Journal of Computer Security 00 (2009) 1–39
DOI 10.3233/JCS-2009-0383
IOS Press

Secure attribute-based systems

Matthew Pirretti [a], Patrick Traynor [b], Patrick McDaniel [c] and Brent Waters [d]

[a] *Motorola Labs, USA*
E-mail: Matthew.Pirretti@motorola.com
[b] *College of Computing, Georgia Institute of Technology, USA*
E-mail: traynor@cc.gatech.edu
[c] *SIIS Laboratory, CSE, Pennsylvania State University, USA*
E-mail: mcdaniel@cse.psu.edu
[d] *SRI International, USA*
E-mail: bwaters@csl.sri.com

Attributes define, classify, or annotate the datum to which they are assigned. However, traditional attribute architectures and cryptosystems are ill-equipped to provide security in the face of diverse access requirements and environments. In this paper, we introduce a novel secure information management architecture based on emerging attribute-based encryption (ABE) primitives. A policy system that meets the needs of complex policies is defined and illustrated. Based on the needs of those policies, we propose cryptographic optimizations that vastly improve enforcement efficiency. We further explore the use of such policies in two proposed applications: a HIPAA compliant distributed file system and a social network. A performance analysis and characterization of ABE primitives demonstrates the ability to reduce cryptographic costs by as much as 98% over previously proposed constructions. Through this, we demonstrate that our attribute system is an efficient solution for securely managing information in large, loosely-coupled, distributed systems.

1. Introduction

Attributes define, classify, or annotate the datum to which they are assigned. The semantics of an attribute indicate some purpose or characteristic and, when used within larger collections, enable efficient identification and classification of like objects. For example, individuals in enterprise systems are often segregated into groups of common interest or duty based on a given set of attributes [36], e.g., function, department, university. These attributes are then used to associate sets of permissions and tasks to the specified individuals. Existing systems principally rely on the assignment and subsequent enforcement of policies by trusted and often centralized servers. However, these servers are acutely ill-equipped to deal with disconnected and asynchronous clients. Reliance upon centralized servers further limits scalability and mandates a single point of trust.

Attribute-based encryption (ABE) [35], a generalization of identity-based cryptosystems, incorporates attributes as inputs to its cryptographic primitives. Objects

0926-227X/09/$17.00

are encrypted using a set of attributes describing the intended receiver. A principal possessing this subset as part of their pool of attributes can recover the original plaintext. More flexible requirements are achievable through the use of a thresholding primitive, for which only *k-of-n* attributes are necessary to perform decryption. Furthermore, decryption under both the standard and threshold approaches is collusion-resistant as multiple parties are unable to meaningfully pool attributes. Such cryptographic mechanisms allow encryption to inextricably bind expressive, enforceable access policy to objects.

Attribute-based systems have enormous potential for providing data security in distributed environments. Peer-to-peer systems are an example of one such beneficiary: individuals may publish documents that implicitly target those users who are assigned the appropriate attributes. Moreover, such publishing can be completely transparent to the peer-to-peer system. For example, a user Bob looking for employment in the field of secure systems engineering could place a copy of his résumé in publicly accessible web space encrypted with the attributes "secure systems engineering" and "human resources manager". Only potential employers satisfying these attributes would be able to decrypt this information and contact Bob.

In this paper, we develop and evaluate a secure attribute system built on attribute-based encryption (ABE). A descriptive policy system is defined that predicates access on logical expressions over attributes. We show how these policies can be realized through applications of novel ABE constructions. We also demonstrate their semantic depth through their use in two proposed applications: a HIPAA compliant distributed file system and a social network.

We have developed an extensive ABE implementation tailored for the rapid creation of attribute systems – the first known implementation and characterization of such cryptographic constructions. In an effort to aid development and subsequent system use, we perform an in-depth empirical analysis of the input parameter space. Our implementation includes several novel optimizations to the original ABE cryptosystem described by Sahai and Waters [35]. The major operations of the system, including system initialization, key generation, and the encryption and decryption of objects are benchmarked. We then measure the cost of implementing complex attribute policies. Whereas past work has suggested that these constructions were too expensive for use in real systems [33], this analysis shows that such policies are not only feasible, but can also be highly efficient. For instance, we demonstrate that the cost of key generation and encryption can be reduced by more than 80% and 98%, respectively, by using constructions secure in the random oracle model.

The remainder of this paper is organized as follows: Section 2 presents an overview of the cryptographic mechanisms supporting ABE; Section 3 compares ABE systems to PKI systems; Section 4 introduces a descriptive policy system for use in ABE-based systems; Section 5 offers sample policies for two example applications; Section 6 gives the results of our performance analysis; Section 7 examines open problems in the sphere of policy expression in attribute-based systems; Section 8 discusses some of the issues inherent to managing ABE-based systems; Section 9 explores relevant related work; Section 10 offers concluding remarks.

2. Attribute-based encryption

We now give an overview of attribute-based encryption (ABE) algorithms. The Sahai–Waters [35] (ABE) cryptosystem as implemented in this paper is specifically detailed. We focus our efforts on providing the description of the scheme and intuition for its construction. For the proof of security see [35].

Attribute-based encryption can be viewed as a generalization of identity-based encryption (IBE) [8,13,39]. In IBE a user's identity is a string such as "bobsmith@yahoo.com". A party in the system can encrypt a message to this particular user with only the knowledge of the recipient's identity and the system's public parameters. In particular the encryption algorithm does not need to have access to a separate public key certificate of the recipient.

In attribute-based encryption a user's identity is composed of a *set*, S, of strings which serve as descriptive attributes of the user. For example, a user's identity could consist of attributes describing their university, department, and job function. A party in the system can then specify another set of attributes S' such that a receiver can only decrypt a message if his identity S has at least k attributes in common with the set S', where k is a parameter set by the system. Like traditional identity-based encryption, a party in an attribute-based encryption system only needs to know the receiver's description in order to determine their public key, allowing such systems to benefit from the lazy distribution of keys. However, the expressiveness of an ABE system is potentially much more powerful. For example, there could be several different recipients that are able to decrypt a message encrypted for a set S'. Although there could in theory exist even more expressive ABE systems, the threshold constructions described and illustrated in the following sections are sufficiently semantically deep that we can define complex and precise encryption policies.

2.1. ABE algorithms

We now informally specify a threshold attribute-based encryption system as a collection of four algorithms:

Setup(k): The Setup algorithm is run by an authority in order to create a new ABE system. Setup takes as input a threshold value, k and outputs a master key MK and a set of public parameters PK.

Key-Gen(S, MK): The authority executes the Key-Gen algorithm for the purpose of generating a new secret key SK. The algorithm takes as input the user's identity, S, as a set of strings representing a user's attributes and the master-key MK and outputs S's secret key SK.

Encrypt(M, S', PK): The Encrypt algorithm is run by a user to encrypt a message M, with a target set S', and the public parameters. It outputs a ciphertext, C.

Decrypt($C, S', S,$ SK)**:** The Decrypt algorithm is run by a user with identity S and secret key SK to attempt to decrypt a ciphertext C that has been encrypted with S'. If the set overlap $|S \cap S'|$ is greater than or equal to k the algorithm will output the decrypted message M.

We discuss the how the value of k can be varied in Section 4.3.

2.2. ABE constructions

We have investigated the use of two separate ABE constructions: the Sahai–Waters construction and a variant of the Sahai–Waters construction that we refer to as the random oracle construction. The Sahai–Waters construction is from the Sahai–Waters Large Universe system as described in Section 6 of [35]. A more complete and formal explanation of both constructions can be found in Appendix B.

Both constructions use elliptic curves to perform pairing-based cryptography. Bilinear maps (pairings) $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ upon elements of an elliptic curve are the basis for pairing-based cryptosystems. In the Sahai–Waters construction, decryption is possible by performing pairings between k components of a ciphertext and k private key components. We refer the reader to the IBE paper by Boneh–Franklin [8] for more details on pairing-based cryptosystems.

2.2.1. Sahai–Waters construction

We note two observations about the Sahai–Waters construction, which have led us to design the random oracle construction. Both observations pertain to the use of following function used in both the **Key-Gen** and **Encrypt** algorithms:

$$T(i) = g^{x^i} \prod_{j=1}^{n+1} t_j^{\Delta_{j,N}(i)}, \qquad (1)$$

where N is the set $\{1, \ldots, n+1\}$.

Our first observation is that because of T, the **Setup** algorithm must take as input a ciphertext size n in addition to the threshold value k. Without the techniques proposed in Section 4.3, this construction mandates that each ciphertext must contain exactly n attributes and that the threshold must be a fixed value k for all ciphertexts.

Our second observation is based upon our experience in building an implementation of the Sahai–Waters construction. We have found T requires a great deal of computational effort. It is easily seen that the number of exponentiations required to solve T is equal to $n + 1$.

Because of these two observations we have proposed the following modification to the Sahai–Waters construction.

2.2.2. *Random oracle construction*

We drastically reduce computational overhead in the key generation and encryption algorithms by replacing T with a hash function used as a random oracle [5]. A simple argument shows that the random oracle can be "programmed" such that the security proof of Sahai and Waters still holds. We refer the reader to the literature [5,12] for further discussion on the random oracle model.

Implementing T as a random oracle has the following characteristics. First, ciphertexts can contain a variable number of attributes, rather than be required to contain n. Second, the $n+1$ exponentiations needed to solve T in the Sahai–Waters construction have been replaced with a single cryptographic hash.

However, using model that requires the random oracle heuristic results in a slightly weaker security model; the use of random oracles makes the security of the cryptosystem dependent upon the security of the hash function used to compute T. In Section 6, we experimentally compare implementations of the original Sahai and Waters construction with our variant.

2.2.3. *Decryption optimization*

Under both constructions, the dominant operations are pairings followed by exponentiations. Decryption as described by Sahai and Waters has the following form [35]:

$$M = E' \prod_{i \in S} \left(\frac{e(d_i, E_i)}{e(D_i, E'')} \right)^{\Delta_{i,S}(0)}, \tag{2}$$

where e denotes a pairing operation. In the equation above, there are $2k$ pairings and k exponentiations. Decryption can be optimized to reduce the number of bilinear map operations by bringing the Lagrange coefficients in:

$$M = \frac{E' \prod_{i \in S} e(d_i^{\Delta_{i,S(0)}}, E_i)}{e(\prod_{i \in S} D_i^{\Delta_{i,S(0)}}, E'')}. \tag{3}$$

This optimization reduces the number of bilinear map operations from $2k$ to $k+1$ at the expense of increasing the number of exponentiations from k to $2k$. Because bilinear map operations are more computationally intensive than exponentiations, this optimization increases the overall speed of decryption.

3. Attribute key infrastructure

Systems using ABE, which are a generalization of IBE, do not vouch for identity in the traditional sense, as users are represented by the summation of their attributes. Accordingly, as identities are no longer necessarily unique, there is no need to validate bindings between keys and users. Like IBE systems, this alleviates many of the managerial problems found in traditional PKI systems [16] (e.g., user name collisions, per-user revocation). However, new challenges arise. We briefly consider these issues, as well as their similarity to traditional PKI systems. Note that this work is not intended to solve the problems of PKIs, but to apply available approaches where possible and invent others where needed.

The process by which users are certified in an ABE system is analogous to certification in a PKI. Similar to a traditional PKI, a user presents the authority with a set of credentials that prove their right to fulfill an attribute. Instead of mapping a user to an identity, certification establishes that the user fulfills the semantic of the attribute. Such semantics are specific to the supported community (e.g. job function in a business system, clubs belonged to in a social network). This process is repeated for all attributes appropriate to each user. Key distribution is significantly simplified in such a system, as public keys are simply the combination of the cryptosystem's public parameters and attribute names.

The revocation process is significantly different in a ABE system as attributes, not users or keys, are revoked. In fact, there is no way to revoke a user, save revoking all of his attributes. Like traditional PKI systems, revocation can impact all users who either have or use an attribute. Unlike traditional PKI systems, however, the compromise of a particular attribute may not mandate its revocation. As Section 4 details, it is the specific application of multiple attributes that defines policy. The compromise of any single attribute may therefore be a necessary but not sufficient condition for its revocation. Consequently, it may be desirable to revoke all, a subset or none of the compromised user attributes. Explored in depth in Section 8.1, we consider both online and offline revocation approaches.

A superficial reading of the above issues may lead one to falsely conclude that ABE systems must be online. The creation of keys, certification of users, and adding attributes are largely isomorphic to certification issuance operations present in current PKI. Revocation can also be handled offline (however, online approaches such as OCSP [32] are likely to be desirable in some environments). Hence, ABE systems can operate entirely offline in largely the same that current PKI systems do.

4. Attribute policy

We now informally define an expository system for describing encryption policies in attribute-based systems. An *attribute policy* (or just policy throughout) is a specification of cryptographic operations carried out on a plaintext in the attribute-based system. Hence, through encryption, a party is able to embed expressive policies into objects themselves, allowing for the decentralized enforcement of such policies. Note that the following policy description is not particular to the specific constructions of our implementation [34], and is appropriate for defining policy in any ABE system that supports threshold constructions.

4.1. Definition

There are two components central to the definition of policies: attributes and objects. An *attribute* consists of a uniquely identifying string, *Name*, and its hash, $H(\mathit{Name})$. The semantics of the *Name* identifier are irrelevant to the policy itself and will be driven by the application it supports (see Section 5 for examples). The hash is necessary for the ABE construction (see Appendix B), but plays no role in the formulation of policy. We broadly refer to all encrypted or recovered data as *objects*. For example, objects in a distributed file system would be the files that it stores. Conversely, objects in a social network may include a mix of personal communiques (e.g. emails, instant messages, etc.), profile information, and pictures. We refer to the universe of all attributes as the set $A = \{a_1, a_2, \ldots, a_x\}$, the set of objects $O = \{o_1, o_2, \ldots, o_y\}$. Where meaning is obvious or differentiation unnecessary, subscripts may be omitted.

The attribute policy is a specification of the attributes and threshold used to encrypt an object. For example, consider a policy P that mandates encryption using a single attribute a under a threshold of 1. We denote this policy:

$$P = T_1(a). \tag{4}$$

Of course, policies are of most interest when they are applied to objects. The application of the example attribute policy on an object o is denoted:

$$E(o, P) \quad \text{or equivalently} \quad E(o, T_1(a)) \tag{5}$$

which states that o has been encrypted under attribute a using a 1-out-of-1 threshold encryption function. This object can therefore only be decrypted by a user possessing this attribute. Now consider the application of a similar policy P' on o that encrypts using 2-out-of-3 threshold using attributes a_1, a_2, and a_3:

$$E(o, P') \quad \text{or equivalently} \quad E(o, T_2(a_1, a_2, a_3)). \tag{6}$$

Now consider the most general case. An arbitrary policy P'' is defined:

$$P'' = T_k(S) \mid S \subseteq A, S \neq \emptyset, \quad 1 \leqslant k \leqslant |S|, \tag{7}$$

which states that the following must be true for any legal policy, (a) the set of attributes must be a nonempty subset of A and (b) the threshold must at least 1 and no more than the total number of attributes.

Note that policies can be arbitrarily nested. Users can build complex expressions of attributes, thresholds, and logical operators. For example one may wish to combine P and P' over o above to achieve,

$$E\bigl(E(o, P'), P\bigr) \quad \text{or equivalently} \quad E\bigl(E(o, T_2(a_1, a_2, a_3)), T_1(a)\bigr) \tag{8}$$

which states that one must decrypt under policy P first, then P' to recover o. Explored more fully in the following section, logical conjunction and disjunction are expressible through attribute policy. For example, if P_i and P_j are policies, then:

$$P_k = (P_i) \wedge (P_j), \qquad P_l = (P_i) \vee (P_j). \tag{9}$$

The semantics of these policies are straightforward. A logical 'and' policy states that one must be able to decrypt both under both policies to extract the plaintext. The logical 'or' policy requires that one must be able decrypt under either or both of the policies to obtain the plaintext. Such constructions are examined in greater detail in Section 5.

The remainder of this paper explores how we build constructions meeting the semantics of these policies and how they can be applied to build novel and interesting applications.

4.2. *Implementing policy*

Implementing singular attribute or threshold policies is straightforward using ABE constructions. An example of a threshold policy is depicted in Fig. 1. This policy states that decryption is possible if the party performing decryption possess at least three of the following attributes: a_1, a_2, a_3, a_4. This is illustrated by the requisite attributes being fed to the threshold operator T_3. The output of the threshold primitive is the desired policy P_1, which can then be used to define an encryption operation.

We refer to a policy where *k-out-of-k* attributes are required to decrypt an object as an **and logic** policy, and 1-*in-k* attributes as **or logic** policy. These policies can be easily implemented using the thresholding primitive, where the threshold is k in the case of **and logic** and 1 in **or logic**. Figure 2 illustrates an **and** policy. The policy P_2 requires that the party performing decryption must possess all four of the following attributes: a_1, a_2, a_3, a_4. P_2 is implemented by giving the threshold operator T_4 the four required attributes. An **or** policy is trivially similar, and is thus not illustrated.

Expressing policy becomes somewhat more complex when the input policies are not subsets of S, i.e., not expressions over atomic policies. Consider the case of an **or** policy spanning three (possibly complex) policies P_1, P_2 and P_3. In this case, one need only encrypt each of the input objects under each policy and concatenate them together; anyone able to decrypt at least one of these objects should be able to recover the underlying object. Denoting concatenation as "$\cdot$", the ciphertext of an object o_i encrypted under a policy $P_1 \vee P_2 \vee P_3$ would be $E(o_i, P_1) \cdot E(o_i, P_2) \cdot E(o_i, P_3)$.

Now consider the case of an **and** policy spanning three (possibly complex) policies P_1, P_2 and P_3. One cannot simply use a threshold as above because the input policies do not reflect a threshold over atomic attributes. Hence, another construction must be used. Observe that we can achieve **and** semantics by sequentially encrypting the object with each policy. Thus the policy $P_1 \wedge P_2 \wedge P_3$ would be $E(E(E(o_i, P_1), P_2), P_3)$.

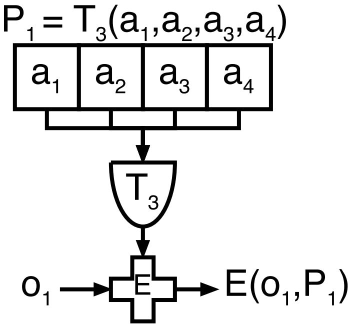

Fig. 1. Encryption using threshold policy P_1. Object o_1 can only be decrypted by a principal in possession of at least three of the requisite attributes.

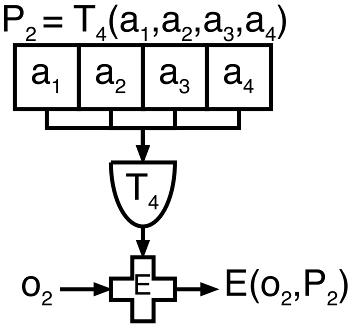

Fig. 2. Encryption using **and** policy P_2. Object o_2 can be decrypted by principals who possess all four requisite attributes.

This satisfies the policy semantic because only principals which possess the underlying attributes satisfying all policies can recover the plaintext.

Conjunction and disjunction constructions can be nested arbitrarily. In Figs 3 and 4 we illustrate policies that use both **and logic** and **or logic**. Specifically, in the **and-or** policy, any party performing decryption must possess either the attributes a_1 and a_2 or the attributes a_3 and a_4. The **or-and** policy requires the decrypting party possess a_1 or a_2 in addition to attribute a_3.

Observe that the conjunction constructor has a weaker security model than the original ABE constructions, where the base objects are encrypted under attributes. Whereas ABE encryption prevents any collusion attack, it is possible for adversaries to collude to recover the plaintext in this construction. To illustrate, in the **or-and** example[1] in Fig. 4, two colluding parties satisfying P_7 and P_8 independently can recover the plaintext. The first adversary need decrypt the outer encryption using its

---

[1] This policy expression could be optimized to reduce the required number of encryptions. Specifically, $(a_1 \vee a_2) \wedge a_3 \equiv (a_1 \wedge a_3) \vee (a_2 \wedge a_3)$. This optimization would require two encryptions with T_2, as

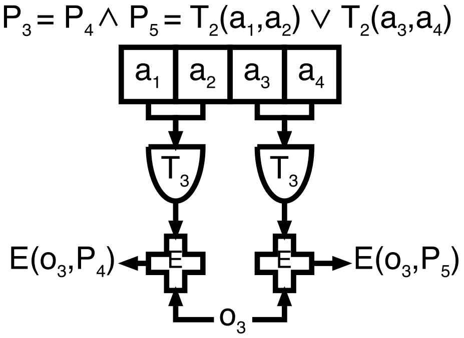

Fig. 3. Encryption operation using **and-or** policy P_3. Principals who possess either attributes a_1 and a_2 or a_3 and a_4 are capable of decrypting object o_3.

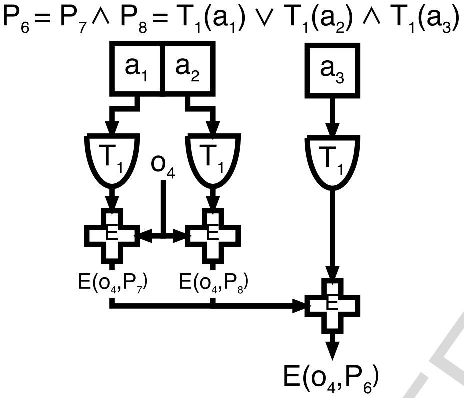

Fig. 4. Encryption operation using **or-and** policy P_4. A principal possessing attribute a_3 and either a_1 or a_2 may decrypt object o_4.

a_3 assignment, then pass the inner $E(o_4, T_1(A_1))$ ciphertext to the second adversary who can then decrypt using a_1 to recover the original plaintext o_4. Work to improve this aspect of the associated cryptographic constructions is currently under way. Until then, our constructions are no weaker than standard cryptographic methods e.g., public key cryptosystems.

4.3. Extending the flexibility of ABE

ABE natively supports a *k-of-n* threshold primitive. However the cryptographic constructions discussed in Appendix B mandate that k be a fixed constant across all

---

opposed to four encryptions with T_1. Such logic expression reductions have been thoroughly studied by other works [23]. We discuss policy embodiment briefly in Section 8.3.

ciphertext objects created by a given attribute system. Further, for the implementation without random oracles, the number of attributes in each ciphertext n, must also be fixed. These requirements greatly limit the liberty at which principals can draft policies; each policy must be created with a single type of threshold primitive. Thus, if $k = 4, n = 4$, then all policies would have to be written using T_4 threshold operators and each ciphertext would have to contain exactly 4 attributes. As a result the policy $P_{10} = T_2(a_1, a_2, a_3)$ could not be implemented.

Because we are interested in enabling the creation of highly expressive policies, we discuss three separate approaches which circumvent the fixed n, k requirement for constructions without random oracles.[2] The first two approaches were initially introduced by Sahai and Waters [35]. To better understand the difference between these three solutions it is helpful to introduce the following notation. Let (k_i, n_i) denote a valid pairing of k, n for a particular system.

The first solution is to provide all principals in a system with "default attributes", which act as placeholders and are devoid of semantic meaning (i.e. they are given to all users regardless of their attributes). The purpose of these attributes is to enable objects to contain any threshold operator, $T_{k'}$ such that $1 \leqslant k' \leqslant k$. To attain this end, each object must contain the maximum number of possible attributes n. The default attributes can then pad for any of the required n attributes. This scheme extends a system where only (k, n) is valid to a system where any of the following are valid: $(1, n - (k - 1)), (2, n - (k - 2)), \ldots, (k, n)$. Given $k, n = 10$, this method allows a single cryptosystem to express the ten **and** policies between (1, 1) and (10, 10); however, this example cryptosystem could not express any system in which k, n were not equal.

The second solution entails creating n separate cryptosystems, each with a different value of k, enabling policy to use any of the following: $(1, n), (2, n), \ldots, (n, n)$. Similar to the previous approach, this scheme extends policy expressiveness, but requires a large number of systems to express a diverse set of policies.

Our approach is a hybrid of the above two techniques. Specifically, this approach enables any policy that is expressed with at most n attributes. More exactly, $\forall n_j \leqslant n$ and $\forall k_i \leqslant n_j, (k_i, n_j)$ are all valid pairings of k and n. This scheme is implemented by creating n separate cryptosystems as described in the second solution. From these cryptosystems, default attributes can be used to attain all the desired policies. This is illustrated in Fig. 5, where $n = 10$. Each of these 10 cryptosystems, denoted as the white circles on the right side of the figure, giving the following (k, n) pairs: (1, 10), (2, 10), ..., (10, 10). The diagonal lines indicate the other pairings of (k, n) that are obtainable by using default attributes from within the 10 cryptosystems. Consider the (9, 10) cryptosystem, in which default attributes allow for the expression of the policies (1, 2), (2, 3), ..., (9, 10). Similar expressiveness is possible for the remaining cryptosystems.

---

[2] While this discussion focuses on circumventing the fixed n, k in the constructions without random oracles, our approaches can be extended to constructions with random oracles, which is only limited by having a fixed k.

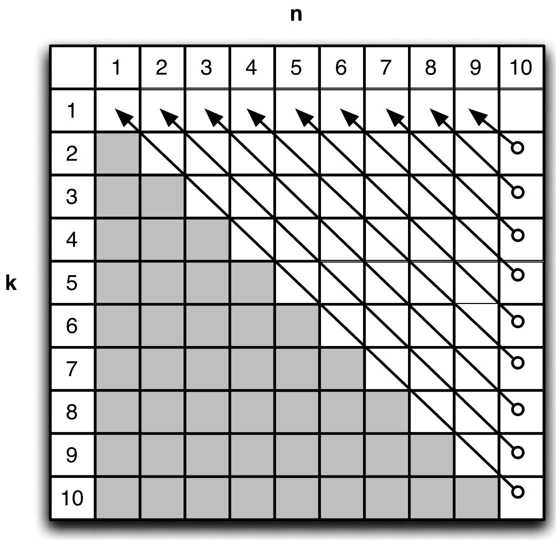

Fig. 5. Example of our method for extending flexibility of ABE. ABE constructions without random oracles can only directly implement a *k-out-of-n* policy where k and n are fixed. In this example we show how any possible pairing of $k \leqslant n$ and n, given that no more than 10 attributes will be present in any ciphertext. To attain this end, 10 separate cryptosystems (denoted by white circles) are implemented. Arrows indicate the use of semantically void default attributes. These attributes can be used on each of the 10 cryptosystems to attain any k, n pairing.

This scheme can easily be extended to meet the needs of the target application. For instance, a system may opt to only create a subset of the possible cryptosystems (e.g. the values of k for powers of 2 less than or equal to n). Section 6 explores the performance trade-offs associated with using such a sampling. Trade-offs between expressibility, performance and overhead must be carefully considered.

5. Application of policy

The threshold, conjunction and disjunction constructions discussed in Section 4 result in an expressive policy system. In this section we illustrate the use of policy in two separate applications: HIPAA compliant distributed storage systems and social networks.

5.1. Distributed file systems

A *content-addressable* file system enables users to locate files based on attributes or keywords describing their contents. Accordingly, data becomes searchable in a more meaningful fashion than the traditional approach of specifying file paths. To date, most work on content-addressable file systems has focused on automatically generating descriptions of a file's contents [9,19,22]. The use of ABE strengthens the security properties of such systems. Because the access control policy of every object is embedded within it, *the enforcement of policy becomes an inseparable characteristic of the data itself.* This is in direct contrast to most currently available systems, which rely directly upon a trusted host to mediate access and administer policy. As file systems become more distributed in nature and rely upon domains of varying trust to control access, traditional approaches no longer provide adequate guarantees.

Example systems include large multisite research efforts such as the Human Genome Project, which was formed in order to map the sequence of chemical building blocks composing the human genome. While this multinational research effort requires a total of only 3 gigabytes of space to store the genome itself, the space estimated for additional annotations will sufficiently dwarf the initial sequence data in the system [3]. As this and other research projects begin to require petabytes of storage, the ability to securely store such information across multiple sites becomes increasingly critical.

5.1.1. Policy for HIPAA compliant medical systems

We use a Health Insurance Portability and Accountability Act (HIPAA) compliant medical system as an example of a loosely-coupled, content-addressable file system with strict security requirements. HIPAA was designed to clearly enumerate the security requirements and provide information flow control for electronically stored medical information such that patient privacy is maintained [42]. While the system below is by far not a comprehensive example of HIPAA requirements, it demonstrates the ease with which a fully compliant system could be constructed using ABE.

In this example, a patient i's medical information is composed of several fields. Patients define the following privacy policies to best protect each component of their medical information: currently used medications ($P_{i,\mathrm{Med}}$), medical history ($P_{i,\mathrm{Hist}}$), contact information ($P_{i,\mathrm{CI}}$), and insurance information ($P_{i,\mathrm{Ins}}$). A patient's policy describes the attributes that must be possessed by medical personnel in order to access their medical information. As illustrated in Fig. 6, these attributes describe various job functions of medical personnel as well as the health insurance plans they accept.

A patient, Robert Oppenheimer, supplements his limited insurance coverage through the ACME Corporation with a "Medicare D" prescription plan. His policy therefore stipulates that only doctors (*Dr*) and nurses (*Rn*) supporting his insurance plan can view his full medical history, contact information, and a listing of his current medications. Oppenheimer's policy also allows a pharmacist (*Rx*) in his plan to view the medications he is currently taking, so that he/she can ensure no conflicts between prescriptions exist. Further, a pharmacist is allowed access to contact information to notify him when prescriptions have been filled. Oppenheimer's policy also restricts the access of billing personnel (*Bill*) to his insurance and contact information such that charges can be filed with his insurance providers without danger of exposing private information. Lastly, without revealing his contact information, Oppenheimer

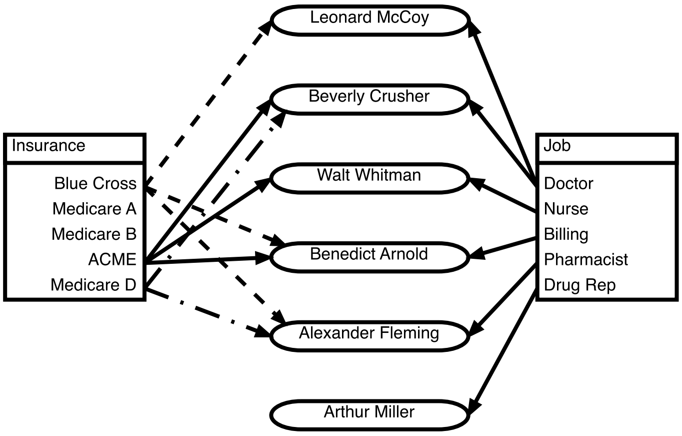

Fig. 6. Mapping of attributes to principals in HIPAA compliant medical system.

allows the list of medications he is currently using to be made available to pharmaceutical representatives (*Rep*) analyzing the combination of drugs with which their products are prescribed in concert. As described above, Oppenheimer's policies are represented as follows:

$$\begin{aligned}
P_{O,\mathrm{Hist}} &= T_1(Dr, Rn) \wedge T_1(ACME, Medicare\ D),\\
P_{O,\mathrm{CI}} &= T_1(Dr, Rn, Bill, Rx) \wedge T_1(ACME, Medicare\ D),\\
P_{O,\mathrm{Ins}} &= T_1(Bill) \wedge T_1(ACME, Medicare\ D),\\
P_{O,\mathrm{Med}} &= \bigl(T_1(Dr, Rn, Rx) \wedge T_1(ACME, Medicare\ D)\bigr) \vee T_1(Rep).
\end{aligned}$$

From the above policies, only Dr. Crusher and Nurse Whitman can access his medical history. Dr. Crusher, Nurse Whitman, Billing Secretary Arnold, and Pharmacist Fleming can access his contact information. Billing Specialist Arnold can also access Oppenheimer's insurance information. Dr. Crusher, Nurse Whitman, Pharmacist Fleming and Pharmaceutical Representative Miller are able to determine Oppenheimer's current regime of medication.

5.2. *Social networks and online communities*

Social networks, such as orkut, Facebook and Friendster [1], are an online application which enable users to find other users with similar interests. To use these applications, users must reveal large quantities of personal information (e.g. name, age, address, personal interests, sexuality, etc.) into the public domain. Groups of people sharing similar attributes and friends are then automatically linked to each other. Currently, such systems provide only weak privacy guarantees; network membership allows access to the wealth of user information. Accordingly, user data can readily be mined and abused by undesirable parties.

ABE-based systems are well suited to provide user controlled-privacy, as users in these communities are already characterized by their attributes. In Friendster, for example, a user with the attribute "Penn State University Alumnus" is automatically enrolled in a group of the same name. Accordingly, the creation of "white-lists" for communication immediately becomes possible without requiring enumeration of all user identities. Constructing a social network using ABE also provides scalability. Current social networks require a trusted central server to store all profile information and enforce policy. Because ABE-based systems do not require a trusted storage system, profile information could be stored on untrusted servers, significantly decreasing the traffic and storage requirements incurred by a system. Further, in an ABE-based system, objects are embedded with policy, enabling distributed enforcement.

5.2.1. Policy in a social network

We now demonstrate policy in a social network through an application where the principals are users of an online dating service. Each user dictates their own policy in order to restrict access to their personal information.

Figure 7 illustrates a sample network. A principal's policy can be viewed as a description of attributes they find desirable in other principals. Possession of the attributes described in the policy is therefore a prerequisite to being able to access another principal's personal information. We begin with a relatively simple policy. Van Buren is only interested in meeting women with black hair, medium wealth, and

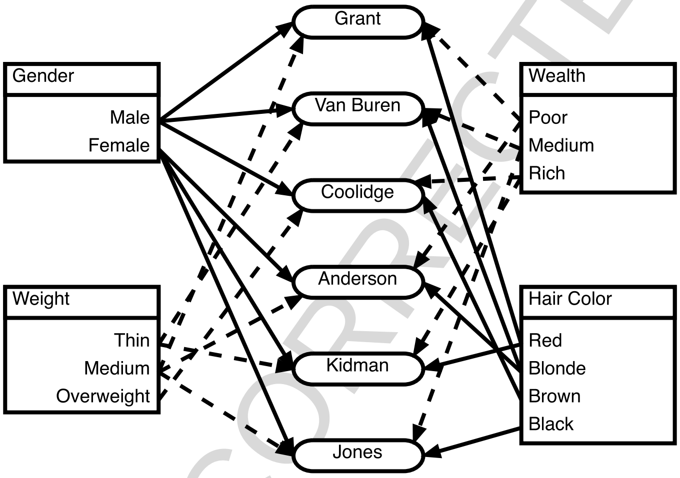

Fig. 7. Mapping of attributes to principals in social network application.

medium weight. His policy is represented as:

$$P_V = T_4(\mathit{Female} \wedge \mathit{Black\ Hair} \wedge \mathit{Med\ Wealth} \wedge \mathit{Med\ Weight}).$$

Of the above principals, only Jones can access Van Buren's profile. This policy, which is equivalent to policy P_2 as depicted in Fig. 2, can be expressed using a single threshold primitive T_4. It is therefore possible to directly implement P_V with a single ABE encryption. Accordingly, data encrypted under this policy will be resistant to collusion.

Grant's policy, whereby only blond or red haired women can access his profile information, is represented as:

$$P_G = T_1(\mathit{Female}) \wedge (T_1(\mathit{Blonde}) \vee T_1(\mathit{Red})).$$

As such, only Anderson or Kidman can access his information. Notice that Grant's policy is equivalent to policy P_6 in Fig. 4 and therefore cannot be implemented using a single threshold operator. Accordingly, P_G is less resistant to collusion than P_V.

Lastly, Anderson is interested in hearing from men who possess at least two of the following attributes: red hair, medium weight, overweight, or medium wealth. Her policy can be represented as:

$$P_A = T_1(\mathit{Male}) \wedge T_2(\mathit{Red}, \mathit{Med\ Weight}, \mathit{Overweight}, \mathit{Med\ Wealth}).$$

Given Anderson's policy Coolidge and Grant can access her information. Notice, however, that a principal's policy is not necessarily symmetric. For instance, of these two, only Grant has a policy that would allow Anderson to contact him. Note also that some attributes are inherently mutually exclusive. For example, none of Anderson's suitors could logically be both medium weight and overweight. Enforcement of an intrinsic dichotomy between attributes is the job of the authority assigning attributes; however, it is possible to write more robust policies such that even the accidental assignment of such attributes would not allow a client to access data (e.g., using a separate 1-out-of-2 threshold encryption for the weight characteristic).

6. System evaluation

The policies discussed in the previous section illustrate the potential expressibility of ABE-based systems. In this section, we characterize the performance of systems providing such functionality. We begin by exploring the cost of the base cryptographic constructions. We then determine the cost of implementing a selection of the previously defined policies. We finish by comparing the performance of an ABE-based system to a comparable system implemented with RSA cryptographic primitives.

As demonstrated by numerous others (e.g. [14]), the selection of cryptographic parameters can have a drastic impact on system performance. In this section, we characterize the parameter space by profiling the performance of attribute systems under different input parameters. Such analysis is necessary to optimize the system for a particular application or environment. All experiments were carried out on a 2.0 GHz Apple Xserve G5 with 4 GB memory running Mac OS X Server 10.3.9. All disk operations were performed on a 1.82 TB RAID 5 disk array. All results are calculated from an average of 500 iterations of the measured operation. The performance of these primitives was noted to improve when similar tests were executed on desktop class machines using x86-based processors [41].

We have implemented an ABE library upon which secure attribute systems can be constructed. This C library contains approximately 5,200 lines of code and has been tested on Solaris, OS X and Linux platforms. To our knowledge, we are the first to implement, measure, and characterize the theoretical mechanisms of attribute-based encryption. Accordingly, we explore a wide range of potential inputs and settings system architects should consider when designing new secure attribute-based environments. For instance, systems using our API can choose between the two most studied elliptic curve groups providing bilinear maps: *supersingular elliptic curves* (SS), which enable fast cryptographic pairing operations [30], and *MNT elliptic curves*, which are used to obtain small ciphertext sizes [31]. We use the Pairing-Based Crypto library [27] for the underlying implementation of these groups and OpenSSL [2] for providing a supporting Key Encapsulation Mechanism (KEM) [40].

The following analysis measures the four central functions of the attribute system as defined in Section 2: `Setup_System`, `Key_Generation`, `Encryption`, and `Decryption`. For reference, Table 1 provides an outline of the base cryptographic operations for each of the base operations. See Appendix A for greater detail on the use of these functions and the design of our attribute system API. All source code and documentation are available at: http://siis.cse.psu.edu/attribute.html.

These experiments indicate several important properties of the parameter space. Firstly, MNT is faster than SS for encryption whereas the opposite is true for decryption. Secondly, encryption costs are significantly improved by the use of random oracles. Hence, the curve selected should be a reflection of the relative number of encryptions and decryptions performed in the system, as well as the capabilities of the

Table 1

Base cryptographic operations for the major attribute functions

	Random oracle			No random oracle		
Operation	Hashes	Expon.	Pairings	Hashes	Expon.	Pairings
`System_Setup`		1	1		1	1
`Key_Generation` (x attributes)	x	$3x$			$3x + (n * x)$	
`Encryption` (y attributes)	y	$2 + y$			$2 + y + (n * y)$	
`Decryption` (threshold k)		$2k$	$k + 1$		$2k$	$k + 1$

encryptor and the intended recipients. Lastly, the ability to express complex policies with ABE allows for practical use of attribute-based systems.

6.1. Experimental results

The first set of experiments measure the degree to which different system parameters affect performance: we vary the number of attributes, length of data, elliptic curve and initialization of randomness parameters. We then perform an R-squared or coefficient of determination analysis over the measured results. This technique identifies the portion of observed variance in one variable that is directly attributable to a second. On a scale from zero to one, numbers closer to one represent a significant correlation between variables. For precision, we also include measurements for two additional subfunctions: `Initialize_Randomness` preloads random bytes from the local entropy pool and `New_Attribute` allocates a new attribute to a principal.[3] The results of these tests reveal that the number of attributes followed by the elliptic curve used are the dominant factors, as shown in Tables 2 and 3 (bold font highlights significant value). Lastly, to characterize the growth of execution time against the number of attributes, we run a regression analysis for the worst case and present our findings in the standard linear form, i.e., $y = mx + b$.

Table 2

Table of r^2 values (no random oracles)

	# Attributes	Data length	Curve type	Rand init
`Initialize_Randomness`	2.083E−5	1.043E−5	1.440E−6	0.9721
`System_Setup`	**0.8052**	1.321E−4	0.0138	2.355E−6
`New_Attribute`	0.0442	1.499E−4	6.959E−4	2.282E−5
`Key_Generation`	**0.8297**	2.480E−4	0.0369	1.158E−9
`Encryption`	**0.7134**	2.120E−4	**0.0692**	6.470E−9
`Decryption`	**0.5355**	1.733E−4	**0.2222**	5.466E−9

Table 3

Table of r^2 values (random oracles)

	# Attributes	Data length	Curve type	Rand init
`Initialize_Randomness`	1.167E−5	1.254E−5	3.363E−7	0.9721
`System_Setup`	**0.7908**	8.915E−7	0.0176	3.827E−7
`New_Attribute`	0.0394	1.014E−4	8.343E−5	3.719E−6
`Key_Generation`	**0.9997**	9.792E−8	1.551E−4	3.916E−8
`Encryption`	**0.4781**	3.167E−7	**0.1993**	5.456E−8
`Decryption`	**0.5608**	4.543E−9	**0.2041**	4.078E−9

---

[3] In all tests in this section, `Initialize_Randomness` is included in measurements of `System_Setup`, and `New_Attribute` is included by `Key_Generation`.

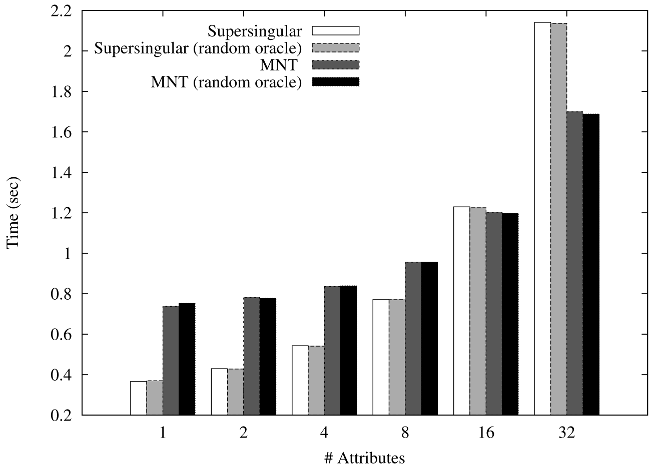

Fig. 8. `System_Setup`: As the number of attributes grows, MNT curves become more efficient.

Figure 8 shows the cost of `System_Setup` as a function of the number of attributes. Systems using a SS curve without random oracles average 0.366 seconds ($\sigma = 0.049$, 95% CI $= \pm 0.004$) and 2.141 seconds ($\sigma = 0.133$, 95% CI $= \pm 0.011$) for 1 and 32 attributes, respectively. A system using the MNT elliptic curve without random oracles averages between 0.737 seconds ($\sigma = 0.202$, 95% CI $= \pm 0.017$) and 1.699 seconds ($\sigma = 0.284$, 95% CI $= \pm 0.025$) for the same range. Execution time for both curves scales linearly in the number of attributes (SS w/o random oracles: $y = 0.572x + 0.3126; r^2 = 0.9999$). Random oracles have no role in system setup, and hence have no bearing on performance. System setup therefore poses no significant computational burden in real systems.

Figure 9 illustrates the cost of key generation, which is consistently cheaper for MNT curves. For a system using 32 attributes without random oracles, MNT curves require an average of 12.355 seconds ($\sigma = 0.035$, 95% CI $= \pm 0.003$) to generate a user key, compared to 25.05 seconds ($\sigma = 0.052$, 95% CI $= \pm 0.004$) for SS curves. Random oracle constructions are significantly faster – systems built on SS and MNT curves perform similarly at all numbers of attributes, e.g., at 5.051 ($\sigma = 0.017$, 95% CI $= \pm 0.002$) and 4.927 ($\sigma = 0.017$, 95% CI $= \pm 0.002$) seconds, respectively for 32 attributes. Execution time scales linearly for both curves with and without the use of random oracles (SS w/o random oracles: $y = 0.8003x - 2.37$; $r^2 = 0.9584$). Note that key generation for each user is performed infrequently (likely once). If the user community is fairly static, such costs will be amortized by operations on data. Conversely, in environments where users may join frequently, it behooves the administrator to select parameter choices that minimize these costs, e.g., MNT elliptic curves using random oracles.

As shown in Fig. 10 for both SS and MNT elliptic curves, the construction without random oracles requires an average of 11.213 ($\sigma = 0.031$, 95% CI $= \pm 0.002$) and

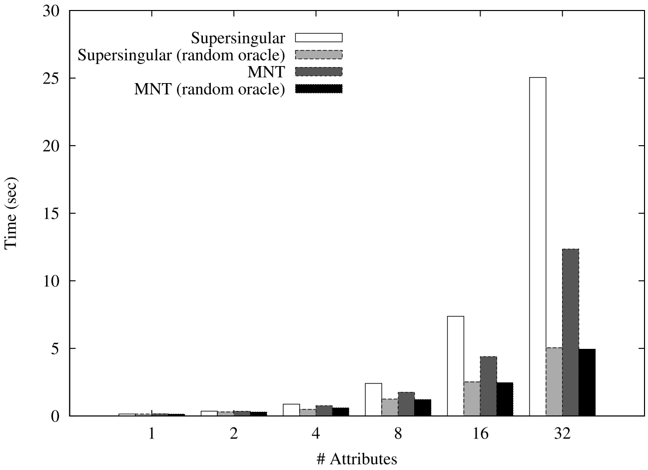

Fig. 9. `Key_Generation`: Performance becomes nearly identical for systems using either type of curve with random oracles.

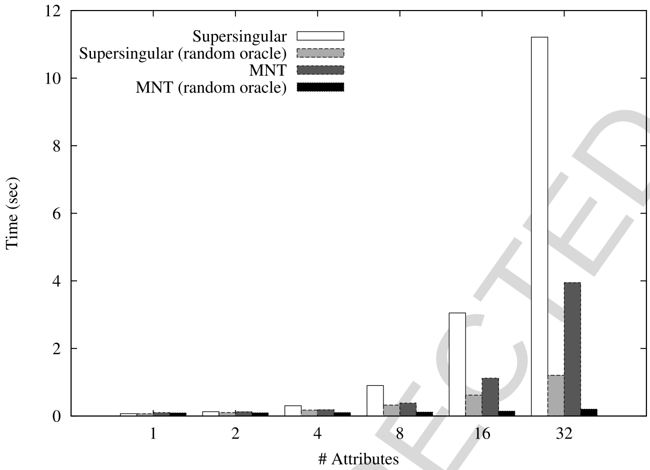

Fig. 10. `Encryption`: SS constructions are significantly slower than MNT constructions.

3.946 ($\sigma = 0.017$, 95% CI $= \pm 0.002$) seconds to encrypt data using 32 attributes. Systems implementing the construction with random oracles experience dramatically improved encryption performance, i.e., 1.207 ($\sigma = 0.009$, 95% CI $= \pm 0.0001$) and 0.204 ($\sigma = 0.006$, 95% CI $= \pm 0.001$) seconds, respectively. Here, MNT elliptic curves are approximately 65–85% faster than their SS counter-parts (in constructions with and without random oracles, respectively). Systems using MNT curves with random oracles are in fact 98% faster those using SS curves without random oracles.

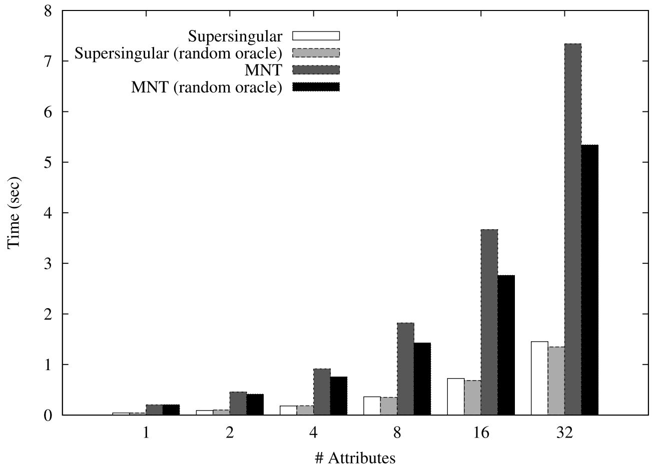

Fig. 11. `Decryption`: SS constructions are significantly faster than MNT constructions.

Both systems scale linearly in the number of attributes with and without random oracles (SS w/o random oracles: $y = 0.3590x - 1.148$; $r^2 = 0.9487$). Conversely, as illustrated in Fig. 11, a system of 32 attributes with and without random oracles exhibits a decryption time of 1.452 ($\sigma = 0.009$, 95% CI $= \pm 0.003$) and 1.348 ($\sigma = 0.044$, 95% CI $= \pm 0.0001$) for an SS construction and 7.341 ($\sigma = 0.029$, 95% CI $= \pm 0.073$) and 5.342 ($\sigma = 0.841$, 95% CI $= \pm 0.003$) seconds in MNT, respectively. Execution time for both systems scales linearly in the number of attributes with and without random oracles (MNT w/o random oracles: $y = 0.2298x - 0.103; r^2 = 0.9999$). Note the systems using SS curves experience approximately 80% faster performance than their MNT counterparts.

Lastly, we compare the performance of such a system against traditional offline cryptographic techniques.[4] From OpenSSL's benchmarking tool [2], the platform used for ABE benchmarking is capable of performing RSA public key encryption in 0.0003 and 0.00097 seconds for 1024 and 2048-bit keys, respectively. To offer similar semantic expressiveness and prevent the need for $2^N - 1$ keys (there are $2^N - 1$ nonempty subsets in a set of size N), we assume that each attribute in an ABE system has a corresponding RSA key pair. For simple policies, encryption under a single attribute/key is 300 and 93 times faster under RSA (0.0003 and 0.00097 vs 0.09 seconds). ABE's thresholding primitive, however, allows much more efficient execution. For example, a policy requiring a threshold of 2 of 32 attributes has nearly identical execution times (0.1488 vs 0.2043 seconds) for both RSA-1024 and ABE

---

[4] Our analysis is designed to express the performance disparity of semantically meaningful cryptography against more traditional offline schemes. Online techniques such as idemix [10] can greatly improve performance over the offline use of RSA, assuming the tradeoffs of online systems discussed in Section 8.2.

with MNT curves and random oracles. RSA-2048 requires approximately 0.5 seconds to achieve the same ends. A system requiring 16 of 32 attributes would also require 0.2043 seconds for an ABE system; however, the equivalent RSA systems would require approximately 33.4 and 107.97 days, over 46.6 million times slower, to perform the necessary $\frac{32}{16}$ encryptions. ABE's inherent expressibility makes it a practical means of constructing real attribute systems.

6.1.1. HIPAA system policy analysis

We now determine the cost incurred for implementing expressive policies in a proposed HIPAA system. Encrypting with the policy $P_{O,\mathrm{CI}}$ from Section 5.1.1 requires an initial encryption of the principal Oppenheimer's contact information using $T_1(\mathit{ACME}, \mathit{Medicare\ D})$, a process requiring E_2 time to complete. This ciphertext object is then independently re-encrypted with each of the following attributes: *Dr*, *Rn*, *Billing*, *Rx*. Each of these encryptions requires 1 attribute, and thus takes E_1 time to complete. This policy could alternatively be implemented using two encryptions if the **or** construction is replaced with $T_1(\mathit{Dr}, \mathit{Rn}, \mathit{Bill}, \mathit{Rx})$. Table 4 shows the timing values for this optimized policy, noted as $P_{O,\mathrm{CI}} = E(E(\mathit{CI}, T_1(\mathit{ACME}, \mathit{Medicare\ D})), T_1(\mathit{Dr}, \mathit{Rn}, \mathit{Bill}, \mathit{Rx}))$.

Decrypting data encrypted under $P_{O,\mathrm{CI}}$ requires two operations. The first decryption occurs with any of the following attributes: *Dr*, *Rn*, *Billing*, *Rx*. The second decryption, which enables recovery of the plaintext, requires decryption of $T_1(\mathit{ACME}, \mathit{Medicare\ D})$. Table 4 shows average execution time.

6.1.2. Social network analysis

We now examine the cost of expressing policy as described in Section 5.2.1 for a hypothetical social network application. Consider the time required to encrypt a message under Grant's policy, P_G. Grant's information I_G is first independently encrypted under $T_1(\mathit{Red\ Hair})$ and $T_1(\mathit{Blonde})$. Both values, noted as $I'_G = E(I_G, T_1(\mathit{Red\ Hair}))$ and $I''_G = E(I_G, T_1(\mathit{Blonde}))$ respectively, are then encrypted under $T_1(\mathit{Female})$, yielding $E(I'_G, T_1(\mathit{Female}))$ and $E(I''_G, T_1(\mathit{Female}))$. Note that the total number of encryptions can be halved if the **or** semantic is equivalently implemented as $T_1(\mathit{Red\ Hair,\ Blonde})$. The total time to encrypt P_G is given by $E(P_G)$.

Table 4

Average performance (s) for $P_{O,\mathrm{CI}}$

	No rand oracles		Rand oracles	
	SS	*MNT*	*SS*	*MNT*
E_2	0.13	0.12	0.10	0.10
E_4	0.31	0.18	0.18	0.10
D_2	0.09	0.46	0.10	0.42
D_4	0.18	0.91	0.18	0.75
$E(P_{O,\mathrm{CI}})$	0.44	0.30	0.28	0.20
$D(P_{O,\mathrm{CI}})$	0.27	1.37	0.28	1.17

Table 5

Average performance (s) for P_G

	No rand oracles		Rand oracles	
	SS	*MNT*	*SS*	*MNT*
E_1	0.07	0.10	0.07	0.09
D_1	0.04	0.20	0.04	0.20
$E(P_G)$	0.28	0.40	0.28	0.36
$D(P_G)$	0.08	0.40	0.08	0.40

Table 5 details the time required to perform the unoptimized operations required to formulate P_G. These values represent the encryption and decryption operations for SS and MNT elliptic curves with and without random oracles.

In the case of decryption for P_G, two decryptions are required. The decrypting party initially performs two decryptions with a_3. From this, only a_1 or a_2 must be decrypted in order to recover the original plaintext. The total time to decrypt P_G is given by $D(P_G)$.

6.2. *Ciphertext size and user key length*

Ciphertext size and key length are important to some classes of applications, e.g., in high traffic volume or low bandwidth networks or on resource poor devices. Here, we briefly detail the size of ciphertexts and the size of a user's private key. Specifically, we quantify ciphertext length and user key length as described in Appendix B. Because the focus of this paper is on attribute systems and ABE, we do not include structured data framing or data encrypted with symmetric cryptography in our treatment of ciphertext size.

We shall first discuss a discrepancy between MNT curves and SS curves that is necessary to understand our analysis. Recall that the Sahai–Waters construction makes use of a bilinear group $\mathbb{G}$ to perform bilinear map operations: $e : \mathbb{G}\times\mathbb{G} \rightarrow \mathbb{G}_T$. This type of bilinear map is said have symmetric groups. A bilinear map that is asymmetric has the following form: $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T, \mathbb{G}_1 \neq \mathbb{G}_2$. SS curves are characterized by having symmetric bilinear groups. Both $\mathbb{G}$ and $\mathbb{G}_T$ require 512 bits to be represented. MNT curves are characterized by having asymmetric bilinear groups. $\mathbb{G}_1$ is represented with 170 bits while $\mathbb{G}_2, \mathbb{G}_T$ are represented with 510 bits.

Each attribute i possessed by a principal corresponds to two private key components d_i and D_i. For SS curves both of these components are members of $\mathbb{G}$. For MNT curves $D_i \in \mathbb{G}_1, d_i \in \mathbb{G}_2$. This yields (for a private key with n attributes):

$$\textit{Supersingular KeySize}(n) = 2 \cdot n \cdot 512 \text{ bits}, \tag{10}$$

$$\textit{MNT KeySize}(n) = (170 + 510) \cdot n \text{ bits}. \tag{11}$$

A ciphertext C scales with the number of attributes it contains as follows. A ciphertext with n attributes is composed of C', C'' and n elements C_i. For SS curves

$C'', C_i \in \mathbb{G}$ and $C' \in \mathbb{G}_T$. For MNT curves $C'' \in \mathbb{G}_2$, $C' \in \mathbb{G}_T$, and $C_i \in \mathbb{G}_1$. This yields (for a n attribute ciphertext):

$$\mathit{Supersingular\ CTLen}(n) = (n + 2) \cdot 512 \text{ bits}, \tag{12}$$

$$\mathit{MNT\ CTLen}(n) = 2 \cdot 510 + 170n \text{ bits}. \tag{13}$$

7. Open problems in policy expression

In the previous sections, we characterized the flexibility and efficiency with which semantically deep policy could be expressed using attribute-based encryption. However, significant challenges remain in a number of areas. In this section, we specifically focus on a number of the open problems with regards to the expression of policy in these systems.

7.1. Non-monotonic policy

One of the primary thrusts of this work has been to extend policy expressiveness beyond the fixed *k of n* supported by the underlying cryptographic constructions. As demonstrated in our previous results [34], a combination of default attributes and multiple cryptosystems can be used to express any monotonic Boolean policy (i.e. a policy constructed by using **and**s and **or**s to arbitrarily combine attributes) containing at most n attributes. Such policies represent a significant expansion in flexibility over operations provided by the cryptography itself. However, it may be preferable in some cases to be able to express non-monotonic policies. Specifically, the existence of a **not** ($\neg$) logical primitive would be a powerful addition for the declaration of policy. The ability to apply negation could be especially valuable in situations where the declaration of those for whom access should be denied is simpler than the enumeration of positive access rights.

The presence of a negation primitive also offers the opportunity for improved performance for systems and specific policies in which user attributes are mutually exclusive. Building from an example in Section 5, we examine patient Openheimer's policy $P_{O,\mathrm{CI}}$ regarding the protection of his contact information. Specifically, Openheimer allows for physicians, nurses, members of the billing department and pharmacists to contact him: $T_1(\mathit{Dr}, \mathit{Rn}, \mathit{Bill}, \mathit{Rx})$. This policy can be seen as indirectly expressing $\neg \mathit{Rep}$, by including the attributes of all parties except that of *Rep* in the encryption of contact information. It is the specific embodiment of this policy that allows for the system to potentially realize performance gains. Specifically, the monotonic policy expressed above could be achieved using a cryptosystem in which $n = 4$. The non-monotonic representation of the same expression could be accomplished in a cryptosystem in which $n = 1$. Accordingly, as detailed in Section 6, the number of attributes required to express monotonic policy directly impacts overall system execution time.

Of course, not all policies and systems allow for implicit negation via exclusion. Plainly stated, not all attributes are necessarily mutually exclusive. From our social network example in Section 5, no combination of positive attributes would express “non-smoker” or “not interested in silent movies”. Thus, the specific representation of such characteristics must be explored.

The most logical method of attaining non-monotonicity would be to design a **not** primitive analogous to the **and** and **or** primitives introduced in Section 4. However, such a primitive would be ineffective in a distributed setting where recipients perform decryption. This is largely due to the fact that a potential recipient could erase or hide their A_3 attribute in order to circumvent the **not** primitive. Unless the system required decryption to always use all attributes [25], users could not be coerced into demonstrating they did not possess a specific attribute. While useful in the expression of policy, it should be noted that the introduction of a **not** primitive makes compliance checking, at worst, undecidable [7]. Such tradeoffs must therefore be carefully considered when building a system.

To address the shortcomings introduced by the **not**, one can embed negation within the semantics of the actual attributes. In this way, the cryptography would be blind to the fact that there are negative attributes. For instance, A_4 could be the attribute $\neg A_3$. We could then express P_N as $A_1 \wedge A_2 \wedge A_4$. To decrypt an object with this policy requires that the party performing decryption has proven that he/she possesses the attributes A_1 and A_2, and that he/she does not possess the attribute A_3.

A benefit of this scheme is that negative attributes are handled by the cryptosystem in the same way as regular attributes. Whether an attribute is negative or not is indicated by its semantics. Thus, the pair of attributes *male* and ¬*male* are handled identically to attributes that are inherently negatives of each other, such as the attributes *male* and *female*.

The introduction of negative attributes may cause a significant increase in the management duties of both users and the authority. Consider an implementation where every attribute in the system must be generated in tandem with a corresponding negative version. Moreover, each user is required to possess an attribute or its negative for all attributes in the system. Such requirements would make the management of an attribute system with even a moderate number of attributes infeasible. Users would need to carry a potentially massive quantity of keying information. The system would also need to dedicate significant resources towards the generation of additional attributes. Lastly, users would have to be immediately notified when new attributes were added to the system, which may be difficult in disconnected environments.

To make the management of keying information feasible, we advocate the use of a system built on *lazy negative attributes*. Using lazy negative attributes, the addition of an attribute to the system does not mandate addition of its negative unless a user or policy explicitly requires its presence. This enables the system to only have to maintain the attributes that its users are actually using.

This model can be extended to include the specific attributes possessed by a user. During the process of verification with the authority, a user may select to prove only a subset of their total attributes. Such decisions may be made for reasons ranging from privacy [4] to perceived usefulness. Consider a user in our social network example that could prove that they posses the attribute "enjoys long walks on the beach". This user will not possess this attribute until after they specifically request it from the authority. If they do not find this attribute to be useful, then they are not required to demonstrate that they possess it. Such a system not only allows the needs and desires of the users to dictate the attributes they posses, but is also more realistic (i.e., it may be difficult to force a user to prove a private preference).

7.2. Hierarchical systems

In many situations, attributes are semantically equivalent to the concept of roles. From our HIPAA example in Section 5, the principal Leonard McCoy embodies the roles of "Doctor" and "Accepts Blue Cross Insurance" simply by possessing the associated attributes. In order to efficiently manage a large attribute-based system, it may be beneficial to develop a role hierarchy [17,36]. As is shown in Fig. 12, we create such relationships using an expanded set of roles in a HIPAA system. Under such a hierarchy, Leonard McCoy (and any other principal possessing the attribute associated with "Doctor") would also have the authority to act as a "Care Provider", "Medical" Official and member of the "HIPAA System". Given a medical emergency in which triage is required, Dr. McCoy could therefore naturally assume the role of "Care Provider" to assist with patient admission. On the other side of the organizational structure, the representative of the billing department Benedict Arnold, who is also a certified public accountant (CPA), may act in the roles of "Accounts

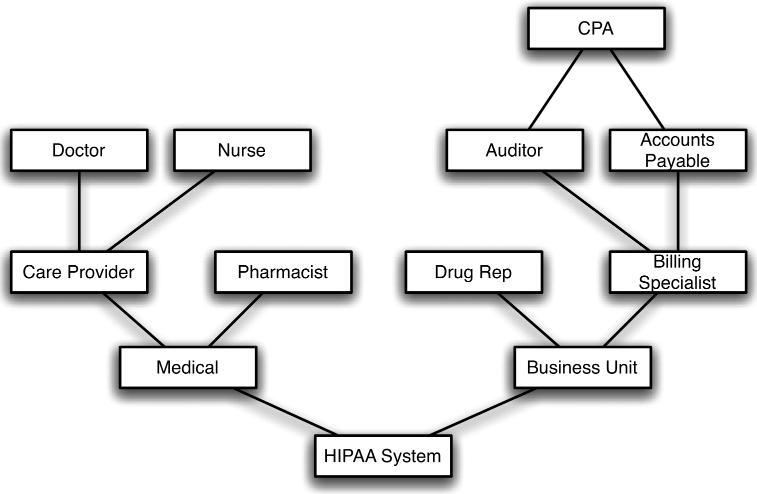

Fig. 12. A sample hierarchy of the roles found in a HIPAA system.

Payable" Official, "Billing Specialist" and a member of both the "Business Unit" and the "HIPAA System". Accordingly, Specialist Arnold can both send bills to customers as an "Accounts Payable" Official and order paper supplies for the office as a member of the "Business Unit".

The realization of such an expressive system can be achieved using the current constructions, but requires a significant amount of overhead. Specifically, under the Sahai–Waters construction [35] used in this work, there is no means of expressing the inheritance of privilege between two attributes. It is therefore not possible for a "Doctor" principal to directly apply this attribute to access messages encrypted under "Care Provider". One approach to addressing this issue is through the use of extensively enumerated policy constructions. Assuming individual attributes exist for each of the roles in the hierarchy, an announcement made to all members of the medical community could be encrypted as follows:

$$P_{Announcement} = T_1(Medical, Care\ Provider, Pharmacist, Doctor, Nurse).$$

Reflecting such access in policy actually addresses a large number of problems. For example, because the attributes "Auditor" and "Accounts Payable" Official are mutually exclusive, the enumeration of all roles allowed to view a message allows for principals in such roles to be separated as necessary. Unfortunately, operations on such constructions may become extremely expensive as policy expressions become arbitrarily long and complex. Policy could only be expressed in a cryptosystem in which all attributes were used in the encryption process. Alternatively, principals fulfilling specific roles could receive the set of attributes representing their specific branch of the hierarchy. Such an approach may also be inefficient due to management and distribution issues. For example, while RBAC systems assume that the tasks assigned to roles are relatively static, the principals assigned to those roles can change with arbitrary frequency. Accordingly, the revocation issues discussed in Section 6 become increasingly pertinent. Techniques from hierarchical identity-based cryptography (HIBE) [18,20,24] or advances in the cryptographic primitives [6] may help to alleviate the policy and management complexity issues discussed above. We examine such issues in the next section.

8. Management issues in ABE systems

While systems built on a foundation of ABE show great potential to support a wide range of applications, a number of crucial issues must be carefully considered before deployment. In this section, we address some of the practical issues relevant to constructing ABE-based systems. For completeness, an in-depth discussion of the implementation and the associated parameters is provided in Appendix A.

8.1. Attribute revocation issues

Revocation of users and keys in systems is a well studied but nontrivial problem [29]. Revocation is even more difficult in attribute systems, given that each attribute is conceivably possessed by multiple different users, whereas public/private key pairs are uniquely associated with a single principal. While an in-depth discussion of revocation is out of the scope of this paper, we give a brief overview of one method by which revocation could be implemented.

One revocation technique would require each attribute to contain a time frame within which it is valid. For instance, the attribute "*Staff Member-December 31st 2006*" denotes that the usefulness of the current attribute expires at the end of 2006. Affixing temporal information to each attribute necessitates the system administrator periodically releasing the latest version of attributes and periodically reissue user keying information. Removal of an attribute from this system would be accomplished by the administrator not releasing the latest version of the attribute. Similarly, revoking an attribute from an individual requires the administrator to withdraw the updated attribute in the user's private key, making the use of short-term keys expensive. There are significant trade-offs between the load placed upon the administrator and the amount of time that can elapse before an attribute/user can be purged. We therefore leave more efficient solutions to future work.

Note that implementation details of ABE systems simplify some aspects of revocation. As discussed in Section 6, because both the Sahai–Waters and Random Oracles constructions are used as KEM, all data are encrypted using a symmetric key. Because this symmetric key is the object directly encrypted using the ABE constructions, the cost associated with re-encryption is largely bounded by this computation as the symmetric encryption of the data file is likely to be very fast.[5]

8.2. Online vs offline systems

One of the advantages of ABE-based systems is their ability to operate offline. In the simple case, after having received the appropriate public parameters and attributes from an authority, a user need not involve that authority in any future data accesses. However, the changing needs of users and the system may require further interaction between these two parties. Accordingly, we briefly discuss the four possible interactions between users and the authority and compare them against such interactions in traditional online systems where appropriate. A more complete experimental quantification and examination of the costs of these operations is offered for a specific application, broadcast encryption service for massive scale content distribution, in our follow-on work [41].

---

[5] Assuming the file is not very large.

8.2.1. Adding attributes to users

Adding attributes to users is the most straightforward interaction in an ABE-based system. As discussed in Section 2, a user is assigned a set of attributes S and a secret key SK via the execution of the Key-Gen algorithm. This set can easily be extended by having the authority rerun the algorithm over an extended set S'. As shown in Fig. 9, the time required perform this operation is linear in the number of attributes assigned to the user and virtually identical for systems using MNT or Supersingular curves with the random oracle construction.

When compared to more traditional online ACL- and capability-based systems, costs of adding an attribute to a user are similar – both require a small amount of involvement by an administrator. Additionally, both changes can take place essentially immediately. However, the offline ABE case can easily amortize the cost of re-keying due to the lack of interaction with the authority during subsequent operations.

8.2.2. Removing attributes from users

Like adding attributes to a user, removing attributes is achieved by executing the Key-Gen algorithm over a modified set of attributes S'. Because SK' does not include the removed attribute, a user can no longer use SK' to decrypt content. The cost to the authority is again linear in the size of S'.

The comparison against a more traditional ACL-based system is less clear-cut in this scenario. For instance, if the user is believed to be an honest player and has not copied SK, possession of SK' is sufficient. However, if it is believed that the user may have retained a copy of SK, revocation may become necessary. The development of efficient forward-secure schemes, which have been discussed for IBE-based systems [43], may also assist in this process. An ACL-based system, however, will behave more like the former case and prevent additional accesses with the revoked attribute at the cost of checking every access attempt. Capability-based systems may require capabilities to be revoked and/or reissued and are therefore generally closer to the ABE-based system in this scenario.

8.2.3. Adding attributes to policies

Adding attributes to a policy protecting an object is achieved using the Encrypt and Decrypt functions. As described in Section 2, the modified set of attributes S'' and the decrypted object to be encrypted are passed to the Encrypt function, which returns the new ciphertext C. Because anyone with possessing the necessary attributes S' can decrypt and then re-encrypt the object under S'', no interaction with the authority is required. The cost of the operation, as shown in Figs 10 and 11, is O(n) in the number of attributes.

Depending on the nature of the added attribute, however, the impact on the encrypted data can be minimized. For instance, if the attribute added expands the pool of users able to access the object (i.e., n in a *k-of-n* grows), only the symmetric key protecting the object need be re-encrypted. Users currently able to access the object would accordingly experience no period in which they were unable to access the object. For the case in which the attribute is added in order restrict access (i.e., k

in a *k-of-n* grows), both the object and the symmetric key protecting it would need to be re-encrypted. During this period, the object should also become unavailable to ensure that it is not accessed until the new policy is encoded. In the case of a more traditional ACL-based system, such changes could quickly be implemented by an administrator and reflected in the next access attempt. Like the ABE-based system, the changes necessary in a capability-based system would depend on the specific change to policy and the system itself.

8.2.4. Removing attributes from policies

Like the previous case, removing attributes from policies is achieved by re-encrypting the targeted object under a new set of attributes. Accordingly, the cost to the re-encrypting party is $\mathrm{O}(n)$ in the number of attributes. Like the previous case, the authority in the system need not be involved in such operations.

The actions associated with removing attributes are inverted when compared to adding them. For instance, if the removal of an attribute makes an object more accessible (i.e., k in *k-of-n* shrinks), no changes to the encrypted data itself need be made. However, if the removal of an attribute makes an access policy more stringent (i.e., n in *k-of-n* shrinks), both the object and the symmetric key protecting it should be re-encrypted. As before, ACL- and capability-based systems could immediately reflect this change with the interaction of the system administrator.

8.2.5. Summary

The average size of objects in the system, in addition to the expected churn rate of user privileges/attributes, should be carefully considered when selecting between these approaches. Scenarios in which access control rules are regularly in flux and communication to centralized authorities is assured are likely to benefit from more traditional access control and policy enforcement mechanisms. However, in environments where a user's attributes are likely to change infrequently or where connectivity to a central authority can not always be guaranteed, the constructions in the paper are an entirely appropriate foundation to manage access to objects.

8.3. Representations of policy

The performance of an arbitrary cryptosystem alone does not tell an engineer how to build a secure attribute-based system. Rather, the results from Section 6 offer only hints toward the realization of efficient policy. In this section, we show how cryptosystem size and object replication can be varied to meet application-specific performance requirements. We begin with the following high-level policy to illustrate the diversity inherent to implementation:

$$P = (A_0 \vee A_1 \vee A_2).$$

As noted above, the owner of object o creates a policy P allowing anyone in possession of 1-out-of-3 attributes to gain access to o. When performing the encryption, this policy can be correctly embodied as any of the following:

$$\begin{array}{rl} \text{1 object:} & E(o, T_1(A_0, A_1, A_2)), \\ \text{2 objects:} & E(o, T_1(A_0)), E(o, T_1(A_1, A_2)), \\ \text{2 objects:} & E(o, T_1(A_1)), E(o, T_1(A_0, A_2)), \\ \text{2 objects:} & E(o, T_1(A_2)), E(o, T_1(A_0, A_1)), \\ \text{3 objects:} & E(o, T_1(A_0)), E(o, T_1(A_1)), E(o, T_1(A_2)). \end{array} \tag{14}$$

An obvious question arises; Given these options, "*Which variant of the high-level policy executes in the shortest time?*". As n becomes large, the party responsible for implementing the encrypted object will be faced with an unmanageable number of potential variants. The results in the Section 6 answer this question. Because of the linear growth of encryption costs, all of the above constructions exhibit approximately the same running time. However, given the overhead of setup and memory management associated with multiple objects, the use of a single atomic expression is the most efficient for policy expressions containing a small number of attributes. This rule holds stronger as the value of k is increased, as this causes an exponential increase in the number of copies of an object needed to express a policy.

Note that the semantic flexibility of this system allows even efficiently implemented policies be unsatisfiable. For example:

$$P = T_2(\mathit{Male}, \mathit{Female})$$

is an efficient representation of a policy that should render the object it protects as unrecoverable by all but the system authority.[6] Policy must therefore be applied carefully to ensure that the system remains usable.

9. Related work

Securing the sharing of information between groups is a fundamental problem that arises in numerous applications. Such applications include multilevel security, secure multicast, collaborative online communities, and distributed file systems. The fundamental importance of the secure exchange of information has resulted in a wide range of solutions.

Traditional access control mechanisms can be categorized into three groups: mandatory access control (MAC) [15], discretionary access control (DAC) [26,37], and role-based access control (RBAC) [17,36]. In MAC, an administrative mechanism enforces centralized access control on every object. Systems implementing DAC require the owner of an object to dictate policy. Under RBAC, a user's role in an organization inherently dictates their ability to access and manipulate data. Each

---

[6] Assuming that attributes are distributed logically.

role in an RBAC system is associated with a set of permissions required to carry out that role. While these mechanisms are highly effective at controlling access for systems under a single administrative authority, they have been largely unsuccessful at providing the same for unconnected and distributed environments.

ABE can enforce access control policy in such environments because it cryptographically binds objects to their policies. Only users possessing the requisite set of attributes are able to view and/or manipulate data. The ability to make policy portable through cryptography is not new. Several works have attempted to use a public key infrastructure (PKI) [21] or secure group communications mechanisms [11,28] to provide similar access control mechanisms. The difficulty with applying standard cryptographic techniques is they are designed to control access to *single* groups. In real systems, however, users are often members of *multiple* groups. Unique keys must therefore be assigned or negotiated for each of the subgroups for which a user is a member. Such solutions do not scale for complex organizations with significant communication across groups. In contrast, users in ABE-based systems automatically belong to every possible attribute subset group without the need for additional keying.

By using cryptographic mechanisms that are in and of themselves able to express complex policies, ABE-based systems become a highly practical means of ensuring the efficient and secure exchange of information between groups.

10. Conclusion

This paper has presented a novel secure information management architecture and implementation. We extended existing constructions for attribute-based encryption (ABE) and promoted them as a practical systems building block. The needs of complex attribute applications were met via the introduction of a policy system and an associated implementation for its enforcement. We illustrated the infrastructure through the creation and performance evaluation of two applications: a HIPAA compliant distributed file system and a social network. A further empirical study shows that a careful selection of parameters and use of construction optimizations can lead to significant cost savings. These analyses demonstrate that our attribute approach is an attractive solution for securely managing information in large, loosely-coupled, distributed systems.

Appendix A. ABE systems design and issues

ABE API – We have created the ABE API to enable rapid development of systems and applications which use attribute-based cryptography. Our API uses the PBC library [27] to implement our attribute-based cryptography. This C language API has been specifically designed to enable a programmer with no knowledge of ABE to

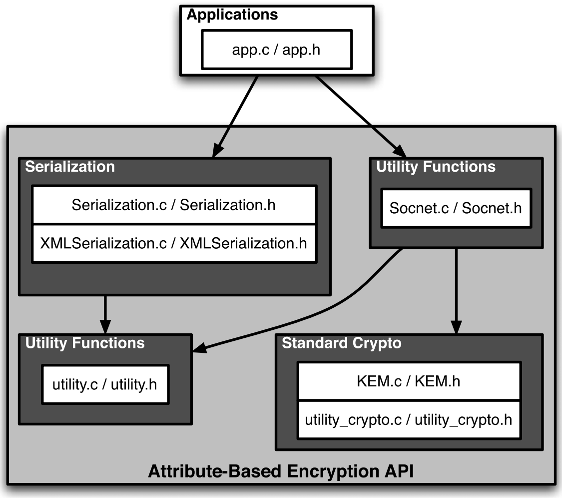

Fig. 13. Components of attribute-based cryptosystem API.

quickly write applications; the complex cryptography inherent to ABE is entirely handled by the API.

For didactic purposes we present the API as four distinct modules: attribute-based cryptography, standard cryptography, serialization and utility functions.

Attribute-based cryptography – The majority of application level code interacts with the API through the attribute-based cryptography component. This module was specifically designed for ease of use, consisting of seven simple functions: `Setup_System`, `Create_User`, `New_Attribute`, `Give_Attribute`, `Key_Generation`, `Encryption` and `Decryption`.

The `Setup_System` function creates and initializes a new attribute-based cryptosystem. Specifically, this instantiates two key structures: `global_params` and `authority_priv`. `global_params` contains global parameters required to perform encryption and decryption operations. `authority_priv` contains the master secret, from which all attribute keys are defined. `authority_priv` must be kept secret in order to ensure the security of the system.

`Setup_System` must be given `pbc_param_file_name`, the name of an XML file defining an elliptic curve from which all of the API's ABE cryptography is formulated. Included with the API are two such parameter files, `a_param.xml` (Supersingular curve) and `c159_param.xml` (MNT curve). Supersingular curves are optimized for fast cryptographic pairings, and MNT curves are optimized to result in small cryptographic group elements.

The nature of ABE cryptography is such that every ciphertext in a given cryptosystem is of a fixed length n. The user can specify what this length is by providing the API with `ct_len`.

To increase the flexibility of the API, `Setup_System` creates several "default" attributes. The default attributes can be included in a ciphertext to take the place of non-default attributes, enabling the user to create ciphertexts with less than `ct_len` attributes.

The `New_Attribute` function is used to add a new attribute, whose name is specified by `att_name`, to the universe of attributes in the system. Upon completion of this function the new attribute's name and the hash of its name can be made publicly available. At this point the new attribute can be used for encryption operations.

The `Create_User` function adds a user named `user_id` to the system. This function instantiates `user_id`, a structure which stores the user's name, the user's default and non-default attribute information, and a polynomial. Each user is given a unique polynomial. Tieing each user's per-attribute keying information to their polynomial prevents users from colluding in order to attain more attributes.

The `Give_Attribute` function is used to give a user a new attribute. Specifically this function is used to update the user's attribute data structures and does **not** generate any keying information. The `Key_Generation` function is used to create a user's keying information based on the attributes that they possess. Keeping key generation separate enables the `Give_Attribute` function to be executed with fewer trust assumptions than is needed to perform the `Key_Generation` function.

The `Encryption` function is used by a user to create a new ciphertext, `ciphertext`. The user specifies, `message`, a string they would like to encrypt and, `uid`, a list of attributes that they would like to encrypt to. The user can encrypt with at most `ct_len` attributes. The API will pad the ciphertext with as many default attributes as is necessary to make the ciphertext contain a total of `ct_len` attributes. A list of the attributes used to perform encryption are included in each ciphertext in order for the party performing decryption to know which attributes are required to decrypt the message.

Encryption is significantly more complicated than the API's function calls would seem to indicate. Specifically, the ABE constructions mandate that a ciphertext's payload must be a group element. To enable ABE to carry non-group element payloads we use the Key Encapsulation Mechanism (KEM). In our API, KEM takes a group element payload and uses SHA-1 to convert it into a HMAC key and an AES key. The AES key is then used to encrypt the user's message.

`Decryption` decrypts a ciphertext encrypted by the `Encryption`. This process begins with the decrypting party verifying that they have the required attributes. The party performing decryption will then use their attributes to decrypt the decrypt the ciphertext in order to obtain the AES and HMAC key. The party will then use the HMAC key to verify the ciphertext. If the ciphertext can be verified, then the AES key will be used to decrypt the actual payload.

Standard cryptography – In addition to the attribute-based cryptography we have also used standard cryptographic tools. The implementation of these tools are contained in the `crypto_utility` and `KEM` code. The `crypto_utility` code implements some of the low level cryptographic operations required by ABE. `KEM` implements all of the operations required to enable ABE to encrypt non-group member payloads.

Serialization – The serialization routines enable the API data structures to be written out to disk for long-term storage. There are two different implementations of this functionality. `Serialization` stores API data structures into byte-encoded files. `XMLSerialization` stores API data structures into XML files. `XMLSerialization` is human readable, has better platform independence, and is more fault tolerant. `Serialization` results in slightly less disk space.

Utility functions – The `utility` functions are a group of functions that increase the ease of programming with the attribute-based cryptography API. Included in the `utility` routines are functions that print API data structures and conversion routines.

Appendix B. Attribute-based encryption

For our system we use a variant of the Sahai–Waters Large Universe system [35] (Section 6) which we now describe.

In this construction we will make use of a bilinear group $\mathbb{G}$ of prime order p. The group will have an efficiently computable bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ that maps two elements from the bilinear group into an element of the "target group". The salient feature of these groups is that if g is a generator of $\mathbb{G}$ then for all $a, b \in \mathbb{Z}_p$ we have that $e(g^a, g^b) = e(g, g)^{ab}$. We refer the reader to the IBE paper of Boneh and Franklin [8] for more details on bilinear groups.

The Sahai–Waters construction works by computing a bilinear map between k components of the ciphertext with corresponding pieces of the private key. The result of these are interpolated using the secret sharing method of Shamir (in the exponent). We first define the following Lagrangian coefficients, which we will use in our construction, as the following function over $\mathbb{Z}_p$:

$$\Delta_{i,S}(X) = \prod_{j \in S, j \neq i} \frac{x - j}{i - j}.$$

Additionally, we will assume all systems will work in some predetermined bilinear group $\mathbb{G}$ of appropriate size.

The cryptosystem follows:

Setup(k): The setup algorithms first chooses a random exponent $y \in \mathbb{Z}_p$ and lets the public parameter be $Y = e(g, g)^y$ and the threshold value k. It keeps the public key and the secret exponent y as the master key.

Key-Gen(S, MK): Let $H : \{0, 1\}^* \rightarrow \mathbb{Z}_p$ be a collision-resistant hash function and let $T : \mathbb{Z}_p \rightarrow \mathbb{G}$ be a function that we will model as a random oracle [5].

First let Γ be the set defined as $\Gamma = \bigcup_{s \in S} H(s)$. The set Γ is essentially the set of the hash of all attributes. (Note that since H is collision-resistant Γ should contain $|S|$ unique elements of $\mathbb{Z}_p$.) Then the authority will choose a new random degree $k-1$ polynomial $q(x)$ over $\mathbb{Z}_p$ such that $q(0) = y$ and for all $i \in \Gamma$ the authority chooses a random r_i. Then for all $i \in \Gamma$ the private keys components are:

$$D_i = g^{q(i)} T(i)^{r_i}, \quad d_i = g^{r_i}.$$

Encrypt(M, S', PK): The encryption algorithm first computes the set $\Gamma' = \bigcup_{s \in S'} H(s)$. Next, it chooses a random exponent $t \in \mathbb{Z}_p$. The ciphertext is output as:

$$C = \left(C' = MY^t, C'' = g^t, \{C_i = T(i)^t\colon\ i \in \Gamma'\}\right).$$

Notice that both the size of the ciphertext and the encryption time grows linearly with the size of the set S.

Decrypt(C, S', S, SK): The decryption algorithm first computes the sets Γ and Γ' as before. If the size of the intersection $|\Gamma \cap \Gamma'| < k$ the algorithm aborts, this will occur if the overlap between the private key attribute set S and the ciphertext set S' is below the threshold k. Otherwise it chooses an arbitrary set U such that $|U| = k$ and $U \subseteq \Gamma \cap \Gamma'$. For each $i \in U$ the decryptor computes a temporary value

$$A_i = \frac{e(D_i, C'')}{e(d_i, C_i)} = \frac{e(g^{q(i)} T(i)^{r_i}, g^t)}{e(g^{r_i}, T(i)^t)} = e(g,g)^{tq(i)}.$$

This computation gives k shares of the polynomial $tq(i)$ in the exponent. Using polynomial interpolation the algorithm recovers the blinding value $e(g,g)^{yt}$ and divides it out by computing:

$$M = C' / \left(A_i^{\Delta_{i,U}(0)}\right) = C' / e(g,g)^{tq(0)} = C' / e(g,g)^{ty} = M.$$

The decryption algorithm interpolates a polynomial in the exponent using Shamir's [38] secret sharing method. However, since a new random polynomial is chosen for each private key, the system is secure against collusion attacks such that different users are unable to combine their separate attributes.

The difference between the construction given here and that of Sahai and Waters is in the computation of the function $T(i)$. In their construction there is a upper bound, n, on the number of attributes that can label a ciphertext which is set at setup. The setup function publishes values $t_1, \ldots, t_n$. The function $T(i)$ is computed as:

$$T(i) = g^{x^i} \prod_{j=1}^{n+1} t_j^{\Delta_{j,N}(i)},$$

where N is the set $\{1, \ldots, n+1\}$.

It is easily seen that the number of exponentiations required to compute $T(i)$ is equal to $n+1$ in the original Sahai and Waters construction. We drastically reduce the computation overhead replacing the computation of T with a hash function as a random oracle. A simple argument shows that the random oracle can be "programmed" such that the simulation in the security proof of Sahai and Waters goes through. We refer the reader to the literature [5,12] for further discussion on the random oracle model. In Section 6 we experimentally compare implementations of the original Sahai and Waters construction with our variant.

References

[1] Friendster, http://www.friendster.com, 2006.

[2] The OpenSSL project, http://www.openssl.org, 2006.

[3] The Human Genome Project, http://www.ornl.gov/sci/techresources/Human_Genome/home.shtml, 2006.

[4] G. Antenise, M. Blanton and J. Kirsch, Secret handshakes with dynamic and fuzzy matching, in: *Proceedings of the ISOC Network & Distributed System Security Symposium (NDSS)*, 2007.

[5] M. Bellare and P. Rogaway, Random oracles are practical: A paradigm for designing efficient protocols, in: *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 1993.

[6] J. Bethencourt, A. Sahai and B. Waters, Ciphertext-policy attribute-based encryption, in: *Proceedings of IEEE Symposium on Security and Privacy*, Oakland, 2007.

[7] M. Blaze, J. Feigenbaum and M. Strauss, Compliance checking in the PolicyMaker trust management system, in: *Financial Cryptography (FC)*, 1998.

[8] D. Boneh and M.K. Franklin, Identity-based encryption from the Weil pairing, in: *Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, Springer-Verlag, 2001, pp. 213–229.

[9] M. Bowman, C. Dharap, M. Baruah, B. Camargo and S. Potti, A file system for information management, in: *Proceedings of the ISMM International Conference on Intelligent Information Management Systems*, March 1994.

[10] J. Camenisch and E. Van Herreweghen, Design and implementation of the idemix anonymous credential system, in: *Proceedings of the ACM Conference on Computer and Communications Security*, 2002.

[11] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor and B. Pinkas, Multicast security: A taxonomy and some efficient constructions, in: *Proceedings of IEEE INFOCOM'99*, 1999.

[12] R. Canetti, O. Goldreich and S. Halevi, The random oracle methodology, revisited (preliminary version), in: *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, 1998, pp. 209–218.

[13] C. Cocks, An identity based encryption scheme based on quadratic residues, in: *IMA International Conference*, 2001, pp. 360–363.

[14] E. Cronin, S. Jamin, T. Malkin and P. McDaniel, On the performance, feasibility, and use of forward secure signatures, in: *Proceedings of 10th ACM Conference on Computer and Communications Security (CCS)*, ACM, Washington, DC, October 2003, pp. 131–144.

[15] D.E. Denning, A lattice model of secure information flow, *Communications of the ACM* **19**(5) (1976), 236–243.

[16] C. Ellison and B. Schneier, Ten risks of PKI: What you're not being told about public key infrastructure, *Computer Security Journal* **16**(1) (2000), 1–7.

[17] D.F. Ferraiolo, R. Sandhu, S. Gavrila, D.R. Kuhn and R. Chandramouli, Proposed NIST standard for role-based access control, *ACM Transactions on Information System Security* **4**(3) (2001), 224–274.

[18] C. Gentry and A. Silverberg, Hierarchical ID-based cryptography, in: *Proceedings of ASIACRYPT*, 2002.

[19] B. Gopal and U. Manber, Integrating content-based access mechanisms with hierarchical file systems, in: *OSDI'99: Proceedings of the Third Symposium on Operating Systems Design and Implementation*, USENIX Association, Berkeley, CA, 1999, pp. 265–278.

[20] V. Goyal, O. Pandey, A. Sahai and B. Waters, Attribute-based encryption for fine-grained access control of encrypted data, in: *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2006.

[21] T. Hardjono and B. Weis, The multicast group security architecture, RFC 3740 (Informational), March 2004.

[22] D.R. Hardy and M.F. Schwartz, Essence: A resource discovery system based on semantic file indexing, in: *Proceedings of the USENIX Winter Conference*, USENIX Association, Berkeley, CA, January 1993, pp. 361–374.

[23] F.J. Hill and G.R. Peterson, *Computer Aided Logical Design with Emphasis on VLSI*, 4th edn, Wiley, 1993.

[24] J. Horwitz and B. Lynn, Toward hierarchical identity-based encryption, in: *Theory and Application of Cryptographic Techniques*, 2002, pp. 466–481.

[25] A. Kapadia, P.P. Tsang and S.W. Smith, Attribute-based publishing with hidden credentials and hidden policies, in: *Proceedings of the ISOC Network & Distributed System Security Symposium (NDSS)*, 2007.

[26] B. Lampson, Protection, in: *Proceedings of the 5th Annual Princeton Conference on Information Sciences and Systems*, Princeton University, 1971, pp. 437–443,

[27] B. Lynn, PBC library, 2006, available at: http://crypto.stanford.edu/pbc/.

[28] P. McDaniel, A. Prakash and P. Honeyman, A flexible framework for secure group communication, in: *USENIX Security Symposium*, 1999, pp. 99–114.

[29] P. McDaniel and A.D. Rubin, A response to "can we eliminate certificate revocation lists?", in: *FC'00: Proceedings of the 4th International Conference on Financial Cryptography*, Springer-Verlag, London, UK, 2001, pp. 245–258.

[30] A.J. Menezes, T. Okamoto and S.A. Vanstone, Reducing elliptic curve logarithms to logarithms in a finite field, *IEEE Transactions on Information Theory* **39**(5) (1993), 1639–1646.

[31] A. Miyaji, M. Nakabayashi and S. Takano, New explicit conditions of elliptic curve traces for FR-reduction, *IEICE Transactions on Fundamentals* **E84-A**(5) (2001), 1234–1243.

[32] M. Myers, R. Ankney, A. Malpani, S. Galperin and C. Adams, X.509 internet public key infrastructure: Online Certificate Status Protocol – OCSP, 1999, available at: http://www.ietf.org/rfc/rfc2560.txt.

[33] D. Nali, C. Adams and A. Miri, Using threshold attribute-based encryption for practical biometric-based access control, **1**(3) (2005), 173–182.

[34] M. Pirretti, P. Traynor, P. McDaniel and B. Waters, Secure attribute-based systems, in: *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2006.

[35] A. Sahai and B. Waters, Fuzzy identity based encryption, in: *Eurocrypt 2005*, 2005.

[36] R.S. Sandhu, E.J. Coyne, H.L. Feinstein and C.E. Youman, Role-based access control models, *Computer* **29**(2) (1996), 38–47.

[37] R.S. Sandhu and P. Samarati, Access control: Principles and practice, *IEEE Communications Magazine* **32**(9) (1994), 40–48.

[38] A. Shamir, How to share a secret, *Communications of the ACM* **22**(11) (1979), 612–613.

[39] A. Shamir, Identity-based cryptosystems and signature schemes, in: *Proceedings of CRYPTO 84 on Advances in Cryptology*, Springer-Verlag, New York, 1985, pp. 47–53.

[40] V. Shoup, Using hash functions as a hedge against chosen ciphertext attack, in: *EUROCRYPT*, 2000, pp. 275–288.

[41] P. Traynor, K. Butler, W. Enck and P. McDaniel, Realizing massive-scale conditional access systems through attribute-based cryptosystems, in: *Proceedings of the ISOC Network & Distributed System Security Symposium (NDSS)*, 2008.

[42] United States Department of Health and Human Services, Health Insurance Portability and Accountability Act, 1996, available at: http://aspe.hhs.gov/admnsimp/pl104191.htm.

[43] D. Yao, N. Fazio, Y. Dodis and A. Lysyanskaya, ID-based encryption for complex hierarchies with applications to forward security and broadcast encryption, in: *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2004.