDH assumptions, there is an ABS scheme supporting claim-predicates represented as monotone span programs, with signatures consisting of $O(k+s)$ group elements, where s is the size of the monotone span program.*

4.5 Practical Instantiation 3

In Figure 2 we present a credential bundle scheme based on weak Boneh-Boyen signatures [6] that avoids some of the inefficiencies of standard Boneh-Boyen and Waters signatures. Namely, the entire credential bundle consists of elements of the group $\mathbb{H}$, and no exponents in $\mathbb{Z}_p$. While this credential bundle no longer consists of standard digital signatures on messages of the form "userid||attr", the credential bundles no longer induce any dependence on the security parameter k in the ABS instantiation.

Assume there is a bilinear pairing $e : \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$, where $\mathbb{G}$ and $\mathbb{H}$ have prime order p, and where g is a generator of $\mathbb{G}$, and h is a generator of $\mathbb{H}$.

CB.Setup**:** Choose random $b, c \leftarrow \mathbb{Z}_p$ and compute $B = g^b$, $C = g^c$. The verification key is $(B, C) \in \mathbb{G}^2$, and the signing key is $(b, c) \in (\mathbb{Z}_p)^2$.

CB.Gen$(sk, m_1, \ldots, m_n \in \mathbb{Z}_p)$**:** Choose a random $x \leftarrow \mathbb{Z}_p$. Set $\tau = (h^x, h^{x/c}) \in \mathbb{H}^2$. For each i, compute $\sigma_i = h^{x/(b+m_i)} \in \mathbb{H}$. Output $(\tau, \sigma_1, \ldots, \sigma_n)$.

CB.Ver$(vk, m, (\tau, \sigma))$**:** Parse τ as (X, X_c). Output 1 if $X \neq h^0$ and $e(C, X_c) = e(g, X) = e(Bg^m, \sigma)$; otherwise output 0.

Figure 2: Simpler Credential Bundle Construction Used in Scheme 3

In Appendix C, we prove that this credential bundle construction is unforgeable in the generic group model.[7] Thus, we have:

Theorem 4. *Using Groth-Sahai proofs and the credential bundle scheme in Figure 2 yields an ABS scheme supporting claim-predicates represented as monotone span programs, with signatures consisting of $O(s)$ group elements, where s is the dimensions of the monotone span program.*

[7]However, we also note that this construction is similar in many respects to the weak Boneh-Boyen signature scheme presented in [6], which is statically unforgeable. Indeed, we can show that under a concrete assumption similar to q-SDH, our credential bundle scheme is similarly unforgeable when the adversary's signature queries in the experiment are made statically. A credential bundle scheme with such security implies an ABS scheme with similar static unforgeability (where both signing key and signature oracle requests are made statically).

4.6 Practical Instantiation 4

We now present an ABS scheme which is our most practical. Signatures in the scheme consist of *exactly* $s+2$ group elements, where s is the size of the claim-predicate's monotone span program. This scheme does not use the Groth-Sahai proof system; we use our own randomization techniques to blind the attributes that are used in signing. Our approach is motivated by the construction of mesh signatures [10], but incorporates the efficient credential bundles of the previous construction, as well as the concept of "pseudo-attributes" to bind a message to the signature. In Appendix D, we give a high-level motivation of the details of this scheme. Below we give a description of the construction:

This construction supports all claim-predicates whose monotone span programs have width at most t_{max}, where t_{max} is an arbitrary parameter. We treat $\mathbb{A} = \mathbb{Z}_p^*$ as the universe of attributes, where p is the size of the cyclic group used in the scheme.[8]

ABS.Setup: Choose suitable cyclic groups G and H of prime order p, equipped with a bilinear pairing $e : G \times H \to G_T$. Choose a collision-resistant hash function $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_p^*$. Choose random generators:

$$g, C \leftarrow G; \qquad h_0, \ldots h_{t_{\mathsf{max}}} \leftarrow H.$$

Choose random $a_0, a, b, c \leftarrow \mathbb{Z}_p^*$ and set:

$$A_0 = h_0^{a_0}; \qquad A_j = h_j^a \text{ and } B_j = h_j^b \quad (\forall j \in [t_{\mathsf{max}}]).$$

The master key is $MK = (a_0, a, b)$. The public key PK is a description of the groups G, H and their pairing function, as well as:

$$(\mathcal{H}, g, h_0, \ldots, h_{t_{\mathsf{max}}}, A_0, \ldots, A_{t_{\mathsf{max}}}, B_1, \ldots, B_{t_{\mathsf{max}}}, C)$$

ABS.KeyGen: On input MK as above and attribute set $\mathcal{A} \subseteq \mathbb{A}$, Choose random generator $K_{\mathsf{base}} \leftarrow G$. Set:

$$K_0 = K_{\mathsf{base}}^{1/a_0}; \qquad K_u = K_{\mathsf{base}}^{1/(a+bu)} \quad (\forall u \in \mathcal{A})$$

The signing key is then:

$$SK_{\mathcal{A}} = (K_{\mathsf{base}}, K_0, \{K_u \mid u \in \mathcal{A}\}).$$

ABS.Sign: On input $(PK, SK_{\mathcal{A}}, m, \Upsilon)$ such that $\Upsilon(\mathcal{A}) = 1$, first convert Υ to its corresponding monotone span program $\mathbf{M} \in (\mathbb{Z}_p)^{\ell \times t}$, with row labeling $u : [\ell] \to \mathbb{A}$. Also compute the vector $\vec{v}$ that corresponds to the satisfying assignment $\mathcal{A}$. Compute $\mu = \mathcal{H}(m \| \Upsilon)$.

Pick random $r_0 \leftarrow \mathbb{Z}_p^*$ and $r_1, \ldots r_\ell \leftarrow \mathbb{Z}_p$ and compute:

$$Y = K_{\mathsf{base}}^{r_0}; \qquad S_i = (K_{u(i)}^{v_i})^{r_0} \cdot (Cg^{\mu})^{r_i} \quad (\forall i \in [\ell]);$$

$$W = K_0^{r_0}; \qquad P_j = \prod_{i=1}^{\ell} (A_j B_j^{u(i)})^{\mathbf{M}_{ij} \cdot r_i} \quad (\forall j \in [t]).$$

We note that the signer may not have $K_{u(i)}$ for every attribute $u(i)$ mentioned in the claim-predicate. But when this is the case, $v_i = 0$, and so the value is not needed. The signature is:

$$\sigma = (Y, W, S_1, \ldots, S_\ell, P_1, \ldots, P_t)$$

[8] As always, the universe of attributes can be further extended to $\{0,1\}^*$ by applying a collision-resistant hash having range $\mathbb{Z}_p^*$. For simplicity of presentation, we do not include this modification.

ABS.Ver: On input $(PK, \sigma = (Y, W, S_1, \ldots, S_\ell, P_1, \ldots, P_t), m, \Upsilon)$, first convert Υ to its corresponding monotone span program $\mathbf{M} \in (\mathbb{Z}_p)^{\ell \times t}$, with row labeling $u : [\ell] \to \mathbb{A}$. Compute $\mu = \mathcal{H}(m \| \Upsilon)$. If $Y = 1$, then output reject. Otherwise check the following constraints:

$$e(W, A_0) \stackrel{?}{=} e(Y, h_0)$$

$$\prod_{i=1}^{\ell} e\Big(S_i, (A_j B_j^{u(i)})^{\mathbf{M}_{ij}}\Big) \stackrel{?}{=} \begin{cases} e(Y, h_1)\, e(Cg^{\mu}, P_1), & j = 1 \\ e(Cg^{\mu}, P_j), & j > 1, \end{cases}$$

for $j \in [t]$. Return accept if all the above checks succeed, and reject otherwise.

In Appendix D.1, we provide the detailed proof of security, which is carried out in the generic group model.

Theorem 5. *In the generic group model, there is an ABS scheme supporting claim-predicates represented as monotone span programs, with signatures consisting of $s + 2$ group elements, where s is the size of the monotone span program.*

5 Multiple Attribute-Authorities

Our first two intantiations of ABS (indeed, our general framework) can be easily extended for use in an environment with multiple attribute authorities. Except in a centralized enterprise setting, a single user would acquire her attributes from different authorities (e.g., different government agencies, different commercial services she has subscribed to, different social networks she is registered with and so on). These different attribute authorities may not trust each other, nor even be aware of each other. Indeed, some attribute authorities may be untrustworthy, and this should not affect the trustworthiness of attributes acquired from other authorities, or of ABS signatures involving trustworthy attributes.

Apart from these mutually distrusting attribute authorities, there should be some entity to set up the various public parameters of the signature scheme itself. We call this entity the **signature trustee**. A signature trustee does not have to trust any attribute authority. The attribute authorities use only the public keys from the signature trustee. As long as the signature trustee is trusted, then the ABS signatures are secure.

Our main change in multi-authority ABS syntax is to separate the key material into pieces originating from different authorities. For concreteness, we describe the proposed usage for our pairing-based instantiations. In this case, the signature trustee provides the Groth-Sahai reference string as well as a digital signature verification key, for use with the scheme's "pseudo-attributes." Attribute authorities each publish digital signature verification keys, and give out attributes as signatures on messages of the form "userid$\|$attr". We assume that all attribute authorities have a mechanism for agreeing on each user's userid (say, an email address).

Finally, the claim-predicate in the ABS signature must carry the identity of the attribute-authorities who *own* the various attributes (possibly as meta-data attached to the attribute description). Given this information, the statement Φ used in the Groth-Sahai proof can be modified to refer to the appropriate digital signature verification keys corresponding to each attribute, including the pseudo-attribute. In fact, different attribute authorities need not even agree on a digital signature scheme for their attributes (though all parties must agree on the cryptographic groups $\mathbb{G}$ and $\mathbb{H}$); one attribute authority might choose Boneh-Boyen signatures, while another might choose Waters signatures. If one attribute authority's signatures are compromised, then an ABS verifier should not give much importance to attributes from that authority. However, the ABS signatures themselves are still valid (in that they indeed attest to the given claim-predicate being satisfied) as long as the trustee is uncorrupted.

In Appendix E, we present formal security definitions for multi-authority ABS. The unforgeability definition is modified to account for the case where some of the attribute authorities are corrupt. The unforgeability requirement is then with respect to an uncorrupted signature trustee and attributes from uncorrupted attribute authorities.

6 Simulation-Extractable Identity-Based NIZK

Our technique of augmenting a NIWI proof with a digital signature scheme can be used to make any NIWI argument of knowledge into a *simulation-extractable*, identity-based NIZK argument of knowledge. Identity-based NIZK was defined in [22] as a NIZK proof with an associated identity. The soundness definition requires that seeing a proof under one identity does not help one to construct proofs under another identity (even proofs of the same statement).

Simulation-soundness [30], informally, requires that seeing a *simulated* proof does not help one to violate the soundness property of a proof system. In the case of proofs (or arguments) of knowledge, the corresponding notion is termed "simulation-extractability." We are interested in incorporating simulation-extractability into identity-based NIZK arguments of knowledge. We note that the identity-based NIZK from [22] is not simulation-sound.

Theorem 6. *Any statement provable in via Groth-Sahai proofs can be made an identity-based, simulation extractable, NIZK argument of knowledge, with overhead linear in the number of variables (independent of the number of constraints).*

The complete proof is given in Appendix F, and follows a similar approach as in our ABS instantiations: At a high level, we augment the common reference string of the Groth-Sahai system to include a Waters digital signature verification key.[9] Then to prove Φ under identity id, the prover proves using the NIWI that he either has a witness for Φ or a valid digital signature on message (id, Φ). The resulting scheme is only an argument (not proof) of knowledge, since unbounded parties may forge the digital signature.

This transformation results in simulation-extractable proofs which are several orders of magnitude more efficient than the only other comparable scheme, constucted by Groth in [19].

7 Applications

We identify several natural applications of ABS schemes; with more comprehensive details given in Appendix G.

Attribute-based messaging Attribute-Based Messaging, or ABM, (e.g., [4]) provides an example of a quintessential attribute-based system. In an ABM system, messages are addressed not by the identities of the recipients, but by a predicate on users' attributes which the recipients must satisfy. The users need not be aware of each other's identities or attributes. To provide *end-to-end* message privacy (against users whose attributes do not satisfy the sender's policy), one can use *ciphertext-policy attribute-based encryption*, as proposed by Bethencourt, Sahai and Waters [3]. However, there was no satisfactory way to achieve *authentication* (i.e., for the receiver to verify that the sender also satisfied a particular policy) in an ABM system until now. Existing cryptographic technology, including certificates and mesh signatures, would not provide an adequate level of anonymity for the senders while simultaneously preventing collusions.

[9] We choose Waters signatures because the cryptographic assumption required by Waters signatures is already implied by the assumption required for the Groth-Sahai proofs. A similar construction can be done with Boneh-Boyen signatures, but the additional q-SDH assumption is required.

In a typical ABM system, a certain degree of authorization is required to send messages to certain groups of users. That is, an attribute-based access control mechanism must decide whether to allow a messaging attempt from a sender, depending on both the attributes of the sender and the attribute-based address attached to the message. ABS can be used to authenticate a sender to the ABM system itself (as opposed to the scenario above, where the sender was authenticating to the message recipient). As the messaging system can publicly verify the ABS signature, this solution eliminates the need for the messaging system to query the attribute database to determine the sender's authorization. Indeed, the messaging system need not know the sender's identity at all.

Finally, because our construction is so readily suited for multi-authority settings, ABS is a natural choice for inter-domain ABM systems. However, there are many engineering and cryptographic challenges involved in other aspects of a truly inter-domain ABM system. For example, Chase's proposal [12] for multi-authority attribute-based encryption (originally for the schemes in [31, 18], but can be extended to the one in [3]) requires all the attribute-authorities to share secret keys with a central authority, thereby requiring the central authority to trust all the attribute authorities. In contrast, our ABS system requires no such trust between the signature trustee and attribute authorities. As such, ABS is much better suited to practical inter-domain attribute-based systems than its encryption counterparts.

Attribute-based authentication and trust-negotiation ABS can also be used as a more general fine-grained authentication mechanism. For instance, when a user requests access to a resource in a server, the server can give its access policy along with a random challenge string. The client can then generate a session key for (private-key) communication, generate an ABS signature of $(challenge, sessionkey)$ under the server's policy, and send these to the server on an encrypted channel. Thereafter, the client and server can communicate using the shared session key. This simple protocol is robust even against a man in the middle.

This technique can be extended to multiple rounds as a simple *trust negotiation* protocol, in which two parties progressively reveal more about their attributes over several rounds of interaction. Several recent works also consider cryptographic approaches to trust negotiation that give more privacy to users than is achieved when they simply take turns revealing their attributes [26, 17]. Instead of these techniques, ABS can provide a sophisticated way to reveal partial information about one's attributes that is natural for this setting. Being able to bind a message to such a proof about one's attributes, as ABS permits, also allows one to protect the trust negotiation from outside attack, using an approach as above. At each step of the negotiation, the active party can choose an "ephemeral key" for secure (private-key) communication and sign it using ABS. This approach prevents a man-in-the-middle attacks by an adversary who has enough attributes to intercept the first few steps of the negotiation.

Leaking secrets The classical application for which the notion of ring-signatures was developed by Rivest, Shamir and Tauman [28] is "leaking secrets," that we used as the motivating example in the opening of this paper. Ring signatures support only claim-predicates which are disjunctions. Mesh signatures are an extension of this concept which allow more sophisticated claim-predicates, but permit multiple parties to pool their attributes (atomic signatures). This is not necessarily the intended semantics in natural secret-leaking environment. ABS, on the other hand, provides the semantics that a *single* user (not a coalition) whose attributes satisfy the stated predicate attests to the secret.

Acknowledgement

We thank Amit Sahai for several useful discussions and collaboration during a part of this work.

References

[1] M. Bellare, D. Micciancio, and B. Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In E. Biham, editor, *EUROCRYPT*, volume 2656 of *Lecture Notes in Computer Science*, pages 614–629. Springer, 2003.

[2] M. Bellare, C. Namprempre, and G. Neven. Security proofs for identity-based identification and signature schemes. *Journal of Cryptology*, 22(1):1–61, January 2009. Preliminary version appeared in Eurocrypt 2004.

[3] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *IEEE Symposium on Security and Privacy*, pages 321–334, 2007.

[4] R. Bobba, O. Fatemieh, F. Khan, C. A. Gunter, and H. Khurana. Using attribute-based access control to enable attribute-based messaging. In *ACSAC*, pages 403–413. IEEE Computer Society, 2006.

[5] R. Bobba, O. Fatemieh, F. Khan, A. Khan, C. Gunter, H. Khurana, and M. Prabhakaran. Attribute based messaging: Access control and confidentiality. Manuscript (under submission), 2008.

[6] D. Boneh and X. Boyen. Secure identity based encryption without random oracles. In M. K. Franklin, editor, *CRYPTO*, volume 3152 of *Lecture Notes in Computer Science*, pages 443–459. Springer, 2004.

[7] D. Boneh and X. Boyen. Short signatures without random oracles. In C. Cachin and J. Camenisch, editors, *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 56–73. Springer, 2004.

[8] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In M. K. Franklin, editor, *CRYPTO*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer, 2004.

[9] D. Boneh and M. K. Franklin. Identity-based encryption from the Weil pairing. *SIAM J. Comput.*, 32(3):586–615, 2003.

[10] X. Boyen. Mesh signatures. In M. Naor, editor, *EUROCRYPT*, volume 4515 of *Lecture Notes in Computer Science*, pages 210–227. Springer, 2007.

[11] X. Boyen and B. Waters. Compact group signatures without random oracles. In S. Vaudenay, editor, *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 427–444. Springer, 2006.

[12] M. Chase. Multi-authority attribute based encryption. In S. P. Vadhan, editor, *TCC*, volume 4392 of *Lecture Notes in Computer Science*, pages 515–534. Springer, 2007.

[13] M. Chase and S. S. M. Chow. Improving privacy and security in multi-authority attribute-based encryption. In E. Al-Shaer, S. Jha, and A. D. Keromytis, editors, *ACM Conference on Computer and Communications Security*, pages 121–130. ACM, 2009.

[14] D. Chaum. Security without identification: Transaction systems to make big brother obsolete. *Commun. ACM*, 28(10):1030–1044, 1985.

[15] D. Chaum and E. van Heyst. Group signatures. In *EUROCRYPT*, pages 257–265, 1991.

[16] A. De Santis and G. Persiano. Zero-knowledge proofs of knowledge without interaction. In *33rd FOCS*, pages 427–436. IEEE Computer Society Press, 1992.

[17] K. B. Frikken, J. Li, and M. J. Atallah. Trust negotiation with hidden credentials, hidden policies, and policy cycles. In *NDSS*. The Internet Society, 2006.

[18] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In A. Juels, R. N. Wright, and S. D. C. di Vimercati, editors, *ACM Conference on Computer and Communications Security*, pages 89–98. ACM, 2006.

[19] J. Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In X. Lai and K. Chen, editors, *ASIACRYPT*, volume 4284 of *Lecture Notes in Computer Science*, pages 444–459. Springer, 2006.

[20] J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In N. P. Smart, editor, *EUROCRYPT*, volume 4965 of *Lecture Notes in Computer Science*, pages 415–432. Springer, 2008.

[21] S. Guo and Y. Zeng. Attribute-based signature scheme. In *International Conference on Information Security and Assurance*, pages 509–511. IEEE, 2008.

[22] J. Katz, R. Ostrovsky, and M. O. Rabin. Identity-based zero knowledge. In C. Blundo and S. Cimato, editors, *SCN*, volume 3352 of *Lecture Notes in Computer Science*, pages 180–192. Springer, 2004.

[23] D. Khader. Attribute based group signature with revocation. Cryptology ePrint Archive, Report 2007/241, 2007. `http://eprint.iacr.org/2007/241`.

[24] D. Khader. Attribute based group signatures. Cryptology ePrint Archive, Report 2007/159, 2007. `http://eprint.iacr.org/2007/159`.

[25] J. Li and K. Kim. Attribute-based ring signatures. Cryptology ePrint Archive, Report 2008/394, 2008. `http://eprint.iacr.org/2008/394`.

[26] N. Li, W. Du, and D. Boneh. Oblivious signature-based envelope. *Distributed Computing*, 17(4):293–302, 2005.

[27] H. K. Maji, M. Prabhakaran, and M. Rosulek. Attribute-based signatures: Achieving attribute-privacy and collusion-resistance. Cryptology ePrint Archive, 2008. `http://eprint.iacr.org/`.

[28] R. L. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In C. Boyd, editor, *ASIACRYPT*, volume 2248 of *Lecture Notes in Computer Science*, pages 552–565. Springer, 2001.

[29] J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proc.* 22*nd STOC*, pages 387–394. ACM, 1990.

[30] A. Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *FOCS*, pages 543–553, 1999.

[31] A. Sahai and B. Waters. Fuzzy identity-based encryption. In R. Cramer, editor, *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 457–473. Springer, 2005.

[32] A. Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO*, pages 47–53, 1984.

[33] B. Waters. Efficient identity-based encryption without random oracles. In R. Cramer, editor, *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 114–127. Springer, 2005.

[34] B. Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. Cryptology ePrint Archive, Report 2008/290, 2008. http://eprint.iacr.org/2008/290.

A Using ABS

Attribute-based signatures are just a cryptographic primitive fully defined by the above described algorithms and the security and correctness guarantees. To be useful in a system, ABS has to be used appropriately. Here we describe the typical usage scenario for ABS.

For the sake of expositional clarity, in this section we consider a setting with a single authority who sets up the system parameters and public keys, and also issues private keys for each user, for each of the user's attributes.[10]

Mapping Attributes Before describing the operation of the system, we need to relate the attributes as used in ABS with the attributes that occur in a real-life system. In a typical system one encounters attributes which have a *name* and optionally a *value*. For instance a user may possess an attribute named age, with a numerical value 25. On the other hand, some attributes may not have any value attached to them; for instance a user could possess an attribute named student. ABS, as described above, supports only the latter kind of attributes. Nevertheless, since the names supported by ABS are free-form strings, one could encode a $(\text{name}, \text{value})$ pair into a single string using an appropriate (standardized) encoding scheme.

But it is not enough to encode the attributes; one must also translate the predicates involving the $(\text{name}, \text{value})$ pair into predicates in terms of the encoded string. The above encoding is sufficient if the predicates involve only equality conditions. But for numerical attributes, other comparisons (like "$\geq$", "$\leq$") are also important. This can be taken care of by representing a single numerical attribute by a few value-less attributes, as has been already pointed out in [18, 3]. We remark that at the cost of increasing the number of value-less attributes used (thereby increasing private-key size of the user), one can reduce the size of the predicate representing a comparison condition, leading to faster operations (signing and verification, in our case).

Another issue regarding mapping real-life attributes to ABS attributes relates to attribute expiry and revocation issues. As discussed below, the collusion-resistance property of ABS provides suitable flexibility to support revocation. But the exact manner in which this flexibility is used is a design decision that trades off efficiency and security parameters.

Typical System with ABS In the single authority setting, the authority first runs the algorithm ABS.Setup to generate a global key pair for the scheme, and publishes the public key PK. This public-key will be picked up by all users who need to create or verify signatures in the system.

Later, each user visits the authority to obtain private keys corresponding to her attributes. Let $\mathcal{A} \subseteq \mathbb{A}$ be the set of attributes that the authority wants to give to this user. Then the authority runs ABS.KeyGen to generate a signing key $SK_{\mathcal{A}}$ corresponding to the set of attributes possessed by that user.

After this, parties can sign and verify messages without further interaction with the authority. As long as the authority is uncorrupted, the unforgeability guarantee holds. Further, even if the authority is corrupt, the perfect privacy guarantee holds for the signer.[11]

Changing Attributes In the scenario above the authority issued a single key $SK_{\mathcal{A}}$ for the set of attributes $\mathcal{A}$. Once issued this attribute set is never changed. This is usually not satisfactory. There are two possible solutions that ABS offers.

[10] We do not consider the technical issues of how the authority establishes the identity of a user before handing it any keys. Also, we consider it the authority's prerogative to determine which attributes should be given to each requesting user.

[11] Of course, if the authority wishes to reveal the user's attributes, it can; but irrespective of what the authority reveals, the signer has the guarantee that creating a signature reveals no *further* information about its attributes (beyond the fact that its attributes satisfied the claim-predicate).

When a user's attribute set changes, the authority can reissue an entire new set of signing keys, generated via ABS.KeyGen. This is akin to establishing a new user with new set of attributes. By the collusion-resistance the user cannot combine keys in the new set with keys in the old set (or any other set for that matter). The user can of course still create signatures using the old set of attributes, so the attributes should be designed to include expiry information if the system requires revoking the old attributes.

Alternately, if the user simply acquires new attributes, it is not necessary to issue a totally new key set. Though not apparent from the syntax presented above, in fact our ABS construction allows the authority to augment a key $SK_{\mathcal{A}}$ to $SK_{\mathcal{A}\cup\mathcal{A}'}$. (The syntax for this operation is made explicit in our definitions for multi-authority ABS, where keys are issued for one attribute at a time.) To allow for augmenting signing keys with new attributes, the authority could either maintain some state per user (to remember the randomness used to generate the key $SK_{\mathcal{A}}$), or provide a signed certificate of some public randomness that the user can keep and must bring back when requesting each new attribute, or more practically use a pseudorandom function such as AES to obtain this randomness as a function of the user's identity. In the latter case, the authority only needs to remember just one additional pseudorandom function seed (or AES key).

B Efficiency of Instantiations 1 and 2

Instantiation 1: Boneh-Boyen We simplify the following efficiency analysis by noting that $n \leq \ell$.

The proof requires $\ell(5+4k)$ variables: for each $i \in [\ell]$, the prover must commit to $g^{v_i}, S, S^{v_i}, D^t, C^\tau$ as well as all k of the bits of τ, t in both groups (if $\mathbb{G} = \mathbb{H}$, then only $\ell(5+2k)$ variables are needed).

There are $t + 4\ell k$ quadratic $\mathbb{Z}_p$ equations (these are equations where both the variables and the coefficients have known discrete logs): t to perform the matrix multiplication and $4\ell k$ to establish $\tau_j, t_i \in \{0,1\}$. The t matrix multiplication equations are of a special form (all variables are in $\mathbb{G}$) that some instantiations of Groth-Sahai can optimize. When $\mathbb{G} = \mathbb{H}$, $2\ell k$ of these equations are not needed.

There are 3ℓ multi-scalar product equations (these are equations where all variables and coefficients in one of the two groups have known discrete logs): for each $i \in [\ell]$, the equations involving S, D^t, and C^τ.

Finally, there are ℓ pairing-product equations (the most general form supported by Groth-Sahai): the equations that verify the main pairing equation for each $i \in [\ell]$.

Depending on the instantiation of Groth-Sahai used (based on either the SXDH or DLIN assumptions), the entire size of the ABS signature (measured in number of group elements) is given in the following table:

# of group elts	SXDH	DLIN ($\mathbb{G} = \mathbb{H}$)
$(5+4k)\ell$ vars	$10\ell + 8\ell k$	$15\ell + 6\ell k$
$t + 4\ell k$ quadratic	$2t + 16\ell k$	$2t + 12\ell k$
3ℓ multi-scalar	18ℓ	27ℓ
ℓ pairing-prod	8ℓ	9ℓ
signature size	$36\ell + 2t + 24\ell k$	$51\ell + 2t + 18\ell k$

We remark that the most efficient Groth-Sahai proof instantiation uses a composite subgroup decision problem, but working in a prime order subgroup of unknown size within a composite order group is incompatible with our approach. First, users must be able to compute $\vec{v}$ and matrix $\mathbf{M}$ given a description of Υ and a satisfying assignment. This may not always be possible if the linear algebra is in a field of unknown size. Second, Boneh-Boyen signatures are only known to be useful in groups of prime order.

Instantiation 2: Waters Again we simplify the following efficiency analysis by noting that $n \leq \ell$.

The proof requires $5\ell + 2k + 1$ variables: Each of the k bits of τ, in both groups (unless $\mathbb{G} = \mathbb{H}$), plus the product $\prod_i V_i^{tau_i}$, which is shared among several clauses. Then for each $i \in [\ell]$, the prover must commit to $g^{v_i}, h^{v_i}, \sigma_1, \sigma_2, A^{v_i}$.

There are $t + 2k + \ell$ quadratic $\mathbb{Z}_p$ equations (these are equations where both the variables and the coefficients have known discrete logs): t to perform the matrix multiplication, $2k$ to establish $\tau_j \in \{0, 1\}$, and ℓ to ensure consistency between g^{v_i} and h^{v_i}. The t matrix multiplication equations are of a special form (all variables are in $\mathbb{G}$) that some instantiations of Groth-Sahai can optimize. When $\mathbb{G} = \mathbb{H}$, the $2k$ equations involving τ_j are not needed.

There are $\ell+1$ multi-scalar product equations (these are equations where all variables and coefficients in one of the two groups have known discrete logs): one equation involving each A^{v_i}, and one overall involving $\prod_i V_i^{\tau_i}$.

Finally, there are ℓ pairing-product equations (the most general form supported by Groth-Sahai): the equations that verify the main pairing equation for each $i \in [\ell]$.

Depending on the instantiation of Groth-Sahai used (based on either the SXDH or DLIN assumptions), the entire size of the ABS signature (measured in number of group elements) is given in the following table:

# of group elts	SXDH	DLIN ($\mathbb{G} = \mathbb{H}$)
$5\ell + 2k + 1$ vars	$10\ell + 4k + 2$	$15\ell + 3k + 3$
$t + 2k + \ell$ quad	$2t + 8k + 4\ell$	$2t + 6k + 3\ell$
$\ell + 1$ multi-scalar	$6\ell + 6$	$9\ell + 9$
ℓ pairing-prod	8ℓ	9ℓ
signature size	$28\ell + 2t$ $+ 12k + 8$	$36\ell + 2t$ $+ 9k + 12$

Compared to the instantiation using Boneh-Boyen signatures, this instantiation is much more efficient. Each Boneh-Boyen signature involves bitwise operations on an element of $\mathbb{Z}_p$, but Waters signatures avoid this, thus eliminating the dominating $O(nk)$ factor in the total Groth-Sahai proof size. We note that this improvement comes at the cost of having $O(k)$ group elements in the verification key, instead of $O(1)$ group elements as in the Boneh-Boyen instantiation.

C Notes on Instantiation 3

Given our general framework (Theorem 1), to complete the security proof of the instantiation in Section 4.5 (Theorem 4.5), it is enough to prove the security of the Credential Bundle scheme it uses.

First we recall the Credential Bundle scheme. Below, let $\mathbb{G}$ and $\mathbb{H}$ be two groups with a bilinear pairing $e : \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$. Let g, h be generators of $\mathbb{G}$ and $\mathbb{H}$ respectively.

CB.Setup: Pick $b, c \leftarrow \mathbb{Z}_p^*$. Let $sk = (b, c)$. Let $vk = (g, h, B = h^b, C = h^c)$.

CB.Gen$(sk, m_1, \ldots, m_n)$**:** Pick a random $x \in \mathbb{Z}_p^*$. Compute $\tau = (g^x, g^{x/c})$ and $\sigma_i = g^{\frac{x}{b+m_i}}$, for all $i \in [n]$.

CB.Ver$(vk, m, (\tau, \sigma))$**:** Let the signature for $m_1, \ldots, m_n$ be $\tau = (X, Y)$ and $\sigma_1, \ldots, \sigma_n$. We accept, if the signature passes all the following checks:

- $X \neq 1$,
- $e(X, h) = e(Y, C)$, and
- $e(\sigma_i, Bh^{m_i}) = e(X, h)$, for all $i \in [n]$.

Lemma 1. *The credential bundle scheme above is secure in the generic group model.*

Proof. WLOG we can assume that $\mathbb{H} = \mathbb{G}$, because this can only help the adversary. Suppose the adversary queries q times and its i-th request for credentials was on the messages $\{m_{i,1}, \ldots, m_{i,n(i)}\}$, where $i \in [q]$. Let the i-th credential received by the adversary be:

$$\left(g^{x_i}, g^{x_i/c}\right), g^{\frac{x_i}{b+m_{i,1}}}, \ldots, g^{\frac{x_i}{b+m_{i,n(i)}}} \tag{2}$$

For any generic-group adversary, we consider a modified security experiment, carried out by a simulator as follows. For each group element seen or created by the adversary, this simulator keeps track of its discrete logarithm by means of multivariate rational functions in the following indeterminate formal variables:

$$\Sigma = \{b, c\} \cup \{x_k : k \in [q]\}$$

The simulation associates each group element with some rational function. For each *distinct* rational function in its collection, it chooses a random distinct representation f and gives it to the adversary as the encoding of that particular group element. The functions are associated with the group elements in the simulation as follows:

1. Public key components generated by CB.Setup:
   (a) 1, representing g
   (b) b, representing g^b
   (c) c, representing g^c
2. Signatures given out by CB.Gen:
   (a) x_i, representing g^{x_i}
   (b) x_i/c, representing $g^{x_i/c}$
   (c) $x_i/(b + m_{i,j})$, representing $g^{x_i/(b+m_{i,j})}$
3. Queries to the generic group oracle:
   (a) When the adversary asks for the group operation to be performed on α, β (specified by their encodings), where the group elements are associated with function F_α, F_β, associate with the result the function $F_\alpha + F_\beta$.
   (b) When the adversary asks for a group element α (specified by its encoding) to be raised to the power d, where α is associated with function F_α, associate with the result the function dF_α.

   The simulator returns a distinct handle for each group element that is associated with a distinct *function* over the formal variables.

We note that in the actual experiment, the values of the formal variables are chosen uniformly at random in $\mathbb{Z}_p^*$. Two distinct functions may in that case evaluate to the same value. The simulation is faithful to the standard interaction in a generic group, except in the event that two of the distinct functions evaluate to the same value on a random assignment to the formal variables. For any two distinct function of the form listed above, the probability of this happening is at most $O(\sum_{i\in[q]} n(i))/p$, since we can multiply both functions by the common denominator $c\prod_{i\in n(i):j\in[n(i)]}(b + m_{i,j})$ to obtain distinct multivariate *polynomials* with total degree at most $O(\sum_{i\in[q]} n(i))$. Since this probability is negligible, we ignore this case.

Now the adversary outputs a frogery $\tau^* = (g^{x^*}, g^{y^*})$ and $\sigma_1^* = g^{z_1^*}, \ldots, \sigma_n^* = g^{z_n^*}$ for $m_1^*, \ldots, m_n^*$.

If the adversary succeeds with non-negligible probability, then the two sides of the following set of constraints are functionally equivalent (otherwise, using the argument mentioned above, these constraints are satisfied with negligible probability):

$$x^* \neq 0$$
$$x^* = cy^*$$
$$z_j^*(b + m_j^*) = x^*, \text{ for } j \in [n]$$

Let $\mathtt{Lin}(\Gamma)$ be the set of all multilinear polynomials over the set of terms Γ with coefficients in $\mathbb{Z}_p$. Let $\mathtt{Hom}(\Gamma) \subset \mathtt{Lin}(\Gamma)$ be the subset of homogeneous polynomials (those with a zero constant coefficient).

Let us define

$$\Gamma = \{b, c\} \cup \{x_i/c | i \in [q]\} \cup \{x_i/(b + m_{i,j}) | i \in [q], j \in [n(i)]\} \tag{3}$$

It is easy to see that $x^*, y^*, z_j^* \in \mathtt{Lin}(\Gamma)$, for $j \in [n]$.

Since $y^*c = x^* \in \mathtt{Lin}(\Gamma)$, we get $y^* \in \mathtt{Lin}(\{x_1/c, \ldots, x_q/c\})$ and $x^* \in \mathtt{Hom}(\{c, x_1, \ldots, x_q\})$. Observe that $z_j^*(b + m_j^*) = x^*$, hence $[c]x^* = 0$. So, we can conclude that:

$$x^* \in \mathtt{Hom}(\{x_1, \ldots, x_q\})$$
$$y^* \in \mathtt{Hom}(\{x_1/c, \ldots, x_q/c\})$$
$$z_j^* \in \mathtt{Hom}(\{x_i/(b + m_{i,j}) | i \in [q], j \in n(i)\})$$

Since $x^* \neq 0$, there exists i_0 such that $[x_{i_0}]x^* \neq 0$. Hence coefficient of $\frac{x_{i_0}}{b+m_j^*}$ in z_j^* is not 0. Thus, $m_j^* \in \{m_{i_0,1}, \ldots, m_{i_0,n(i_0)}\}$. We can conclude that

$$\{m_j^* | j \in [n]\} \subseteq \bigcap_{i \in [q]} \{m_{i,j} | [x_i]x^* \neq 0, j \in [n(i)]\} \tag{4}$$

This shows that any adversary can forge credentials by only removing elements from previous queries, except with negligible probability. □

Non-interactive proof via pairing equations. To incorporate the credential bundle scheme from Figure 2 into our general framework, we use the same Groth-Sahai techniques as in the previous section. Thus it suffices to show how statements of the form $\exists\ \tau, \sigma, v : \left[v \neq 0 \Rightarrow \mathsf{CB.Ver}(vk, m, (\tau, \sigma)) = 1\right]$ can be proven about this attribute bundle scheme using Groth-Sahai proofs.

First, we note that any bundle $((X, X_c), \{\sigma_a \mid a \in \mathcal{A}\})$ can in fact be rerandomized as $((X^r, (X_c)^r), \{(\sigma_a)^r \mid a \in \mathcal{A}\})$, where r is chosen randomly in $\mathbb{Z}_p^*$. Thus without loss of generality, we assume that each signature is generated with a freshly random bundle.

To prove the statement in question, we give out X and X_c in the clear (since an adversary in our experiment can always freely obtain a random X, X_c subject to $(X_c)^c = X$). The verifier can check $X \neq 1$ and check $e(C, X_c) = e(g, X)$. Then we also commit to σ^v and g^v and prove the following pairing equations:

$$e(Bg^m, \langle \sigma^v \rangle) = e(\langle g^v \rangle, X); \quad e(\langle g^v \rangle, \langle \sigma \rangle) = e(g, \langle \sigma^v \rangle)$$

Efficiency. The proof sends 2 group elements in the clear, and then uses 3ℓ variables: for each $i \in [\ell]$, the prover must commit to $g^{v_i}, \sigma_i, \sigma_i^{v_i}$.

There are t quadratic $\mathbb{Z}_p$ equations used to carry out the matrix multiplication. Again, these are of a special form (all variables are in $\mathbb{G}$) that some instantiations of Groth-Sahai can optimize.

There are ℓ multi-scalar product equations: those equations comparing σ to σ^v.

Finally, there are ℓ pairing-product equations: those that involve Bg^m.

Depending on the instantiation of Groth-Sahai used (based on either the SXDH or DLIN assumptions), the entire size of the ABS signature (measured in number of group elements) is given in the following table:

# of group elements	SXDH	DLIN
2 sent in the clear	2	2
3ℓ variables	6ℓ	9ℓ
t quadratic eqns	$2t$	$2t$
ℓ multi-scalar eqns	6ℓ	9ℓ
ℓ pairing-product eqns	8ℓ	9ℓ
total signature size	$20\ell + 2t + 2$	$27\ell + 2t + 2$

D Notes on Instantiation 4

Our construction from Section 4.6 is perhaps the best choice in a typical attribute-based system, especially if the system already involves attribute-based encryption schemes whose security is proven in the generic-group model. We make a few notes about the efficiency and extensions of the scheme.

Delegation This scheme supports delegation of attributes in a natural way. Suppose a party has a signing key for $\mathcal{A}$, say, $(K_{\mathsf{base}}, K_0, \{K_u \mid u \in \mathcal{A}\})$. Then for any $\mathcal{A}' \subseteq \mathcal{A}$, when choosing random $r \leftarrow \mathbb{Z}_p^*$, the quantity $((K_{\mathsf{base}})^r, (K_0)^r, \{(K_u)^r \mid u \in \mathcal{A}'\})$ is a valid, correctly distributed signing key for attribute set $\mathcal{A}'$.

Probabilistic Verification Using a standard technique, signatures in this scheme can be verified probabilistically with only $\ell + 4$ pairings instead of $\ell t + 2$, at the cost of additional exponentiations and a very small probability of false positives.

To probabilistically verify a signature, proceed as in the normal verification algorithm, but replace the final t checks with the following random one: Choose random $r_1, \ldots r_t \leftarrow \mathbb{Z}_p^*$, and check the single constraint:

$$\prod_{i=1}^{\ell} e\left(S_i, \prod_{j=1}^{t} (A_j B_j^{u(i)})^{\mathbf{M}_{ij} \cdot r_j}\right) \stackrel{?}{=} e(Y, h_1^{r_1})\, e\left(Cg^{\mu}, \prod_{j=1}^{t} P_j^{r_j}\right)$$

This is essentially a random linear combination of the t original constraints. Legitimate signatures pass such a check with probability 1, while invalid signatures pass with probability at most $1/p$.

Efficiency The total public key data consists of $3(t_{\mathsf{max}} + 1)$ group elements, which we emphasize is independent of the number of possible attributes in the system. Signatures have linear size, consisting of $\ell + t + 2$ group elements, where ℓ and t are the dimensions of the claim-predicate's monotone span program. Signing can be done using a maximum of $2w + \ell(1 + 2t) + 3$ exponentiations in G and H, where w is the minimum number of attributes needed for the signer to satisfy Υ.

D.1 Security Proof

In this section we prove Theorem 5, by showing that Scheme 4 is a secure ABS scheme in the generic group model. Note that in this scheme the signature indeed has $s + 2$ elements (where, by definition, the size of the monotone span program $s = \ell + t$). We break the security proof into the following two lemmas.

Lemma 2. *Scheme 4 is a correct (Definition 5) and perfectly private (Definition 6) ABS scheme.*

Proof. Correctness can be seen by straight-forward substitutions. To prove perfect privacy it suffices to show that for any claim-predicate Υ and any attribute set $\mathcal{A}$ that satisfies Υ, the output of $\mathsf{ABS.Sign}(PK, SK_{\mathcal{A}}, m, \Upsilon)$ is uniformly distributed among signatures σ, subject to the constraint that $\mathsf{ABS.Ver}(PK, m, \Upsilon, \sigma) = 1$. For $\sigma = (Y, W, S_1, \ldots, S_n, P_1, \ldots, P_t)$ it is easy to see that for any setting of $Y \neq 1$ and $S_1, \ldots, S_n$, there is a unique value of $W, P_1, \ldots, P_t$ for which the signature successfully verifies. We conclude by observing that Y and $S_1, \ldots, S_n$ output by $\mathsf{ABS.Sign}$ are distributed uniformly in their respective domains and that the signature output by $\mathsf{ABS.Sign}$ successfully verifies. □

Lemma 3. *Scheme 4 is unforgeable (Definition 7) in the generic group model.*

Proof. We first observe that the distribution $\mathsf{AltSign}(MK, m, \Upsilon)$ can be sampled in the following way:
Let $\mathbf{M} \in (\mathbb{Z}_p)^{l \times t}$ be the monotone span program for Υ, with row labeling $u : [l] \to \mathbb{A}$; let $\mu = \mathcal{H}(m \| \Upsilon)$.

1. Pick random $s_1, \ldots, s_l \leftarrow \mathbb{Z}_p$ and $y \leftarrow \mathbb{Z}_p^*$
2. For all $j \in [t]$, compute

$$p_j = \frac{1}{(c+\mu)} \left[\sum_{i=1}^{l} s_i (a + u(i)b) \mathbf{M}_{i,j} - y z_j \right],$$

   where $\vec{z} = [1, 0, \ldots, 0]$.
3. Output $\sigma = (g^y, g^{y/a_0}, g^{s_1}, \ldots, g^{s_l}, h_1^{p_1}, \ldots, h_t^{p_t})$

It is straight-forward to check that this distribution matches $\mathsf{AltSign}(MK, m, \Upsilon)$. WLOG, we assume that in the security experiment, responses to signature queries are generated in this way.

We now proceed with the proof of unforgeability, following the standard approach for generic groups.

Using arguments similar to the ones provided in Appendix C we assume that the groups G and H coincide. Suppose, the adversary outputs a purported forgery signature σ^* on a policy Υ^* and message m^* such that $(m^*, \Upsilon^*) \neq (m^{(q)}, \Upsilon^{(q)})$ for all q. Let $\mathbf{M}^* \in \mathbb{Z}_p^{l^* \times t^*}$ be the corresponding monotone span program with row labeling $u^*(\cdot)$. Let $\mu^* = \mathcal{H}(m^* \| \Upsilon^*)$, and suppose the signature has the form $\sigma^* = (g^{y^*}, g^{w^*}, g^{s_1^*}, \ldots, g^{s_{l^*}^*}, g^{p_1^*}, \ldots, g^{p_{t^*}^*})$.

To be a forgery, we need $y^* \neq 0$, and $w^* = y^*/a_0$, and

$$\sum_{i=1}^{l^*} s_i^* \mathbf{M}_{i,j}^* (a + u^*(i)b) \Delta_j = y^* z_j \Delta_j + (c + \mu^*) p_j^* \ \ (\forall j \in [t^*])$$

WLOG we can assume that these constraints are functionally satisfied. The rest of our proof proceeds by assuming these constraints are functionally equivalent, and eventually obtaining a contradiction: that there exists a $k_0 \in [n]$ such that $\Upsilon(\mathcal{A}_{k_0}) = 1$. In other words, the adversary could have generated a signature legitimately with the signing key for $\mathcal{A}_{k_0}$, and thus the output is not a forgery.

We know that $y^*, w^*, s_1^*, \ldots, s_{l^*}^*, p_1^*, \ldots, p_{t^*}^* \in \mathtt{Lin}(\Gamma)$, where

$$\begin{aligned}\Gamma = &\{1, a_0, \Delta_0, c\} \cup \{\Delta_j, a\Delta_j, b\Delta_j \mid j \in [t_{\mathsf{max}}]\} \\ &\cup \{x_k, x_k/a_0, x_k/(a+bu) \mid k \in [n], u \in \mathcal{A}_k\} \\ &\cup \{s_i^{(q)}, y^{(q)}, w^{(q)}, p_j^{(q)} \mid q \in [\nu], i \in [l^{(q)}], j \in [t^{(q)}]\}\end{aligned}$$

The rest of our analysis proceeds by comparing terms in these constraints. We can show that the multilinear functions given by the adversary's forgery cannot contain terms of certain kinds. Since $y^* = w^* a_0$, we get that:

$$y^* \in \texttt{Hom}\left(\{\Delta_0 a_0\} \cup \{x_k : k \in [n]\} \cup \{y^{(q)} : q \in [\nu]\}\right)$$

It is easy to see that $\Delta_j | (c + \mu^*) p_j^*$ and hence $\Delta_j | p_j^*$. So,

$$p_j^* \in \texttt{Hom}\left(\{\Delta_j, \Delta_j a, \Delta_j b\} \cup \{p_j^{(q)} : q \in [\nu]\}\right)$$

Consider j_0 such that $z_{j_0} \neq 0$. Then suppose y^* has a $\Delta_0 a_0$ term. Then there is a $\Delta_0 a_0 \Delta_{j_0}$ monomial in $y^* z_j \Delta_{j_0}$. This monomial cannot occur in $(c + \mu^*) p_{j_0}^*$, nor can it occur in $\sum_i^{l^*} s_i^* \mathbf{M}_{i,j_0}(a + u^*(i)b)\Delta_{j_0}$, since all monomials from the sum have a factor of a or b. Hence,

$$y^* \in \texttt{Hom}\left(\{x_k : k \in [n]\} \cup \{y^{(q)} : q \in [\nu]\}\right)$$

Suppose p_j^* has Δ_j term. Then the right hand side contributes monomials Δ_j and $b\Delta_j$. Because y^* has no constant term, $y^* z_j \Delta_j$ can not contribute a Δ_j monomial. And similar to above, $\sum_i^{l^*} s_i^* \mathbf{M}_{i,j}(a + u^*(i)b)\Delta_j$ can not contribute a monomial with Δ_j alone, hence

$$p_j^* \in \texttt{Hom}\left(\{\Delta_j a, \Delta_j b\} \cup \{p_j^{(q)} : q \in [\nu]\}\right)$$

Suppose p_j^* has a $p_j^{(q)}$ term. Then on the right hand side we will have a contribution of $(c + \mu^*) p_j^*$, producing a term with a factor of $(c + \mu^*)/(c + \mu^{(q)})$. Since $\mu^* \neq \mu^{(q)}$ for any q, this is a proper rational. No setting of y^* or $\{s_i^*\}_{i \in [l^*]}$ can yield terms in the final equation with a factor of $(c + \mu^*)/(c + \mu^{(q)})$. Hence:

$$p_j^* \in \texttt{Hom}\left(\{\Delta_j a, \Delta_j b\}\right)$$

Consider j_0 such that $z_{j_0} \neq 0$. Now, y^* can not have a $y^{(q)}$ term, because neither $(c + \mu^*) p_{j_0}^*$ nor $\sum_i^{l^*} s_i^* \mathbf{M}_{i,j_0}(a + u^*(i)b)\Delta_{j_0}$ can contribute a monomial of this form. Hence:

$$y^* \in \texttt{Hom}\left(\{x_k : k \in [n]\}\right)$$

Finally we conclude that:

$$p_j^* \in \texttt{Hom}\left(\{\Delta_j a, \Delta_j b\}\right) \quad \text{and} \quad y^* \in \texttt{Hom}(\{x_k : k \in [n]\})$$

Observe that any term which appears in y^* must also be contributed from the left hand side, to make the expression equal. So, we can split s_i^* into two parts; one whose terms involve x_k variables, and one which does not. Let

$$s_i^* = t_i^*\left(X_i\right) + \delta^*(\Gamma \setminus X_i)$$

where $X_i = \left\{ \frac{x_k}{(a + u^*(i)b)} : u(i) \in \mathcal{A}_k, k \in [n] \right\}$. Observe that $t_i^* \in \texttt{Hom}(X_i)$, and satisfies the following equation for all $j \in [t]$:

$$\sum_{i=1}^{l^*} t_i^* \mathbf{M}_{i,j}^*(a + u^*(i)b) = y^* z_j$$

Consider any x_{k_0} such that it has a non-zero coefficient in y^*. Construct v_i^*, for $i \in [l]$, by defining

$$v_i^* = \frac{1}{[x_{k_0}] y^*} \left[\frac{x_{k_0}}{a + u^*(i)b} \right] t_i^*$$

where the $[x]\pi$ notation denotes the coefficient of the term x in π. We see that v^* is a vector of constant coefficients which satisfies the equation $v^* M^* = [z_1 \ldots z_t] = [1, 0 \ldots, 0]$. Further, in every position where $v_i^* \neq 0$, the set $\mathcal{A}_{k_0}$ surely contained the attribute $u^*(i)$. By the properties of the monotone span program, it must be the case that $\Upsilon^*(\mathcal{A}_{k_0}) = 1$, and thus the signature is not a forgery. $\square$

E Multiple Attribute Authorities

When an attribute-based system is in an enterprise setting (say, an attribute-based messaging system for communications within a corporation), there would be a single authority issuing attributes to the users and setting up the ABS scheme. However, for many practically interesting settings, it is important to allow users to obtain attributes from different attribute authorities who may not trust each other, or may not even be aware of each other. Indeed, some of the attribute authorities may be corrupt, and this should not affect the attributes issued by other authorities.

As a simple illustrative example, suppose Alice wishes to anonymously publish an anecdote on user experience in online social networks. To give credibility to her story she decides to use the following claim to endorse her message:

(Facebook user for 2 years AND Has 100 Facebook friends)
OR (Has 100 Orkut friends AND Participated in 100 Orkut discussion forums)
OR ((Princeton professor OR Yale professor) AND Expert on online social networks).

Alice wants to endorse her anecdote using this claim, without having to reveal how she satisfies the claim. These attributes are owned by different attribute authorities like Facebook, Orkut, Princeton University, Yale University and the American Sociological Association, who may not trust each other, or may not even be aware of each other. Nor might all these authorities trust a common central agency. To satisfy the claim Alice may need to use attributes she acquired from different authorities, say Yale and the ASA. To make matters more challenging, Alice may have never interacted with Facebook's attribute-authority, and yet she wishes to use an arbitrary attribute string from Facebook as part of her claim.

In the following we extend the notion of ABS to such a multi-authority setting. Then in Appendix E.2 we will illustrate how Alice can use multi-authority ABS to solve her problem.

In a multi-authority ABS scheme, apart from (mutually distrusting) attribute authorities, there needs to be an entity to set up the various public parameters of the signature system. We call this entity the **signature trustee**. However, *we shall require that the signature trustee does not have to trust any attribute authority.* In particular, the attribute authorities use only the public keys from the signature trustee.

Finally, we shall also allow there to be multiple signature trustees. In this case, the attribute authorities would issue attribute keys to a user for all the signature trustees she wishes to work with. Here, an attribute authority or a signer need not trust the signature trustee.

Below we give a summary of the modifications in the syntax and security definitions of the ABS primitive for the multi-authority setting, followed by the formal definition.

Modified syntax. Our main changes in syntax involve separating the key material into pieces originating from different authorities. Further, the syntax must now include new safety checks on the key material, since (a) the authorities depend on the user to provide key material from the trustees, and (b) the users no longer consider all authorities as trusted entities.

The claim-predicates in the signatures are now required to carry the identity of the attribute-authorities who *own* the various attributes (possibly as meta-data attached to the attribute description). Note that if for any attribute appearing in the claim-predicate the verifier uses a different attribute-authority than what the signer used, the verification will simply fail. So it is in the interest of the signer to point to the correct attribute-authorities.

Definition 10 (Multi-Authority ABS)**.** *A* multi-authority ABS scheme *consists of the following algorithms/protocols:*

ABS.TSetup**:** *The signature trustee runs the algorithm* ABS.TSetup *which produces a* trustee public key PK *and* trustee secret key TSK*. The trustee publishes* PK *and stores* TSK*.*

ABS.TRegister: *When a user with user id* uid *registers with the signature trustee, the trustee runs* ABS.TRegister(TSK, uid) *which outputs a public* user-token τ. *The trustee gives* τ *to the user.*

ABS.ASetup: *An attribute authority who wishes to issue attributes runs* ABS.ASetup(PK) *which outputs an* attribute-authority public key APK *and an* attribute-authority secret key ASK. *The attribute authority publishes* APK *and stores* ASK.

ABS.KeyGen: *When an attribute authority needs to issue an attribute* $u \in \mathbb{A}$ *to a user* uid*, first it obtains (from the user) her user-token* τ*, and runs a token verification algorithm* ABS.TokenVerify(PK, uid, τ). *If the token is verified, then it runs* ABS.KeyGen(ASK, τ, u) *which outputs an* attribute key K_u. *The attribute authority gives* K_u *to the user.*

The user checks this key using ABS.KeyCheck(PK, APK, τ, K_u) *and accepts this attribute key only if it passes the check.*

ABS.Sign: *A user signs a message* m *with a claim-predicate* Υ*, only if there is a set of attributes* $\mathcal{A}$ *such that* $\Upsilon(\mathcal{A}) = 1$*, the user has obtained a set of keys* $\{K_u \mid u \in \mathcal{A}\}$ *from the attribute authorities, and they have all passed* ABS.KeyCheck. *Then the signature* σ *can be generated using*

$$\mathsf{ABS.Sign}\left(PK, \{APK_{\mathsf{auth}(u)} \mid u \in \mathbb{A}_\Upsilon\}, \tau, \{K_u \mid u \in \mathcal{A}\}, m, \Upsilon\right).$$

Here $\mathsf{auth}(u)$ *stands for the authority who owns the attribute* u *(as described in* u*), and* $\mathbb{A}_\Upsilon$ *is the set of attributes appearing in* Υ. (m, Υ, σ) *can be given out for verification.*

ABS.Ver: *To verify a signature* σ *on a message* m *with a claim-predicate* Υ*, a user runs*

$$\mathsf{ABS.Ver}\left(PK, \{APK_{\mathsf{auth}(u)} \mid u \in \mathbb{A}_\Upsilon\}, m, \Upsilon, \sigma\right)$$

which outputs a boolean value, accept *or* reject.

Security Definitions The security definitions are now a little more elaborate to accommodate for the different cases corresponding to different entities (signers, verifiers, attribute-authorities and signature-trustees) being corrupt.

The privacy requirement is formulated as a perfect information-theoretic property: for every PK, m, and Υ, the output distribution of ABS.Sign$(PK, \{APK_{\mathsf{auth}(u)} \mid u \in \mathbb{A}_\Upsilon\}, \cdot, \cdot, m, \Upsilon)$ is the same no matter which τ, and attribute signing keys $\{K_u\}$ are used, as long as the keys $\{K_u\}$ have all passed ABS.KeyCheck. In other words, there is a (computationally infeasible) procedure AltSign such that AltSign$(PK, m, \Upsilon, \{APK_{\mathsf{auth}(u)} \mid u \in \mathbb{A}_\Upsilon\}))$ is distributed exactly as a valid signature on m with claim-predicate Υ.

The unforgeability definition is modified to account for the case where some of the attribute authorities, and some signature trustees are corrupt. The unforgeability requirement is with respect to an uncorrupted signature trustee (whose setup is carried out by the experimenter in the security experiment).

Definition 11. *A multi-authority ABS scheme is* unforgeable *if the success probability of every polynomial-time adversary is negligible in the following experiment:*

1. *Run* $(PK, TSK) \leftarrow$ ABS.TSetup. *The adversary is given* PK *and access to the* ABS.TRegister$(TSK, \cdot)$ *oracle.*
2. *The adversary can ask for honest attribute authorities to be instantiated using* ABS.ASetup. *For each of these, the adversary receives only the public key* APK *and gets access to a* ABS.KeyGen$(ASK, \cdot, \cdot)$ *oracle. The adversary can also instantiate (corrupt) attribute authorities and publish public keys for them.*

3. *The adversary gets access to the alternate signing oracle* $\mathsf{AltSign}(PK, \cdot, \cdot, \cdot)$.
4. *At the end the adversary outputs* (m, Υ, σ).

Let $\mathcal{A}_{\mathsf{uid}}$ *be the set of* $u \in \mathbb{A}$ *such that the adversary queried the* ABS.KeyGen *oracle on* (uid, u). *Let* $\mathcal{A}_0$ *be the set of possible attributes corresponding to corrupt attribute authorities. Then the adversary succeeds if* σ *verifies as a valid signature on* (m, Υ)*, and* (m, Υ) *was never queried to the signing oracle, and* $\Upsilon(\mathcal{A}_0 \cup \mathcal{A}_{\mathsf{uid}}) = 0$ *for all* uid *queried to the* ABS.TRegister *oracle.*

E.1 Construction

All our constructions generalize to the multi-authority setting. In the case of Schemes 1, 2 and 3, recall that the credential-bundles are implemented as signatures by the attribute authority on (nonce,attribute) pairs. In the multi-authority setting each attribute authority publishes their own signature verification key. The nonce will be derived deterministically (using a collision-resistant hash function) from the identity uid of the user, so that all authorities agree on the same nonce. The signature trustee publishes its own signature verification key and a CRS for the NIWI argument of knowledge. The NIWI system will be used to prove possession of sufficient attributes (valid signatures, under different verification keys) or a signature on the (message, predicate) pair under the the verification key of the signature trustee. With this modification, the rest of the construction is almost identical to that in the case of the single authority setting.

Our final scheme, which is secure in the generic group model, also extends to the multi-authority setting. Below we sketch in more detail the modifications required for this.

ABS.TSetup: Here the signature trustee selects the cyclic groups G and H, generators $g, C, h_0, \ldots, h_{t_{\mathsf{max}}}$, hash function $\mathcal{H}$, and $A_0 = h_0^{a_0}$, as in the single-authority setting. In addition, it generates a signature key-pair $(TSig, TVer)$ for a (conventional) digital signature scheme. The private key is $TSK := (a_0, TSig)$, and the public key is $PK := ((G, H), \mathcal{H}, g, A_0, h_0, \ldots, h_{t_{\mathsf{max}}}, C, TVer)$.

ABS.TRegister: Given uid, draw at random $K_{\mathsf{base}} \leftarrow G$. Let $K_0 := K_{\mathsf{base}}^{1/a_0}$, where a_0 is retrieved from TSK. Output $\tau := (\mathsf{uid}, K_{\mathsf{base}}, K_0, \rho)$ where ρ is (conventional) signature on $\mathsf{uid} \| K_{\mathsf{base}}$ using the signing key $TSig$ (also retrieved from TSK).

ABS.ASetup: Choose $a, b \leftarrow \mathbb{Z}_p$ and compute $A_j = h_j^a$, $B_j = h_j^b$ for $j \in [t_{\mathsf{max}}]$. The private key is $ASK := (a, b)$ and the public key is $APK := \{A_j, B_j \mid j \in [t_{\mathsf{max}}]\}$.

ABS.KeyGen: The token verification $\mathsf{ABS.TokenVerify}(PK, \mathsf{uid}, \tau)$ verifies the signature contained in τ using the signature verification $TVer$ in PK. $\mathsf{ABS.KeyGen}(ASK, \tau, u)$ extracts K_{base} from τ, and using (a, b) from ASK, computes $K_u := K_{\mathsf{base}}^{1/(a+bu)}$.

The key K_u can be checked for consistency using $\mathsf{ABS.KeyCheck}(PK, APK, \tau, K_u)$, which checks that $e(K_u, A_j B_j^u) = e(K_{\mathsf{base}}, h_j)$ for all $j \in [t_{\mathsf{max}}]$, where A_j and B_j are from APK.

ABS.Sign, ABS.Ver: These algorithms proceed verbatim as before, except where $(A_j B_j^{u(i)})$ is used (corresponding to the attribute $u(i)$ associated with the ith row of the monotone span program), we use $A_{ij} B_{ij}^{u(i)}$ where A_{ij} and B_{ij} are A_j and B_j from APK (as described in ABS.ASetup above) published by the authority $\mathsf{auth}(u(i))$ who owns the attribute $u(i)$.

In the above construction we used a τ which contained a certificate from the signature trustee binding K_{base} to uid. The need for this certificate can be avoided if we derive K_{base} as $K_{\mathsf{base}} = R(\mathsf{uid})$, where $R : \{0,1\}^* \rightarrow G$ is a hash function modeled as a random oracle. We use a random oracle because it is important that users have no advantage in computing the discrete logarithms of their K_{base} values. This eliminates the need for a user to present the token to the attribute authorities, and the need for token

verification, because the attribute authorities could themselves derive the K_{base}. We stress that in our construction, we do not employ a random oracle anywhere, except for this optional efficiency improvement.

E.2 Using Multi-Authority ABS

As described above, ABS can support multiple, mutually independent (and possibly distrusting) agents who can set up their own signature infrastructure, and multiple agents who can issue their own attributes to users. To illustrate how ABS operates in such a setting, we return to the example introduced in the beginning of this section. Recall that Alice wishes to endorse her message with a claim which includes attributes owned by different attribute authorities like Facebook, Orkut, Princeton University, Yale University and the American Sociological Association. Alice needs to choose one or more signature trustees under whose system she will provide the signatures. Suppose Alice is aware that most of her potential readers use Google or the Department of Motor Vehicles (DMV) as trusted signature-trustees. Then Alice can go about endorsing her story as follows:

1. Alice registers herself with Google and the DMV (using ABS.TRegister). These trustees would use their idiosyncratic ways to bind the user with a user ID. For instance the DMV could use the user's driver's licence number and Google could use the user's social security number. Alice gets two tokens τ_{Google} and τ_{DMV} this way. We stress that the tokens issued by the trustees are *public*. As such it is not important for the trustees to verify the identity of a user while registering.

2. Alice happens to be a professor at Yale, and is certified by the American Sociological Association as an expert on online social networks. To obtain appropriate attributes, first she approaches Yale's attribute authority, and presents her tokens from Google and the DMV. For Yale to be able to issue her attributes under these trustees, Yale needs to have the trustee's public-keys. Further, Yale should be satisfied that Alice is indeed the person who possesses the user ID mentioned in the tokens. We shall assume that the Yale can verify the social security number and licence number of all Yale faculty. After verifying Alice's identity and the tokens she presented, using Google and DMV's trustee public-keys, Yale can issue corresponding attribute keys on the attribute "Professor at Yale" (for simplicity we ignore the fact that Alice is untenured, and Yale would only issue an attribute saying Professor at Yale in 2008). Similarly the American Sociological Association will issue Alice keys for the attribute "Expert on online social networks" under the two trustees. Again, the ASA will need to be able to verify Alice's social security number and driver's licence for this, and have access to Google and the DMV's public trustee keys.

3. Alice has already registered herself with Google and the DMV and obtained her tokens. Later, when she has prepared her anecdote — which we shall denote simply by m — she can decide what claim to attach to it. As mentioned above, she decides on the claim (which we shall call Υ) involving additional attributes owned by the attribute authorities Facebook, Orkut and Princeton (from whom she does not have any attributes). Using her attributes from Yale and the American Sociological Association, she can successfully prepare a pair of signatures σ_{Google} and σ_{DMV} on m using Υ. For this she will need access to the public keys of Facebook, Orkut and Princeton (but need not have interacted with them otherwise). In describing Υ, each attribute is clearly marked as owned by the corresponding attribute authority, so that a verifier knows which public keys are to be used. Further, σ_{Google} and σ_{DMV} include the information that the signature trustee for that signature is Google and the DMV respectively.

4. Suppose Alice has anonymously published $(m, \Upsilon, \sigma_{\mathrm{Google}}, \sigma_{\mathrm{DMV}})$ on the internet. A user in India who trusts Google (but does not know if DMV can be trusted) can verify σ_{Google} and be convinced that the message was endorsed by someone possessing adequate attributes as claimed. For this she should

have access to the public keys issued by all the attribute authorities (Facebook, Orkut, Princeton, Yale and the American Sociological Association).

As an orthogonal issue, this user might believe that Princeton University's attribute authority has been hacked, and an attribute from that authority should not be trusted. In this case she does not attach any significance to the part of the claim (Professor at Princeton OR Professor at Yale).

In this example, Alice herself need not have trusted all the signature trustees. Indeed, she could be concerned that Google is interested in knowing who signed the message, or which attributes were used to sign them. Further, Orkut's attribute authority could be colluding with Google's signature trustee. But even so, the perfect privacy guarantee assures Alice that her signature does not contain any information other than the message and the claim-predicate (and other public information).

Finally, we point out that it is important to use user IDs (social security number or licence number) which cannot be shared among multiple individuals. To see this, suppose Google used an e-mail address as the user ID. Also suppose Alice and her friend Bob shared the e-mail address alice.and.bob@gmail.com . Yale could verify that the e-mail address indeed belongs to Alice. But, meanwhile Bob, who happens to be a professional chess player, can get an attribute Top-100 Chess Player from the World Chess Federation, also under the same user ID and token from Google, because the World Chess Federation verifies that the user ID indeed belongs to Bob. Thus, if they could share a user ID, Alice and Bob would be able to pool their attributes together and endorse messages claiming to have attributes satisfying Professor at Yale AND Top-100 Chess Player.

F Simulation-Extractable Identity-Based NIZK and ABS

General construction of ID-NIZK Below follows a formal description of the simulation extractable identity-based NIZK scheme (ID-NIZK for short) outlined in Section 6:

ID-NIZK.Setup**:** Run $crs \leftarrow$ ID-NIZK.Setup and $(vk, sk) \leftarrow$ DS.KeyGen. Publish $crs' = (crs, vk)$.

ID-NIZK.SimSetup**:** Run $(crs, \psi) \leftarrow$ ID-NIZK.SimSetup and $(vk, sk) \leftarrow$ DS.KeyGen. Publish $crs' = (crs, vk)$, and use $\psi' = (\psi, sk)$ as the trapdoor.

ID-NIZK.Prove$(crs'; \Phi; id; w)$**:** Define $\Phi'_{vk,id} := \exists\, w, \sigma : \Phi(w) \vee$ DS.Ver$(vk, \mu, \sigma) = 1$, where μ is an encoding of (id, Φ). Output the result of NIWI.Prove$(crs; \Phi'_{vk,id}; w)$.

ID-NIZK.Verify$(crs'; \Phi; id; \pi)$**:** Define $\Phi'_{vk,id}$ as above, then output the result of NIWI.Verify$(crs; \Phi'_{vk,id}; \pi)$.

ID-NIZK.Extract(crs', ψ', π)**:** Output the result of NIWI.Extract$(crs, \psi; \pi)$.

ID-NIZK.Simulate$(crs', \psi'; id; \Phi)$**:** Define $\Phi'_{vk,id}$ as above, then compute $\sigma \leftarrow$ DS.Sign$(sk, \mu = (id, \Phi))$. Output the result of NIWI.Prove$(crs; \Phi'_{vk,id}; \sigma)$.

Now we argue that the above construction indeed gives an ID-NIZK.

First, the crs' output by ID-NIZK.Setup is indistinguishable from that of ID-NIZK.SimSetup, directly by the security of NIWI.Setup, NIWI.SimSetup. Next, simulated proofs are indistinguishable from legitimate proofs by the witness indistinguishability of NIWI.Prove.

Finally, we must show that if an adversary has access to an oracle for ID-NIZK.Simulate and outputs a valid (verifying) proof π^* on Φ^* under id^*, where (id^*, Φ^*) have never been queried to ID-NIZK.Simulate, then ID-NIZK.Extract outputs a witness for Φ^* with overwhelming probability. This is easy to see by simulating such an experiment within the signature scheme's unforgeability experiment. Each time a simulated proof is needed, we request a signature on (id, Φ) and use it to generate a simulated signature. By the correctness of NIWI.Extract, when the adversary outputs π^*, we obtain a witness for $(\Phi^*)'_{vk,id^*}$ with

overwhelming probability. This is either a witness for Φ^*, or a signature on (id^*, Φ^*). However, the latter would constitute a signature forgery, thus with overwhelming probability we do in fact obtain a witness for Φ^*, as desired.

Finally, to prove Theorem 6, we describe an efficient application of the above construction, using Groth-Sahai proofs and Waters signatures. Groth-Sahai proofs are zero-knowledge only for statements that can be expressed (perhaps after adding extra variables) as pairing equations of the form:

$$\prod_i e(\mathcal{X}_i, B_i) \prod_j e(A_j, \mathcal{Y}_j) \prod_{i,j} e(\mathcal{X}_i, \mathcal{Y}_j)^{\gamma_{ij}} = e(g, h)$$

where $\mathcal{X}_i, \mathcal{Y}_j$ are the formal variables, and A_i, B_j, γ_{ij} are public coefficients.

Suppose $(A, V_0, \ldots, V_n)$ is the Waters public key, and let $(\sigma_1, \sigma_2) \in \mathbb{G} \times \mathbb{H}$ be a candidate signature. Our approach is to develop a proof of the following statment:

$$\begin{aligned} \prod_i e(\mathcal{X}_i^\beta, B_i) \prod_j e(A_j, \mathcal{Y}_j^\beta) \prod_{i,j} e(\mathcal{X}_i^\beta, \mathcal{Y}_j^\beta)^{\gamma_{ij}} \\ \times e(V^*, \sigma_2^{1-\beta})\, e(g^{1-\beta}, A) = e(g^\beta, h)\, e(\sigma_1, h^{1-\beta}) \\ \wedge \beta \in \{0, 1\} \end{aligned}$$

In the above expression, $V^* = V_0 \prod_i V_i^{\mu_i}$, a public coefficient when μ is known. This new statement says that either the original statement was satisfied, or (σ_1, σ_2) is a valid signature on μ, as desired. We can rewrite any term of the form $e(C, z^{1-\beta})$ as $e(C, z) \cdot e(C^{-1}, z^\beta)$, and thus we must commit to additional values: each $\mathcal{X}_i^\beta$ and $\mathcal{Y}_j^\beta$, as well as $g^\beta, h^\beta, \sigma_1, \sigma_1^\beta, \sigma_2, \sigma_2^\beta$. We prove the expression above, using the commitments to these values.

Finally, we add additional pairing equations to prove that $\beta \in \{0, 1\}$. First, equations $e(\langle g^\beta \rangle, \langle h^\beta \rangle) = e(\langle g^\beta \rangle, h)$ and $e(\langle g^\beta \rangle, h) = e(g, \langle h^\beta \rangle)$. We must also prove that the commitments to $\mathcal{X}_i^\beta, \mathcal{Y}_j^\beta, \sigma_1^\beta, \sigma_2^\beta$ are consistent with β, using equations of the form:

$$e(\langle \mathcal{X}_i \rangle, \left\langle h^\beta \right\rangle) = e(\left\langle \mathcal{X}_i^\beta \right\rangle, h); \qquad e(\left\langle g^\beta \right\rangle, \langle \mathcal{Y}_j \rangle) = e(g, \left\langle \mathcal{Y}_j^\beta \right\rangle).$$

We note that this transformation can be applied to every pairing equation in the Groth-Sahai proof, re-using the same $\beta, \sigma_1, \sigma_2$, and shared $\mathcal{X}_i^\beta, \mathcal{Y}_j^\beta$ variables. Thus the overhead of the transformation is linear in the number of variables, and independent of the number of pairing equations proven on those variables.

G Applications

G.1 Attribute-Based Messaging

Attribute-Based Messaging or ABM (e.g. [4]) provides an example of a quintessential attribute-based system which demands new cryptographic primitives for achieving its natural security goals. In an ABM system, the set of users to whom a message is addressed is not specified by their identities, but by an "attribute-based address": that is, a predicate on the attributes, such that the intended receivers are the users whose attributes satisfy the predicate. An ABM system can also ensure that only users whose attributes satisfy certain conditions can send messages to certain other users. All this must be facilitated without requiring the users to be aware of each other's identities or attributes.

End-to-End guarantees in ABM The goals of an ABM system can be achieved using trusted entities. But as in other communication systems, the users may require an *end-to-end* guarantee on these properties, independent of the entities involved in delivering the messages. That is, (1) senders would like to encrypt

their messages so that only users with appropriate attributes can decrypt them, and (2) receivers would like to verify signatures on messages such that only users with appropriate attributes could have signed them; the signer should not be forced to reveal more details about its attributes or identity than what is relevant to the receiver. Note that here the users would be willing to trust the authority that issues the attributes, as a compromised attribute-authority could give all attributes to any user, thereby rendering the above guarantees meaningless.[12]

The first of these issues can be elegantly handled using attribute-based encryption: in particular the *ciphertext-policy attribute-based encryption* of Bethencourt, Sahai and Waters [3] provides just the right cryptographic tool. Their implementation of this encryption scheme was integrated into the ABM system of Bobba et al. [5] and demonstrated to be practical.

However, the second issue of authentication did not have a satisfactory solution until now. To highlight some of the issues involved, we point out shortcomings of some natural proposals using existing cryptographic tools:

- *Using certificates:* For each attribute that a user has, the attribute authority gives the user a new signing key and a certificate binding the attribute to the corresponding signature verification key. Then, to sign a message using her attributes, a user simply signs it using the signing key from the attribute authority and presents (a subset of) the certificates it received.

This achieves the goal of users not having to be *a priori* aware of other users or their attributes. But this "solution" has at least two drawbacks. First, the user has to reveal (a subset of) its attributes, rather than just some predicate of the attributes. Second, even though the user's identity is not directly revealed by the signature, multiple signatures can be linked together as coming from the same user.

- *Using mesh signatures:* To allow signing with non-trivial predicates of attributes, one could consider using the recently developed tool of mesh-signatures [10]. This does indeed provide a perfect privacy guarantee. However, this approach fails a crucial unforgeability requirement: multiple users can pool their attributes together and create signatures which none of them could have by themselves produced.
- As a "fix" to the above collusion problem, one might consider using disjoint attribute universes for different parties. This would indeed prevent collusion, and would still retain the privacy guarantee that the signature does not reveal how the claim-predicate was satisfied. However this is also not a satisfactory solution, as it allows multiple signatures to be identified as being generated by the same user.

Using an ABS scheme simultaneously overcomes all these problems, and achieves (a) perfect privacy and unlinkability, and (b) collusion resistant unforgeability. In integrating ABS into ABM, the message path of the ABM need not be altered. But in the attribute keying path, during registration the users should obtain keys for signing and verification as well (in addition to keys for encryption and decryption). An implementation would follow the description in Section A.

ABS for Access Control in ABM As suggested above, the primary application of ABS in an ABM system would be to obtain end-to-end authentication guarantees. But in addition, ABS could be used by the system to implement access control: a typical ABM system will require that messages to some addresses be not delivered unless the sender has attributes satisfying a certain policy. That is, an attribute-based access control mechanism must decide whether to allow a messaging attempt from a sender or not, depending on the attributes of the sender and the attribute-based address of the message.

In the current implementations this is achieved by the sender authenticating itself to a central server in the message path, who then consults the attribute database to determine whether the sender's attributes satisfy the requisite predicate. This requires this central server having access to the user's identity as well as

[12] In an ABM system, the entities in the message path are significantly more vulnerable than the attribute authority, because they need to stay online and be involved in every message delivery. The attribute authority interacts with users only when issuing them attributes.

attributes. This in general is not considered a serious issue, because anyway the attribute database has to be queried for obtaining the list of recipients.

However, it is possible that the attributes of the receivers used in the addresses are not the same (and may not be as sensitive) as the attributes of the sender used to determine access privileges. In such a scenario, using ABS can completely eliminate the need to query the database regarding the more sensitive attributes. Instead, for each message, a sender can decide what predicate regarding its attributes is to be revealed, then sign the message with that predicate using ABS. A server in the message path can ensure that the claimed predicate satisfies the system's sending policy, and if the signature verifies, deliver the message. Note that since this signature verification can be carried out using public keys, it can be done at one of the many points in the message path, instead of at a centralized server.

In a complex ABM system one might require the senders to include two ABS tags with every message — one intended for the message delivery agents, and one for the end recipient. The former would typically involve a claim-predicate that is independent of the contents of the message, and simpler (and hence faster to verify). The signature intended for the receiver could be dependent on the message and more complex; note that this signature is verified by the individual users locally, without putting load on central servers.

ABS for inter-domain ABM There are several engineering and cryptographic challenges in implementing a truly inter-domain ABM system. Neither the current implementations of ABM nor attribute-based encryption schemes known today fully support multiple attribute authorities (so that a user can use attributes from different attribute-authorities in the same message). For instance, Chase's proposal [12] for multi-authority attribute-based encryption (originally for the schemes in [31, 18], but can be extended to the one in [3]) requires all the attribute-authorities to share secret keys with a central authority, thereby requiring the central authority to trust all the attribute authorities.

Remarkably, however, the multi-authority version of our ABS scheme is readily amenable to a full-fledged inter-domain setting. There can be multiple attribute-authorities and signature-trustees who need not trust each other. It is safe for a signer to produce signatures using keys from untrusted trustees, and it is possible to form signatures involving attributes from multiple (untrusted) attribute-authorities; the verifier needs to trust just one of the signature-trustees used.

G.2 Other Applications

ABS offers a unique combination of features that makes it suitable for several other scenarios as well. We point out a few potential applications. These are only meant to illustrate different possibilities of ABS, and not claimed to be solutions for these problems in their most general setting.

Attribute-Based Authentication Consider a server which allows clients to connect to it and carry out transactions depending only on the client's attributes. A client who wishes to carry out a transaction may wish to reveal only minimal information about its identity of attributes as required by the system policy. ABS provides an immediate solution: to establish an authenticated session, the server sends a unique *session-id* to the client. The client responds to the server over an encrypted channel with an ABS signature on (*session-id*, *session-key*), where *session-key* consists of freshly generated keys for symmetric-key encryption (with semantic security) and MAC. After verifying the ABS signature, the server grants the client access depending on the claim-predicate of the ABS tag. All further communication in the session is carried out using the session-key.

Leaking Secrets The classical application for which the notion of ring-signatures was developed by Rivest, Shamir and Tauman [28] is "leaking secrets." In a ring signature the signer can endorse a message and

attach a claim that it is one of the identities (or attributes, in our case) in some set. This is indeed an instance of ABS, with a particularly simple class of claim-predicates, namely disjunctions. *Mesh signatures* [10] are an extension of this concept that allow a rich class of claim-predicates (the same class of claim-predicates supported in our construction). However, when allowing this larger class of predicates an issue arises which is not present in the ring signature setting — namely, the possibility of multiple users colluding to pool their attributes together. Note that when restricted to disjunction, having any one attribute is enough to satisfy the claim, and pooling attributes does not allow a coalition to satisfy any new disjunctions. But for any claim-predicate other than a disjunction, collusion can indeed help. In [10] collusion is considered legitimate: indeed attributes there are considered to be individual identities, and multiple users *must* collude to obtain multiple attributes.

ABS goes beyond mesh signatures and provides collusion-resistance. (If certain groups of users must be allowed to collude, an ABS scheme would treat them as a single user; indeed if there is only one user in the system, an ABS scheme degenerates to a mesh signature scheme.) In that sense ABS is a more appropriate generalization of ring signatures to complex claim-predicates in many settings.

The semantics of leaking a secret with an ABS signature is that a *single entity* who has attributes satisfying a certain claim has endorsed the message. Here it is important that the ABS allows claims to be in terms of some arbitrary attributes *chosen* by the signer (presumably designed to obscure their identity), as well as some attributes the signer might indeed possess.

Trust Negotiations Trust-negotiation between two parties is a well-studied problem in the setting of an attribute-based system. From a theoretical point of view, the problem is a special case of secure two-party computation. However much of the research on this problem focuses on obtaining very efficient solutions when possible. A standard approach to such an efficient protocol is a carefully designed sequence of rounds in which the two parties progressively reveal more and more of their attributes. At its simplest, this can mean simply revealing one or more of one's own attributes in a verifiable manner. However, several recent works also consider cryptographic approaches to trust negotiation that give more privacy to users than is achieved when they simply take turns revealing their attributes [26, 17]. ABS permits a sophisticated way to reveal partial information about one's attributes that is natural for this setting: one party can prove to the other party that her attributes satisfy some complex predicate.

Being able to bind a message with such a proof about one's attributes, as ABS permits, allows for a robust turn-based trust negotiation protocol. At every step of the negotiation, there is an "ephemeral key" for secure communication (private-key encryption and MAC). At each step, the active party picks a new ephemeral key, signs it using ABS with the claim that he or she wants to reveal at that step, and sends it securely (using the ephemeral key from the previous step) to the other party, who verifies the signature. Using new ephemeral keys at each step prevents man-in-the-middle attacks by an adversary who has enough attributes to carry out only the first few steps of the negotiation.