Attribute-Based Signatures: Achieving Attribute-Privacy and Collusion-Resistance

Hemanta Maji* Manoj Prabhakaran* Mike Rosulek*

April 15, 2008

Abstract

We introduce a new and versatile cryptographic primitive called *Attribute-Based Signatures* (ABS), in which a signature attests not to the identity of the individual who endorsed a message, but instead to a (possibly complex) claim regarding the attributes she posseses. ABS offers:

– A strong unforgeability guarantee for the verifier, that the signature was produced by a *single* party whose attributes satisfy the claim being made; i.e., not by a collusion of individuals who pooled their attributes together.

– A strong privacy guarantee for the signer, that the signature reveals nothing about the identity or attributes of the signer beyond what is explicitly revealed by the claim being made.

We formally define the security requirements of ABS as a cryptographic primitive, and then describe an efficient ABS construction based on groups with bilinear pairings. We prove that our construction is secure in the generic group model. Finally, we illustrate several applications of this new tool; in particular, ABS fills a critical security requirement in attribute-based messaging (ABM) systems.

A powerful feature of our ABS construction is that unlike many other attribute-based cryptographic primitives, it can be readily used in a *multi-authority* setting, wherein users can make claims involving combinations of attributes issued by independent and mutually distrusting authorities.

1 Introduction

Attribute-based systems are a natural fit for settings where the roles of the users depend on the combination of attributes they possess. In such systems, the users obtain multiple attributes from one or more *attribute authorities*, and a user's capabilities in the system (e.g., sending messages, access to a resource) depend on their attributes. While offering several advantages, attribute-based systems also present fundamental cryptographic challenges. For instance, to provide *end-to-end* secure communication in an attribute-based messaging system, it must be possible 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 [11, 3]. In this work, we provide a solution for authentication.

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 privacy requirements, similar to more recent signature variants like group signatures [9], ring signatures [18], and

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

mesh signatures [6]. The common theme in all these new kinds of signature primitives is that they allow the signer fine-grained control over *how much of her personal information is revealed by the signature.* For instance, in a group or ring signature, the signature only reveals the fact that the message was endorsed by one of a set of possible signers. It is conceptually useful to think of these primitives as instances of a more abstract "claim-and-endorse" primitive. A claim-and-endorsement σ includes a claim clm and a message msg, with the following (informally stated) semantics:

– **Unforgeability.** By verifying σ, one is assured that the message msg was indeed endorsed by a party who satisfies the condition described in clm.

– **Privacy and Unlinkability.** σ reveals no information about the signer other than the fact that it satisfies clm, even when σ is considered among other signatures. In particular, different signatures cannot be identified as generated by the same party.

Such a primitive has several applications, including authentication, simple trust-negotiations, and "leaking secrets." Our main motivation for developing such schemes comes from the challenging authentication requirements in an Attribute-Based Messaging (ABM) system (see Section 5.1).

1.1 Our Results

The new tool we develop, called *Attribute-Based Signatures* or ABS for short, is a claim-and-endorse primitive that supports claims of the form: "a single user, whose attributes satisfy the predicate Υ, endorsed this message," where the attributes in question can even come from different attribute authorities. As a simple illustrative example (further developed in Section 4.3), 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).

ABS allows her to endorse her anecdote using this claim, without having to reveal how she satisfies the claim. A moment's reflection will reveal that this a rather challenging requirement. Alice could acquire these attributes from multiple attribute-authorities who may not trust each other, or even be aware of each other. Nor might all these authorities trust a common central agency. To make matters more challenging, Alice may have never interacted with Facebook's attribute-authority, and yet she may wish to use an arbitrary attribute string from Facebook as part of her claim.

Another challenge, and an important part of the soundness requirement, is *collusion resistance*: different parties should not be able to pool together their attributes to sign a message with a claim which none of them satisfy alone. For instance, if Alice has an attribute Facebook user for 2 years, and her friend Bob has an attribute Has 100 Facebook friends, they should not be able to sign a message claiming to have *both* the attributes.

We formally define the requirements needed for an ABS scheme — first, in a simpler setting with only one attribute authority, and then in the more general multi-authority setting. We also give an efficient construction that achieves our definitions. We prove our scheme secure in the generic group model.

1.2 Related Work

We use tools and techniques used recently for constructing attribute-based encryption schemes [19, 11, 3] and also "mesh signatures" [6]. Our construction somewhat resembles the one in [6], but crucially differs in that we prevent collusion, where as mesh signatures explicitly allow collusion.

Interestingly, 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 required primitive is a mesh signature, which is a non-trivial cryptographic problem in itself.

Though not pursued in our construction, a line of work relevant to ABS involves non-interactive zero-knowledge (NIZK) proofs (with appropriate properties like simulation-soundness). Indeed the privacy property of the claim-and-endorse primitives is similar to that of a zero-knowledge proof. NIZKs have been used in the context of group signature constructions [1, 7, 12]. Further, [14] proposed a generic notion called identity-based NIZK that, we observe, can in fact be adapted to be an ABS scheme. However, this generic construction will be very inefficient. We point out that in *some* cases specialized NIZK proofs have been used as part of *efficient* schemes (like the group signature scheme of [7] which uses the NIZK proofs of [13]). But for ABS no such construction is known.

Khader [16, 15] proposes a notion called *attribute-based group signatures*. This primitive hides only the identity of the signer, and *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 [9]); in contrast we require signer privacy to hold against everyone, including all authorities.

A related (but much simpler) notion is of identity-based signatures [20]. 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.

Chase [8] extends attribute-based encryption schemes in [19, 11] to a multi-authority setting (but employing a central authority who, unlike in our setting, shares some private keys with the authorities).

1.3 Organization of the paper

In Section 2 we formally introduce and define the ABS primitive, in the simpler setting of a single attribute authority. Section 3 gives the actual construction, and an overview of our proof of security. (The detailed proof is given in the appendix.) In Section 4 we present the more general, multi-authority definitions for ABS, illustrate its flexibility with an example (Section 4.3), and extend our construction to meet its requirements. In Section 5 we discuss important applications of this primitive. We conclude by outlining some immediate and longer term goals in this line of research.

2 Attribute-Based Signatures

We now formally define attribute-based signatures (ABS). For the sake of expositional clarity, in this section we shall restrict ourselves to the case where there is only one authority who manages the attributes as well as the global parameters of the signature scheme. Later in Section 4, we show how these definitions and our construction can be extended to the more complex case where there are multiple attribute authorities, and illustrated using an example scenario (see Section 4.3).

2.1 Syntax of ABS

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 1 (ABS). *An* Attribute-Based Signature (ABS) scheme *is parametrized by a universe of possible attributes $\mathbb{A}$, and consists of the following four algorithms. All procedures (except possibly* Verify*) are randomized.*

Setup: *The authority obtains a key pair $(PK, MK) \leftarrow \mathsf{Setup}()$, and outputs public parameters PK and keeps a private master key MK.*

AttrGen: *To assign a set of attributes $\mathcal{A} \subseteq \mathbb{A}$ to a user, the authority computes a signing key $SK_{\mathcal{A}} \leftarrow \mathsf{AttrGen}(MK, \mathcal{A})$ and gives it to the user.*

Sign: *To sign a message m with a claim-predicate Υ, and a set of attributes $\mathcal{A}$ such that $\Upsilon(\mathcal{A}) = 1$, the user computes a signature $\sigma \leftarrow \mathsf{Sign}(PK, SK_{\mathcal{A}}, m, \Upsilon)$.*

Verify: *To verify a signature σ on a message m with a claim-predicate Υ, a user runs* Verify *$(PK, m, \Upsilon, \sigma)$, which outputs a boolean value,* accept *or* reject.

For definitional clarity, we have treated the signing key $SK_{\mathcal{A}}$ as a monolithic quantity. But in our construction the signing key consists of separate components for each attribute in $\mathcal{A}$. The Sign algorithm will need only as much of $SK_{\mathcal{A}}$ as is relevant to satisfying the claim-predicate.

2.2 Security and Correctness Definitions

The minimal correctness property of ABS schemes is that honestly-generated signatures pass the verification check:

Definition 2 (Correctness). *We call an ABS scheme* correct *if for all $(PK, MK) \leftarrow \mathsf{Setup}$, all messages m, all attribute sets $\mathcal{A}$, all signing keys $SK_{\mathcal{A}} \leftarrow \mathsf{AttrGen}(MK, \mathcal{A})$, and all claim-predicates Υ such that $\Upsilon(\mathcal{A}) = 1$,*

$$\mathsf{Verify}\Big(PK, m, \Upsilon, \mathsf{Sign}\big(PK, SK_{\mathcal{A}}, m, \Upsilon\big)\Big) = \mathsf{accept},$$

with probability 1 over the randomness of all the algorithms.

We present two formal definitions that together capture our desired notions of security.

Definition 3 (Perfect Privacy). *An ABS scheme is* perfectly private *if, for all $(PK, MK) \leftarrow \mathsf{Setup}$, all attribute sets $\mathcal{A}_1, \mathcal{A}_2$, all $SK_1 \leftarrow \mathsf{AttrGen}(MK, \mathcal{A}_1)$, $SK_2 \leftarrow \mathsf{AttrGen}(MK, \mathcal{A}_2)$, all messages m, and all claim-predicates Υ such that $\Upsilon(\mathcal{A}_1) = \Upsilon(\mathcal{A}_2) = 1$, the distributions $\mathsf{Sign}(PK, SK_1, m, \Upsilon)$ and $\mathsf{Sign}(PK, SK_2, m, \Upsilon)$ are equal.*

If the perfect privacy requirement holds, then a signature does not leak (even to the authority and/or a computationally unbounded adversary) which attributes were used to generate it, nor any other identifying information associated with the particular signer. This holds even if the adversary gets access to the signer's private keys: the signature is simply *independent* of everything except the message and the claim-predicate (and the public-keys of the authority).

Another way to interpret perfect privacy is by considering a "simulator" AltSign: Given (MK, m, Υ), AltSign first runs $SK_{\mathcal{A}} \leftarrow \mathsf{AttrGen}(MK, \mathcal{A})$ for some $\mathcal{A}$ satisfying Υ, and then outputs the signature $\mathsf{Sign}(PK, SK_{\mathcal{A}}, m, \Upsilon)$. Since the correct distribution on signatures can be perfectly simulated without taking any private information as input, signatures must not leak any such private information of the signer.

For convenience, in the experiment below we use AltSign to describe the generation of signatures that the adversary gets to see. This is without loss of generality only if the scheme satisfies perfect privacy. With a slight modification to the experiment, this restriction can be easily removed.

Definition 4 (Unforgeability)**.** *An ABS scheme is* unforgeable *if the success probability of any polynomial-time adversary in the following experiment is negligible:*

1. *Run* $(PK, MK) \leftarrow \mathsf{Setup}$ *and give* PK *to the adversary.*
2. *The adversary is given access to oracles* $\mathsf{AttrGen}(MK, \cdot)$ *and* $\mathsf{AltSign}(PK, \cdot, \cdot)$*.*
3. *At the end the adversary outputs* (m, Υ, σ)*.*

We say the adversary succeeds if (m, Υ) *was never queried to the* AltSign *oracle, and* $\Upsilon(\mathcal{A}) = 0$ *for all* $\mathcal{A}$ *queried to the* AttrGen *oracle, and* $\mathsf{Verify}(PK, m, \Upsilon, \sigma) = \mathsf{accept}$*.*

If the unforgeability requirement holds, then the only valid *new* signatures that a coalition can produce are signatures that a *single* member of the coalition could have legitimately generated by himself. Thus, colluding does not confer any advantage in generating signatures.

Note that in this definition, it is not considered a forgery if the adversary requests a signature on (m, Υ) and then produces a *different* signature on the same pair (such a requirement is usually called *strong unforgeability*).

2.3 Using Attribute-Based Signatures

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.[1]

Mapping Real-life Attributes to ABS 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 (name, 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 (name, 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 [11, 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

[1] 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.

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: Single Authority Version. In the single authority setting, the authority first runs the algorithm 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 AttrGen to generate a signing key $SK_{\mathcal{A}}$ corresponding to the set of attributes posessed 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.[2]

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.

When a user's attribute set changes, the authority can reissue an entire new set of signing keys, generated via AttrGen. 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 (Section 4), 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).

3 Constructing an ABS Scheme

We present a construction of an ABS scheme using cryptographic tools recently developed in the context of attribute-based encryption. Below we briefly describe these tools. Also, before describing the construction, we explain the rich class of credential claims that are supported by our ABS scheme.

[2]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).

3.1 Preliminaries

3.1.1 Groups with Bilinear Pairings

Let G, H, G_T be cyclic (multiplicative) groups of order p, where p is a prime. Let g be a generator of G, and h be a generator of H. Then $e : G \times H \to G_T$ is a *bilinear pairing* if $e(g, h)$ is a generator of G_T, and $e(g^a, h^b) = e(g, h)^{ab}$ for all a, b.

We require a triplet of such groups where problems related to discrete-log are hard. Formally, we shall model them as "generic groups." (See Section 3.1.3.)

3.1.2 Monotone Span Programs

Let Υ 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 u that associates each row of $\mathbf{M}$ with an input variable of Υ, that satisfies the following:

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

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

We call ℓ the *length* and t the *width* of the span program. Every monotone boolean function can be represented by some monotone span program. The size of the signatures in our scheme depends on the dimensions of the claim-predicate's monotone span program. A large class of claim-predicates 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 $\sum_i(t_i - 1) + 1$. As an example, a monotone span program for the predicate given in the Introduction involving 7 attributes and several OR and AND gates (which are binary threshold gates with thresholds 1 and 2, respectively) is given by

$$\mathbf{M} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}^T .$$

$\mathbf{M}$ has 7 rows, labeled by the 7 attributes and 4 columns. $\vec{v}$ corresponding to having the last two attributes (Professor at Yale and Expert on online social networks) will be $\vec{v} = [0\ 0\ 0\ 0\ 0\ 1\ -1]$, so that $\vec{v}\mathbf{M} = [1\ 0\ 0\ 0]$.

3.1.3 Security in Generic Group Model

We formally prove the security of our scheme in the generic group model, introduced by Shoup [21]. The model relies on hardness of problems related to finding the discrete logarithm in a group with bilinear pairings. In the model, algorithms can only manipulate group elements via canonical group operations (including the bilinear pairing). Thus the model formalizes the intuition that a cryptographically secure group provides no exploitable structure other than its group structure.

A proof in the generic group model gives significant confidence in the security of a construction, and in particular rules out attacks which use only the group structure of the underlying group. But it falls short of reducing the security to a well-understood hardness assumption. Indeed, it is the only level of security known for several efficient cryptographic tools used in practice. We leave it

as an interesting theoretical problem to come up with an efficient ABS scheme whose security can be reduced to a standard hardness assumption.

We point out that our construction satisfies correctness and the perfect privacy requirement *unconditionally*; only the unforgeability analysis uses the generic group model. As such, even if an attack is found in future, it does not compromise the privacy of existing signatures.

3.2 Our Construction

We now describe our ABS construction, which 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.[3]

We assume that all parties agree on some canonical efficient method of computing a monotone span program given a claim-predicate Υ, and that there is also an efficient algorithm for computing the vector $\vec{v}$ that corresponds to a given satisfying assignment of Υ.

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:

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

AttrGen: 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:

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

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:

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

[3] 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.

Verify: 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.

3.3 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.

3.3.1 Probabilistic Verification

Using a standard technique, signatures in our 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 $1/p$.

3.4 Security Proofs

Theorem 1. *Our construction is correct (Definition 2) and perfectly private (Definition 3).*

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 $\mathsf{Sign}(PK, SK_{\mathcal{A}}, m, \Upsilon)$ is uniformly distributed among signatures σ, subject to the constraint that $\mathsf{Verify}(PK, m, \Upsilon, \sigma) = 1$. 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 Sign are distributed uniformly in their respective domains and that the signature output by Sign successfully verifies. □

Theorem 2. *Our construction is unforgeable (Definition 4) in the generic group model.*

Proof. The full proof is given in the appendix. Here we sketch the main ideas.

In the generic group model, the adversary can only manipulate group elements by using the canonical group operations, independent of the encoding for group elements. Thus if the adversary is given group elements $g^{\gamma_1}, \ldots, g^{\gamma_t} \in G$ as its only inputs, then each element of G output by the adversary must be of the form $g^{\pi(\gamma_1,\ldots,\gamma_t)}$, where π is a fixed multilinear polynomial.

Suppose the adversary gives a signature that passes the verification checks. Using a standard argument for the generic group model, we first show that if this is to happen with non-negligible probability, then the multilinear polynomials described above must also satisfy corresponding constraints.

Thus our approach is to assume that the multilinear polynomials corresponding to the adversary's output satisfy the appropriate constraints, and then obtain a contradiction. We proceed by arguing that in order to satisfy the constraints, the polynomials must have certain structure (i.e., they can only depend on certain variables).

First, we show that because of the check on the Y and W components of the signature, Y can only depend on a few public key components. In fact, without this check, the scheme is forgeable.

After narrowing down the possibilities for Y, we observe that a successful forgery must have a different value of μ than any signature given by the $\mathsf{AltSign}$ oracle, due to the collision resistance of $\mathcal{H}$. Using this property, we show that none of the multilinear polynomials in question can depend on any part of the signatures given by the $\mathsf{AltSign}$ oracle in the experiment.

Then, we apply the properties of the monotone span programs to show that the claim predicate of the adversary's forgery must be satisfied by one of the adversary's signing keys. But this directly contradicts the forgery condition in the experiment's definition. □

4 Multiple Attribute Authorities

In this section, we describe how to extend our definitions and construction for use in an environment with multiple 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, and may not even be aware of each other. Indeed, some of the attribute authorities may be corrupt, and this should not affect the attributes acquired from other authorities.

Apart from these mutually distrusting attribute authorities, there should be some 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 describe in detail the modifications in the syntax and security definitions of the ABS primitive in detail. We follow it up by an illustrative example involving multiple attribute authorities and signature trustees.

4.1 Definitions

Our main changes in definitions involve separating the key material into pieces originating from different authorities. Further, the syntax now includes new safety checks on the key material

(TokenVerify and KeyCheck) before using it, because (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 5 (Multi-Authority ABS)**.** *A* multi-authority ABS scheme *consists of the following algorithms/protocols:*

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

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

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

AttrGen**:** *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* $\mathsf{TokenVerify}(PK, \mathsf{uid}, \tau)$*. If the token is verified, then it runs* $\mathsf{AttrGen}(ASK, \tau, u)$ *which outputs an* attribute key K_u*. The attribute authority gives K_u to the user.*

The user checks this key using $\mathsf{KeyCheck}(PK, APK, \tau, K_u)$ *and accepts this attribute key only if it passes the check.*

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 obatined a set of keys $\{K_u \,|\, u \in \mathcal{A}\}$ from the attribute authorities, and they have all passed* KeyCheck*. Then the signature σ can be generated using*

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

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.*

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

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

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

Security Definitions. We extend the two ABS security definitions to the multi-authority case. 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, as before, is formulated as a perfect information-theoretic property: for every PK, m, and Υ, the output distribution of $\mathsf{Sign}(PK, \{APK_{\mathsf{auth}(u)} | u \in \mathbb{A}_\Upsilon\}, \cdot, \cdot, m, \Upsilon)$ 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 $\mathsf{KeyCheck}$. In other words, there is a (computationally infeasible) procedure $\mathsf{AltSign}$ such that $\mathsf{AltSign}(PK, m, \Upsilon, \{APK_{\mathsf{auth}(u)} \mid u \in \mathbb{A}_\Upsilon\}))$ is distributed exactly as a valid signature on m with claim-predicate Υ.

Note that this is required to hold no matter which trustee keys and attribute keys are used. In particular, even if the trustee or one or more of the authorities are corrupt, the signer is still guaranteed that the signature does not reveal anything beyond m, Υ and the fact that some signer possessed enough attributes to satisfy Υ.

We also modify the unforgeability definition to account for the case where some of the attribute authorities may be corrupt. Further we allow there to be signature trustees under the control of the adversary with respect to whom the attribute authorities can be required to run ASetup and $\mathsf{AttrGen}$. The unforgeability requirement is with respect to an uncorrupted signature trustee (whose setup is carried out by the security experiment below).

As in the single authority case, in the experiment below, we use an $\mathsf{AltSign}$ procedure as the adversary's signature oracle. As before, this restricts our definition of unforgeability to schemes for which perfect privacy holds, but this restriction can be easily removed at the expense of a more complicated security experiment.

Definition 6. *A multi-authority ABS scheme is* unforgeable *if the success probability of every polynomial-time adversary is negligible in the following experiment:*

1. *Run* $(PK, TSK) \leftarrow \mathsf{TSetup}$. *The adversary is given* PK *and access to the* $\mathsf{TRegister}(TSK, \cdot)$ *oracle.*
2. *The adversary can ask for honest attribute authorities to be instantiated using* ASetup. *For each of these, the adversary receives only the public key* APK *and gets access to a* $\mathsf{AttrGen}(ASK, \cdot, \cdot)$ *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* $\mathsf{AttrGen}$ *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* $\mathsf{TRegister}$ *oracle.*

4.2 Construction

Here we briefly sketch the modifications to our construction required to support multiple attribute authorities.

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 $PK := ((G, H), \mathcal{H}, g, A_0, h_0, \ldots, h_{t_{\mathsf{max}}}, C, TVer)$.

$\mathsf{TRegister}$: 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).

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 $APK := \{A_j, B_j \mid j \in [t_{\mathsf{max}}]\}$.

AttrGen: The token verification $\mathsf{TokenVerify}(PK, \mathsf{uid}, \tau)$ verifies the signature contained in τ using the signature verification $TVer$ in PK. $\mathsf{AttrGen}(ASK, \tau, u)$ 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 $\mathsf{KeyCheck}(PK, APK, \tau, K_u)$, 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.

Sign, Verify: 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 ASetup above) published by the authority $\mathsf{auth}(u(i))$ who owns the attribute $u(i)$.

Simpler set-up using a random oracle. 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 $R : \{0,1\}^* \to G$ 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.

4.3 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 Introduction. 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 $\mathsf{TRegister}$). 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 $(m, \Upsilon, \sigma_{\text{Google}}, \sigma_{\text{DMV}})$ 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.

5 Applications

5.1 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.[4]

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 [6]. 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 2.3.

[4] 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.

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 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 recepient. 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 [8] for multi-authority attribute-based encryption (originally for the schemes in [19, 11], 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.

5.2 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 [18] 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* [6] 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 [6] 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 [17, 10]. 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.

6 Conclusion and Future Work

We introduced a new cryptographic primitive called Attribute-Based Signatures (ABS), and presented a construction that provably meets the requirements of ABS. The security proof we provide is in the generic-group model. The construction is fairly efficient, for reasonably complex claim-predicates. ABS is ideally suited for an Attribute-Based Messaging (ABM) system, as well as possibly in several other scenarios that exhibit a tension between authentication and privacy. Our ABS scheme supports multiple attribute-authorities and multiple signature-trustees, who need not trust each other.

In future work, on the theoretical front, one would like to base the security on a standard hardness assumption, without sacrificing too much of the efficiency. The evident drawback of relying on security in the generic group model is in evaluating the underlying group used by a candidate implementation. In fact, even the "standard" hardness assumptions regarding bilinear groups are not as well-studied as more traditional hardness assumptions, as the algorithmic study of such groups is relatively in its infancy.

On the practical front, the next step would be to implement our ABS scheme and integrate it with an application. The natural candidate is an ABM system like that of [5]. We remark that there are several engineering challenges involved in augmenting such an ABM system so that it can fully exploit the power of ABS, which are beyond the scope of the current 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. In C. Cachin and J. Camenisch, editors, *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 268–286. Springer, 2004. Full version at `http://eprint.iacr.org/2004/252`.

[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 (in preparation), 2008.

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

[7] 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.

[8] 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.

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

[10] 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.

[11] 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.

[12] J. Gröth. 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.

[13] J. Gröth, R. Ostrovsky, and A. Sahai. Perfect non-interactive zero knowledge for NP. In S. Vaudenay, editor, *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 339–358. Springer, 2006.

[14] 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.

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

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

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

[18] 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.

[19] 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.

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

[21] V. Shoup. Lower bounds for discrete logarithms and related problems. In *Eurocrypt '97*, pages 256–266, 1997. LNCS No. 1233.

A Unforgeability Proof

Here we present the full proof of Theorem 2.

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. First, we observe that it can only help the adversary if the groups G and H coincide.

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 a multivariate rational functions in the following indeterminate formal variables:

$$\begin{aligned} \Sigma = &\{a_0, a, b, c, \Delta_0\} \cup \{\Delta_j \mid j \in [t_{\mathsf{max}}]\} \\ &\cup \{x_k \mid k \in [n]\} \cup \{s_i^{(q)}, y^{(q)} \mid q \in [\nu], i \in [l^{(q)}]\} \end{aligned}$$

where ν is the number of signature queries made by the adversary, n is the number of queries made to the $\mathsf{AttrGen}$ oracle, and $l^{(q)}$ is the width of the monotone span program corresponding to the qth signature query.

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:

- Public key components generated by Setup:
  1. 1, representing the generator g.
  2. a_0, representing $A_0 = g^{a_0}$
  3. Δ_0, representing $h_0 = g^{\Delta_0}$.
  4. $\{\Delta_j \mid j \in [t_{\mathsf{max}}]\}$, representing $h_j = g^{\Delta_j}$.
  5. $\{\Delta_j a \mid j \in [t_{\mathsf{max}}]\}$, representing $A_j = h_j^a$.
  6. $\{\Delta_j b \mid j \in [t_{\mathsf{max}}]\}$, representing $B_j = h_j^a$.
  7. c, representing $C = g^c$.

- Signing key components given by $\mathsf{AttrGen}$. Let $\mathcal{A}_k$ be the kth set of attributes queried to $\mathsf{AttrGen}$:

  1. x_k, representing $K_{\mathsf{base}}^{(k)} = g^{x_k}$.
  2. x_k/a_0, representing $K_0^{(k)} = g^{x_k/a_0}$.
  3. $\{x_k/(a+bu) \mid u \in \mathcal{A}_k\}$, representing $K_u^{(k)} = g^{x_k/(a+bu)}$.

- Signature queries. Suppose the q-th signature query is on message $m^{(q)}$ under the predicate $\Upsilon^{(q)}$. Let $M^{(q)} \in (\mathbb{Z}_p)^{l^{(q)} \times t^{(q)}}$ be the corresponding monotone span program, with row labeling $u^{(q)} : [l^{(q)}] \to \mathbb{A}$. Let $\mu^{(q)} = \mathcal{H}(m^{(q)} \| \Upsilon^{(q)})$.

  1. $\{s_i^{(q)} \mid i \in [l^{(q)}]\}$, representing $S_i = g^{s_i^{(q)}}$.
  2. $y^{(q)}$, representing $Y^{(q)} = g^{y^{(q)}}$.
  3. $y^{(q)}/a_0$, representing $W^{(q)} = g^{y^{(q)}/a_0}$.
  4. $\{p_j^{(q)} \mid j \in [t^{(q)}]\}$, where

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

     representing $P_j^{(q)} = g^{p_j^{(q)}}$.

- Queries to the generic group oracle:

  1. 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 functions F_α, F_β, associate with the result the function $F_\alpha + F_\beta$.
  2. When the adversary asks for a group element α (specified by its encoding) to be raised to the power d, where the α is associated with function F_α, associate with the result the function $d \cdot F_\alpha$.

  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 functions of the form listed above, the probability of this happening is at most $O(\nu + \sum_k |\mathcal{A}_k|)/p$, since we can multiply both functions by the common denominator $\prod_q (c + \mu^{(q)}) \prod_{u \in \bigcup_{k \in [n]} \mathcal{A}_k} (a + bu)$ to obtain distinct multivariate *polynomials* with total degree at most $O(\nu + \sum_k |\mathcal{A}_k|)$. Since this probability is negligible, we ignore this case.

Now 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 $\sigma^* = (g^{y^*}, g^{w^*}, g^{s_1^*}, \ldots, g^{s_{l^*}^*}, g^{p_1^*}, \ldots, g^{p_{t^*}^*})$.

Then each of these group elements must be associated with a function which must be a *multilinear* combination of the functions given as input to the adversary (public key, signature query responses, and signing keys).

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

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

for a random assignment of the formal variables. But by a similar argument to above, these constraints can only hold with non-negligible probability if they hold with respect to the *actual functions* – i.e., if the two sides of the constraints are functionally equivalent.

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.

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

We know that $y^*, w^*, s_1^*, \ldots, s_{l^*}^*, p_1^*, \ldots, p_{t^*}^* \in \mathbb{L}(\Gamma)$, 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:

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

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

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

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,

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

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

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

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:

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

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:

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

Finally we conclude that:

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

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 $t_i^* \in \mathbb{H}(X_i)$, 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.