Attribute-Based Messaging: Access Control and Confidentiality

Rakesh Bobba, Omid Fatemieh, Fariba Khan, Arindam Khan[1], Carl A. Gunter, Himanshu Khurana and Manoj Prabhakaran
University of Illinois Urbana-Champaign

Attribute Based Messaging (ABM) enables messages to be addressed using attributes of recipients rather than an explicit list of recipients. Such messaging offers benefits of efficiency, exclusiveness, and intensionality, but faces challenges in access control and confidentiality. In this paper we explore an approach to intra-enterprise ABM based on providing access control and confidentiality using information from the same attribute database exploited by the addressing scheme. We show how to address three key challenges. First, we demonstrate a manageable access control system based on attributes. Second, we demonstrate use of Attribute Based Encryption to provide end-to-end confidentiality. Third, we show that such a system can be efficient enough to support ABM for mid-size enterprises. Our implementation can dispatch confidential ABM messages approved by XACML policy review for an enterprise of at least 60,000 users with only seconds of latency.

Categories and Subject Descriptors: D.4.6 [**Operating Systems**]: Security and Protection—*Access controls*; *Cryptographic controls*; H.4.3 [**Information Systems Applications**]: Communications Applications—*Bulletin boards*; *Electronic mail*; E.3 [**Data Encryption**]:

General Terms: Design, Performance, Security

Additional Key Words and Phrases: attributes, messaging, attribute-based encryption

1. INTRODUCTION

Attribute-Based Messaging (ABM) enables messages to be addressed using attributes of recipients rather than an explicit list of recipients or mailing lists with pre-defined members. Such attributes can be derived from any available source, including enterprise databases, and dispatched as Internet electronic mail (email) messages or other types of messaging. For example, a message about a restricted fellowship opportunity could be emailed to all of the female graduate students in engineering who have passed their qualifying exams. Such dynamic lists provide three primary advantages over static mailing lists: efficiency, exclusiveness, and intensionality. *Efficiency* means that messages are more likely to reach only the recipients that care about them. For example, if a message for the faculty on sabbatical is sent only to the ones with that attribute rather than the general faculty mailing list, then six sevenths of the faculty will be spared an unwanted message. *Exclusiveness* means that a sensitive message excludes parties that should not receive the message. For example a message from the dean to the untenured faculty in a given department to solicit feedback on the clarity of tenure standards provided by the department's senior faculty might have this feature. *Intensionality* means that an address *describes* the recipients rather than listing

[1]Arindam Khan is affiliated with Indian Institute of Technology Kharagpur. This work was done when he was at University of Illinois Urbana-Champaign.
A preliminary version of this paper appears in the Proceedings of 22nd Annual Computer Security Applications Conference (ACSAC) 2006.

them. For example, a message to the attending and primary-care physicians for Sara Smith saves the sender the need to know the names or addresses of the recipients. ABM has applications in enterprises using the enterprise database for internal messages. It provides benefits for Customer Relationship Management (CRM) and similar circumstances where a sender needs targeted messaging to clients, members, and so on. It also has applications to alert messaging, like health alert networks.

However, to achieve its full potential, an ABM design must resolve significant security concerns. This paper focuses on two of these: access control and confidentiality. As for access control, ABM messaging becomes more beneficial as it exploits richer attribute information. However, user attribute information is sensitive and allowing anyone and everyone to target messages based on any or all user attributes could increase spam and violate the privacy of recipients. For example, who, if anyone, should be able to target a message to all of the employees who earn more than $150,000? It is possible to appoint an ABM super user as the only party that can send ABM messages, but a more scalable solution would regulate the rights of potential senders based on a general 'address authorization' policy. On the other hand, it is not obvious how to do this, since solutions like Access Control Lists (ACL) are likely to be unmanageable. As for confidentiality, current email systems offer the ability to encrypt messages end-to-end using public keys. For sensitive messages this provides valuable protection against compromised email relays or eavesdropping relay administrators. However, ABM cannot directly use this solution since message senders may not have an explicit list of the recipients of a message and, even if the recipients were known, it is probably not scalable to collect all of the necessary certificates to provide encryption for each recipient.

We propose an integrated architecture for an intra-enterprise ABM that provides access control using Attribute-Based Access Control (ABAC) and end-to-end confidentiality using Attribute-Based Encryption (ABE). Our analysis looks at the enforcement of security policies and the impact of compromises and discusses design issues related to privacy. We implemented the architecture and studied its performance. Overall we conclude that ABM with access control and confidentiality is feasible for deployment in enterprises.

The paper is organized into the following sections. In Sections 2 and 3 we set up preliminaries and outline our approach. In Sections 4 and 5, we discuss how ABAC and ABE are employed to secure ABM. In Section 6, we describe our architecture for ABM using ABAC and ABE. In Section 7, we analyze the security of ABM. In Section 8, we describe our implementation and look at measures of its performance. In Section 9, we discuss issues relevant to deployability and usability of ABM. In Section 10, we outline related work on targeted messaging and practical demonstrations of ABAC and ABE. In Section 11, we make conclusions, including limitations of our current work and possible future work.

2. PRELIMINARIES

We assume a context where there is a set of attributes that can be accessed and used for authorization and messaging. In particular, any enterprise has attribute data about its employees in its databases. We will refer generally to *users* for the parties who can send or receive ABM messages based on these attributes. A user can have zero or more values for any attribute, although we restrict users from having multiple numerical values for the same attribute (this restriction is needed by the encryption system). For example, a university might have the following attribute data on a user: $\{$*UserID = user089, Position = Faculty,* *Designation = Professor, Department = Computer Science, Department = Mathematics, CourseTeaching = CS219, CourseTeaching = CS486, CourseTeaching = MATH523, Date-Join = 24/06/1988, AnnualSalary = $80,000*}. In this example the user is affiliated with two departments and is teaching three courses. So he has multiple values for those attributes. In general, the attribute value pairs used in the system can be classified as, 1) *boolean:* those with a yes or no value, 2) *enumerated:* those with multiple non-numerical values and 3) *numerical:* those with multiple numerical values. This attribute information may not all be available in one centralized database but, instead, might be distributed over multiple databases that are managed by different units of an enterprise. An ABM system makes use of this information, abstracted as user attributes, to dynamically create recipient lists. To have this attribute information available to the ABM system we envision the use of a data services layer that presents a view of the attribute data after extracting it from the disparate databases. Some attributes are verified or established by the enterprise, like immigration status, age and salary, whereas others may be maintained by users, like a list of hobbies. Our focus in this work is on the former attributes.

An ABM system has three primary types of policies. Our approach to each of these will be given in technical detail in subsequent sections.

(1) The delivery policy is a sender-defined policy that specifies the set of users his message is targeted for. This is the 'ABM address' associated with a message. The message is routed only to users who have an attribute-set that satisfies this policy.

(2) The address authorization policy controls the ability of a user to target messages using an ABM address. This is a system-wide policy and specifies which users have permission to target messages to a given attribute based on their own attributes. Conceptually, this policy determines the set of users that have permission to send messages to a given ABM address and the set of ABM addresses to whom a given user is allowed to send messages. This policy therefore controls access to the system.

(3) The encryption policy is another sender-defined policy that indicates which users will have the ability to decrypt an encrypted message. The encryption policy associated with an encrypted message defines the combination of attributes needed to decrypt the message. This would typically be the attributes held by the recipients as specified by the delivery policy, but there are cases where key management is improved by allowing the delivery policy to be a subset of the encryption policy.

3. APPROACH

In this section we discuss the challenges in realizing an ABM system and present an overview of our approach which integrates ABAC and ABE into an ABM architecture. At a high level, we consider an ABM system to comprise the following components: an enterprise attribute database that provides user to attribute mapping functionality, a composition mechanism that enables senders to compose attribute addresses, a bridging mechanism that connects the ABM system with the enterprise messaging system, an ABM server that provides an interface to all enterprise users and other components, and finally components that achieve access control and confidentiality capabilities to enforce desired security requirements. In this work we use e-mail as the enterprise messaging system.

Access Control. When attributes are made available to be used to target messages in the ABM system, it is essential that access to such a system be controlled via appropriate rules.

To do so, the access control system would comprise a policy language that enables administrators to specify policies, a policy engine that acts as the Policy Decision Point (PDP) by evaluating specified policies against a given access request, and a policy enforcement point that enforces the decision. In order for the access control system to be manageable it must use access control techniques that specify an efficient mapping of permissions to services. For example, ACLs [Saltzer and Schroeder 1975] would not be a good policy model for ABM since the creation and management of such lists for a potentially large number of users and attributes would be unwieldy. To address this we turn to ABAC, which has proven to be successful in access control for distributed systems [Bonatti and Samarati 2002; Damiani et al. 2005; Li et al. 2002; Wang et al. 2004; Yu et al. 2003]. In ABAC a requester is granted access to a collection of objects or services based on a furnished collection of attributes. Translating this to our ABM system, a message sender is granted the permission to target messages to a set of recipients described using a collection of attributes based on his own collection of attributes. This approach has two advantages in terms of manageability. First, since the ABM system extracts attributes from enterprise databases for addressing purposes, using ABAC allows us to derive access control information from the same databases. Second, ABAC simplifies assignment and revocation of permissions. ABAC can be used on its own as an access decision mechanism or indirectly in connection with Role Based Access Control (RBAC) [Ferraiolo et al. 2003], where ABAC can be used to make role assignments based on attributes. We pursue the former in our ABM design.

Message Confidentiality. A message confidentiality protection component is needed to protect sensitive message contents from eavesdroppers. While it is relatively easy to protect against packet sniffing, it is more challenging to protect message contents from the intermediate servers such as the ABM server and the Mail Transfer Agent (MTA). Traditional end-to-end message confidentiality protection techniques are not suitable for ABM as the sender does not know who the recipients of the message are. We address this challenge by employing ABE (*e.g.*, [Sahai and Waters 2005; Goyal et al. 2006; Pirretti et al. 2006; Bethencourt et al. 2007; Katz et al. 2008]), an emerging encryption paradigm that enables a user to encrypt messages using attributes. Translating this to our ABM system, a sender can encrypt his message using attributes so that only users that satisfy the specified attributes can decrypt the message. This approach has two advantages. First, a message sender can encrypt his message directly (end-to-end) to the recipients without having to trust intermediate servers with the message contents. Second, the ABM address can also be used to specify the users that could decrypt the message.

Policy Specialization. Imagine an ABM system in which a sender does not have explicit knowledge about the attributes that he is allowed to use for targeting messages. In such a setting, the sender needs to create his desired ABM address, target his message using this address, and then wait for the system to let him know whether he has the permission to target using that address. Clearly this is a cumbersome process for senders. Hence, in order to make the ABAC policy for address authorization a usable tool for the sender it is desirable to give the sender assurance that his message will be delivered. Our approach is a technique we call *policy specialization,* in which a general policy is specialized to a given user and presented to the user in a way they can understand and use. There are several ways to realize policy specialization. First, it can be a feature of a fully web-based email system. Second, it can be a plug-in on traditional email clients. Third, it can be a completely independent system that would assist users in creating address for ABM messages. Deploying this feature on a fully web-based email system, where such systems are used, would be straight forward. However, there are many enterprises that use traditional client-based (non-browser) messaging systems. Deploying this feature as a plug-in implementation for such systems, on the other hand, introduces the need for developing plug-ins for diverse Mail User Agents (MUAs) and thus does not provide platform independence. We opt for the third option for its ease of deployment. In our realization, a sender logs on to a web server to compose an ABM address. The ABM address is returned to the sender in a file that he can attach to his message. The sender then sends his message to a pre-specified email address of the ABM server and the ABM servers delivers it to the intended recipients. The ABM addresses need not be kept secret; their privacy issues are discussed in Section 9.

Integration. An example illustrates how the components of our ABM system are integrated (further details are provided in Section 6). Imagine an ABM system deployed in an academic department. A staff member aims to send a message to all female graduate students who have passed their qualifying exam about a fellowship opportunity. First he accesses the web page of the ABM system to create an ABM address. The policy engine specializes the organizational policy to this user, indicating the ABM addresses that the user is allowed to use. The user then forms the desired attributes into an ABM address. After composing the message, if confidentiality is required, the user encrypts the message using ABE. The ABM address is added to the encrypted message as an attachment and sent to a distinguished email address at an MTA using the user's standard email client. The ABM server collects the email from this in-box and dynamically creates a distribution-list using the attached ABM address to query the enterprise attribute database. The message is sent to this distribution list. Note that the recipients need to have access to proper attribute-based decryption keys in order to decrypt the message. These keys are provided to users by an Attribute Authority (AA). Further details on the AA entity are provided in Section 5.

Efficiency. The efficiency of the ABM system can best be gauged via prototype implementation and experimentation. With an eye towards rapid prototyping and performance we have leveraged several commercial off-the-shelf components to build a prototype ABM system. We envisioned the PDP, database, and AA to be the major computationally intensive components of our system. To achieve high performance, we had to make a number of design and implementation decisions that are detailed and justified in Sections 6 and 8. Our experiments show that with an enterprise of $60,000$ principals using its existing enterprise database or an XML database view of it, both the access control decision and dynamic list creation procedures can be done within reasonable time limits. Confidentiality protection adds only a few seconds of latency at the client side and the AA is shown to support the same user population with a reasonable vulnerability window (see Section 9).

4. ABAC FOR ABM

To target an ABM message to a group of recipients, a sender specifies the attributes in a logical expression. This expression is referred to as an *ABM address* and constitutes a *delivery policy* for the users to whom the message should be delivered. Table I describes the language. For example, if A is the address *(Position = Faculty* **or** *Position = Staff)* **and** *(Salary > 100000)*, then a message with this delivery policy would be sent to faculty and

I: Grammar for ABM Addresses and Rules

	a	$\in$	$Attribute$
	v	$\in$	$Numerals \mid Strings$
Delivery Policy	R	$::=$	c
Condition	c	$::=$	$l \mid (c$ **or** $c) \mid (c$ **and** $c)$
Literal	l	$::=$	$(a\ rel\ v) \mid (v\ rel\ a\ rel\ v)$
Relation	rel	$::=$	$< \mid > \mid = \mid \leq \mid \geq$
Authorization Rule	S	$::=$	$l \leftarrow c$

staff with salaries of more than \$100,000. In practice, delivery policies can be specified using the language of the database (like SQL) or via a commonly used query language that can be executed on a variety of database technologies (like XQuery).

Specifying a unique authorization rule for every possible ABM address is not practical. To address this issue, we take a simplified and pragmatic approach. We specify an authorization rule per *address literal*. An address literal could be a value for an attribute or a range of values for the attribute. That is, a user is permitted to target a message to a given address literal (attribute value pair or attribute value range) based on attributes he has. So each address authorization rule controls sending to one (authorization) literal and not to an ABM address which may contain many such literals. A user is allowed to target a message to a given ABM address if he is allowed to target a message to all literals in the ABM address. It can be argued that this provides a reasonable and manageable alternative to writing an authorization rule per ABM address. Consider a user who is allowed to target a message to two individual attributes. Such a user could send two emails, one targeting each attribute, to achieve the same effect as targeting an ABM address consisting of both the attributes combined with the **or** operand. Similarly, instead of choosing the **and** operand the user can send an email to a broader audience but ask some of them to ignore it. While these arguments do not provide an absolute proof that this is the best way to manage authorization for all applications, they at least suggest it is a good baseline. Furthermore, this approach has the simplifying effect of making it the case that at most one authorization rule is required for each range of values of an attribute (*i.e.* literal).

An authorization rule S has the form $l \leftarrow c$ (see Table I) where l is the *head* and c is the body. For example

$$21 \leq Age < 65 \leftarrow Position = Faculty \textbf{ or } (Position = Staff \textbf{ and } Designation = Coordinator)$$

says that a sender who is a faculty member or a staff coordinator can route a message to users whose age is between 21 and 65. The set of all address authorization rules is the *address authorization policy*.

Policy specialization enables the sender to compose exactly the ABM addresses the authorization rules permit her to use. If the attributes of a user satisfy the condition of an address authorization rule then that rule is included in the user's *specialized policy*. Thus a subset of the authorization rules in the address authorization policy constitute a user's specialized policy. The address literals in his specialized policy are said to be the set of *routable attributes* for that user. A literal in his routable (or permitted) ABM address will define the same value or range, or a subset of range of values from one of his routable attributes. Rather than give a formal semantics of how specialized policies are derived, we offer an example that shows the basic idea. Recall address A from before (faculty and staff with salaries of at least \$100,000). Assume that the sender's specialized policy has routable attributes for *Position = Faculty, Position = Staff,* and *Salary* $\geq$ *5000*. This means that the sender can send a message using a combination of at least these literals, and, in the case of the numeric range, any subset. So this set of routable attributes means that the sender can send a message to address A.

5. ABE FOR ABM

In this section we describe Ciphertext-Policy Attribute-Based Encryption (CP-ABE) and how it is employed to provide end-to-end confidentiality for ABM. CP-ABE is a flavor of ABE where an attribute policy is associated with ciphertext and attributes are associated with users. Only those users whose associated attributes satisfy the policy associated with a ciphertext are able o decrypt the ciphertext. Our design and implementation is based on the system described in [Bethencourt et al. 2007], and much of our background discussion is drawn from there. Hereafter, we use the term CP-ABE to refer to the specific scheme of [Bethencourt et al. 2007]. CP-ABE comprises four basic algorithms: Setup for generation of master parameters, KeyGen to generate user keys, Encrypt for encrypting data and Decrypt for decrypting ciphertexts.

Setup. Setup generates a master public-key PK, and a master private-key MK. PK is required for all the other functions and is made available publicly. MK is only used for key generation. A trusted AA executes the Setup algorithm and is entrusted with the master private-key and the task of issuing private attribute keys to users.

*KeyGen(*MK, A, u*).* This function is executed by the AA and generates a private key set SK_u for the user u, based on his set of attributes A, using the master key MK. The AA checks with the enterprise database before issuing attribute private keys to users to ensure that users get access to only those attribute keys for which they are authorized.

The AA chooses a unique random $r_u \in \mathbb{Z}_p$ for every user u, and a set of random $\{r_{(u,a)} \in \mathbb{Z}_p \mid a \in A\}$ during the key-generation process. Choosing a unique number r_u for each user in U to randomize key components associated with each user attribute ensures that two users cannot collude by combining their keys. The CP-ABE scheme considers attributes simply as labels, *i.e.*, arbitrary strings, rather than as *attribute, value* pairs as described in Section 2. Furthermore, the underlying mathematics can only check for equality of strings and hence only equality of strings is supported by default in encryption policies. However, [Bethencourt et al. 2007] show how *attribute, value* pairs can be supported, and how relations $<, >, =, \leq, \geq$ can be supported in encryption policies when attributes take numerical values. Boolean attributes are represented using just the attribute name since only positive Boolean attributes are ever used as only monotonic policies are supported. Enumerated attributes are converted into multiple unique strings by concatenating the attribute name, a delimiter and one of the attribute values. In order to support numerical attributes and allow numerical comparisons in policies, CP-ABE uses strings to represent individual bits of the numerical value. That is, numerical values are represented using a bag of strings, one for each bit of the value. For example, a 3-bit numerical attribute *Level* with value 4 is represented using the following strings: *Level:1***, *Level:*0**, and *Level:**0*. Now a policy with a numerical comparison can be represented using equalities on the strings representing bits. For example, the policy *Level* $\geq$ *2* can be translated as *Level:1** OR Level:*1**.

*Encrypt(*PK, M, $\mathcal{T}$*).* The encryption function uses public parameters PK and an attribute policy represented as an access structure $\mathcal{T}$ to encrypt the message M and produces a ciphertext CT such that only a user who possesses a set of attributes A that satisfy the access structure $\mathcal{T}$ will be able to decrypt it.

An *access structure* is a tree whose leaf nodes are literals in the policy and internal nodes are threshold gates. A threshold gate is defined by the number of its children and a threshold value. A threshold gate with 2 children and a threshold value of 1 is an OR gate while a gate with 2 children and a threshold value of 2 is an AND gate. When sending a message, users can incorporate the *expiry* attribute into their encryption policy to ensure that users with expired keys cannot decrypt the messages they obtain. Note that the number of leaves in the access tree is same as the number of literals in the policy. However, in case of a numerical comparison literals they will be further expanded into subtrees consisting of string literals as explained in the key generation function above. Hereafter we refer to the attribute policies used to encrypt data as *encryption policies* and use the terms access tree and policy tree interchangeably.

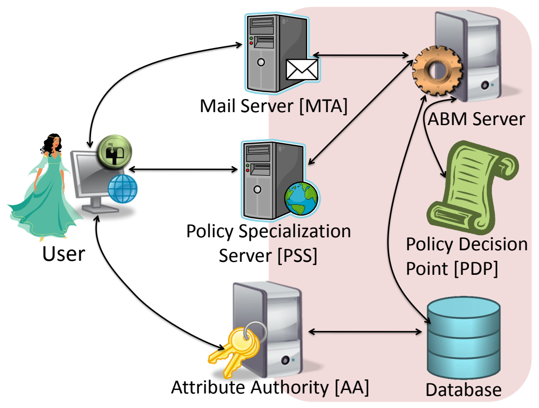
*Decrypt(*PK, CT, SK$_u$). The decryption algorithm decrypts the ciphertext CT for user u only if the attribute set A of the user, represented by SK$_u$, satisfies the access structure $\mathcal{T}$. CT implicitly contains $\mathcal{T}$. Intuitively, each node in access structure holds a secret. The secret held by an internal node can be obtained if, and only if, one has access to secrets held by threshold number of its children. For example, to obtain the secret held by an AND node one needs to obtain the secrets held by both its child nodes. The secret held by a leaf node can be obtained if and only if one has access to attribute key for the attribute represented by the leaf node. Obtaining the secret held by the root node of the access tree allows one to decrypt the message.

To protect a message end-to-end in ABM, users may encrypt the message using CP-ABE with an attribute policy such that only intended receivers can decrypt the message. In our implementation, the encryption policy is effectively the same as the delivery policy. Since the delivery policy uses attribute value pairs we implemented a policy translator that converts a given delivery policy into a valid encryption policy by converting all attribute value pairs, except for numerical attributes (numerical attributes are automatically converted by the CP-ABE implementation we used), in the delivery policy into unique attribute strings as described in the key generation algorithm above. Users obtain attribute keys to decrypt messages from an enterprise AA that checks against the enterprise database before issuing attribute keys to users.

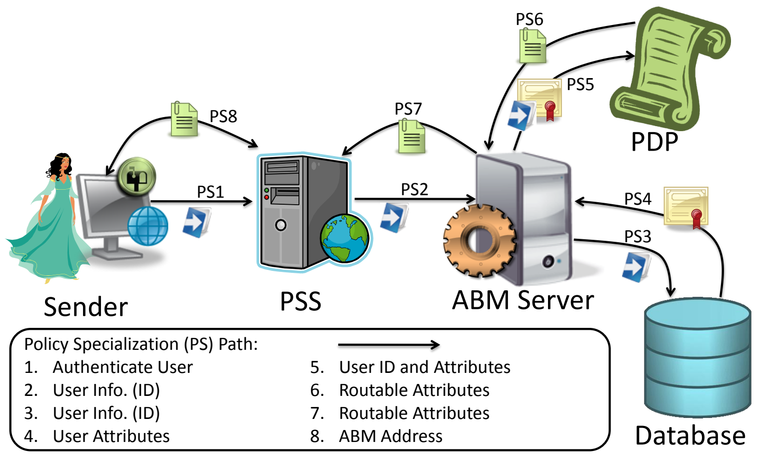
6. ARCHITECTURE

Figure 1 illustrates the architecture of our ABM system and its associated security system, which strongly influences the overall structure. The ABM system comprises a Policy Specialization Server (PSS) to authenticate and help users compose policy compliant ABM addresses, a PDP with the address authorization policy, an attribute database, an ABM server associated with an enterprise MTA that resolves ABM addresses to recipient lists and mediates other components, and an Attribute Authority that issues attribute keys to the users. These components provide an infrastructure for three attribute-based policies for messaging, the system-wide address authorization policy, the user-defined delivery policy and the encryption policy. The communication protocols in our system can be classified into four functional classes which we call *paths* for: policy specialization, messaging, address resolution, and attribute keying. We discuss each in turn.

The *Policy Specialization (PS) Path* authenticates the user, evaluates his attributes from the database with the policy decision point (PDP), and provides the address authorization



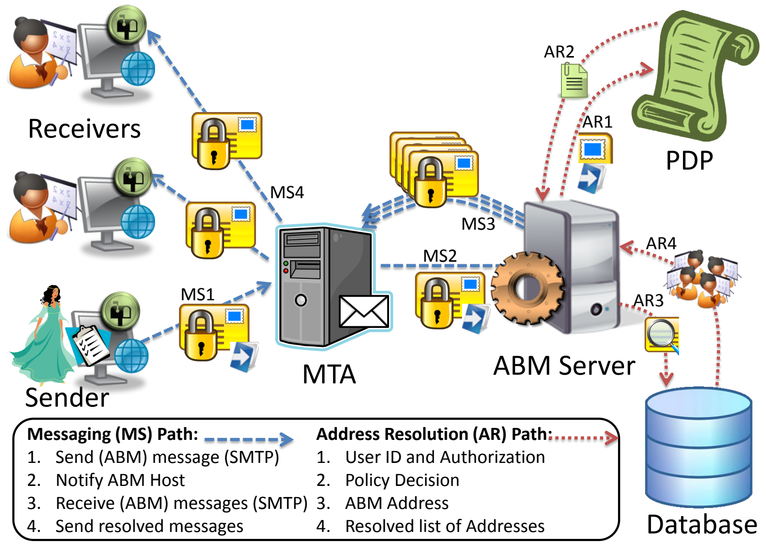
1: ABM Architecture



2: Policy Specialization Path

policy specialized to the user. This eight step communication is represented by solid lines in Figure 2. In the first step, PS1, the user logs into the PSS. The PSS uses the enterprise authentication infrastructure to authenticate the user. Next at PS2, the PSS sends the user's information to the ABM server and requests the specialized address authorization policy for the user. In steps PS3 and PS4 the ABM server queries and retrieves the user's attributes from the attribute database. In step PS5 the ABM server sends the user's attributes to the PDP and requests for the specialized policy. The PDP then evaluates all the address authorization rules in the policy against the user's attributes to form the specialized address authorization policy[2]. In step PS6 it returns the routable attributes (literals) from the specialized authorization policy that the user can route on. The ABM server then returns the specialized authorization policy to the policy specialization server in step PS7. In step

[2]Note that the PDP in our architecture has more built-in functionality than traditional PDPs which just return grant or deny to requests based on their policy.

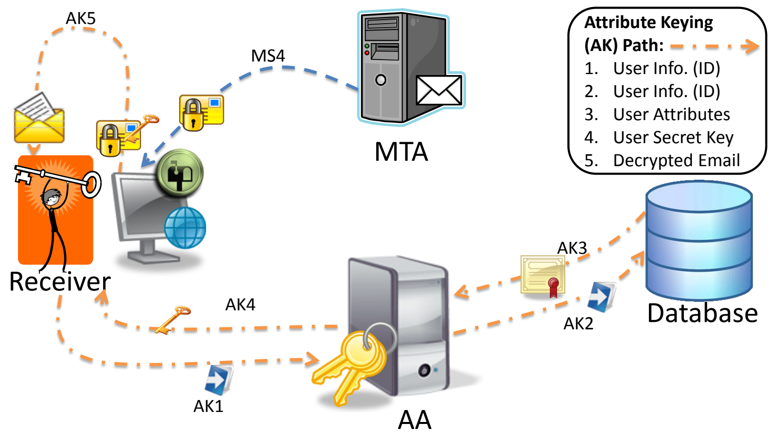

Receivers
MS4
AR2
AR1
PDP
MS3
MS2
AR4
MS1
AR3
MTA
ABM Server
Sender
Database
Messaging (MS) Path:
1. Send (ABM) message (SMTP)
2. Notify ABM Host
3. Receive (ABM) messages (SMTP)
4. Send resolved messages
Address Resolution (AR) Path:
1. User ID and Authorization
2. Policy Decision
3. ABM Address
4. Resolved list of Addresses

3: Messaging and Address Resolution Path

PS8, the policy specialization server provides an interface to the user to create a delivery policy by combining the routable address literals in the specialized authorization policy with ranges and boolean connectives as described in Table I. This is then saved by the user as the ABM address (delivery policy).

In the *Messaging (MS) Path*, users send and receive ABM messages using any standard MUA (dashed lines in Figure 3). The ABM address is translated to an encryption policy. A user composes a message and encrypts the body of the message using the encryption policy. The delivery policy (or ABM address) is included in the message as an attachment. The message with the encrypted body and ABM address as the attachment is signed using S/MIME [Ramsdell 2004a; 2004b] for sender authentication. The user then sends the message to a pre-specified email address such as `abm@localdomain.com` in step MS1. The enterprise MTA is configured to notify the ABM server when it receives a message for the pre-specified address as shown in step MS2. After processing the message, as described in address resolution path below, the ABM server invokes the enterprise MTA in step MS3 to deliver the message to a list of recipients as specified by the ABM address. Each receiver gets her message in her inbox in step MS4.

In the *Address Resolution (AR) Path*, the ABM server processes messages to authenticate the sender, determine whether the sender is authorized to target the message based on the associated delivery policy, and determine the recipients defined by the delivery policy (dotted lines in Figure 3). Upon receiving the message, the ABM server: 1) verifies the S/MIME signature on the message to authenticate the user, and 2) queries the attribute database for the sender's attributes. In step AR1, the ABM server checks with the PDP that the sender is authorized to send the message to the ABM address included in the message. In step AR2, the PDP evaluates the delivery policy for accessing the attributes contained in the ABM address against the sender's attributes and responds in the affirmative only if the user is allowed access to all attribute literals in the ABM address. The ABM server then resolves the ABM address to a list of email addresses by querying the attribute database in



4: Attribute Keying Path

steps AR3 and AR4.

The *Attribute Keying (AK) Path* describes steps of the AA, which is similar to a certificate authority and supports keying needs of users such as attributes and S/MIME certificates. After receiving an encrypted message, if the user does not have a current set of keys to decrypt the message, she requests them from the AA (dashed-dotted lines in Figure 4). A user authenticates to the AA in step AK1. The AA sends the user information (*e.g.* user id) to the enterprise database in step AK2. The database responds with the most current information about the user's attributes in step AK3. With the attribute set the AA gets from the database, it generates cryptographic attribute key set for the user and sends it back to the user over a secure channel in step AK4. The user can now decrypt her message using this key in step AK5.

7. SECURITY ANALYSIS

In this section we analyze the security of the architecture proposed in Section 6 by examining enforcement and the impact of compromise.

7.1 Security Policy Enforcement

Address Authorization Policy. By design, the ABM architecture and protocols enforce the address authorization policy by ensuring conformance to it all the way from address composition to delivery. This is achieved via a three-step process. First, the sender is required to digitally sign (via S/MIME) the message, which includes the ABM address as an attachment. Second, the ABM server verifies the S/MIME signature to authenticate the sender. And third, by checking with the database and PDP, it ensures that the authenticated sender is not violating the address authorization policy. We use S/MIME signatures because current email systems do not provide strong security measures to ensure that the message sender is who he claims to be. An adversary can potentially violate the address authorization policy by sending messages masquerading as an authorized sender by spoofing the "from" address of the email message. When it is supported, SMTP-AUTH [Myers 1999] can prevent external spoofing as it verifies that the authenticated user has permission to use the MTA. However, it does not necessarily bind the message's "from" address to the authenticated user identity. An authenticated user potentially can send email as any other user. So an adversary can send messages masquerading as any user in the system by just compromising one user account. In the proposed solution, the use of S/MIME signature provides the necessary sender authentication. To assist with key management for S/MIME the AA acts as a CA to issue S/MIME certificates in the ABM system.

Delivery and Encryption Policies. In our implementation of ABM, the set of users specified by a the delivery policy and encryption policy of the message is the same. That is, if we denote the set of users specified by delivery policy by $\mathcal{D}$ and the set of users defined by encryption policy by $\mathcal{D}$ then $\mathcal{D} = \mathcal{E}$. The delivery policy is enforced in the ABM system using trusted components. The encryption policy, enforced by attribute-based encryption, ensures end-to-end confidentiality between the sender and the receiver. That is, the sender does not need to trust the ABM server with the confidentiality of his message.

As mentioned in Section 6 a delivery policy is realized to list of receivers from the database online through the address resolution path. However, attribute keys are issued per request and are not synchronized with the attribute database continuously, leading to a potential vulnerability. For example, consider a scenario where an attribute of a user is revoked before his attribute key expires (*e.g.* dishonest manager, course dropped by a student at mid semester or a doctor under the suspicion of malpractice). Since the attribute database is updated immediately, and the recipients specified by the delivery policy are verified against the database only the correct set of users get the message. But as there is no key revocation mechanism in CP-ABE, users with revoked attributes still have the keys for those attributes until the keys expire. For example, a dishonest manager may be revoked on October 2nd, but he might have valid keys for ⟨ *Designation, Manager*⟩ with ⟨*expiry, 31/12/2007*⟩. The time between when a user's attribute is revoked and when his attribute key actually expires is referred to as the *vulnerability window*. During this time a delivery policy defines a subset of the people defined by the corresponding encryption policy, $\mathcal{D} \subset \mathcal{E}$.

The vulnerability window is bounded above by the length of time for which an attribute key is valid. The smaller the length of validity of an attribute key the smaller the vulnerability window. Therefore, an organization should determine the validity length of an attribute key, *i.e.*, its expiry time, based on the sensitivity of the attributes and the vulnerability window it is willing to tolerate for the attributes. Also the risk during the window is probabilistic depending on the rate of revocation, sensitivity of the revoked attribute, and rate of use of that attribute in encryption policies. For example, the dishonest manager has a chance of violating the security only if there is a message addressed for ⟨*Designation, Manager*⟩ that he can steal.

A similar situation may arise when a user joins or gets a new attribute. For example, if a user is promoted to the position manager and the database is updated right after a message is send to all the managers, he will not get it as the database has not been updated yet. But shortly after that he will have the keys to read it. In summary, having no revocation mechanism for user attribute keys means there is a risk that a user has more valid attribute keys than attributes in the current snapshot of the database.

7.2 Compromise Analysis

When no components of the ABM system are compromised, messages are sent, routed and read in compliance with the address authorization policy, delivery policy and encryption policy respectively. In this section we analyze whether this compliance is affected by one or more compromised components or users in the ABM system. We discuss compromise of the ABM server, AA, PDP, PSS, and multiple users. Enterprise attribute database and enterprise authentication server compromise affect the integrity of the ABM system as it is dependent on those services just like other services/apps in the enterprise. So we do not discuss them as part of ABM security analysis. Though MTA is a component external to the ABM system we are protected to some degree against its compromise and we discuss that in a later paragraph.

ABM Server Compromise. First, if an agent is able to compromise the ABM server she will have access to all the messages being relayed through the ABM server. However, as these messages are encrypted by the senders end-to-end, the ABM server does not have access to the contents of the messages. Therefore, the agent cannot read those messages. Second, since the ABM server acts as the policy enforcement point for both address authorization and delivery policies, any adversary that compromises it can violate authorization and delivery policies. That is, unauthorized users may be able to send messages, and legitimate messages may be delivered to unauthorized recipients. However, the unauthorized recipients of a message will not have sufficient attribute keys to decrypt the message. Third, this agent has now read access to all the information contained in the database. So ABM has to be secured as well as the database of the enterprise.

MTA Compromise. A compromised MTA raises vulnerabilities similar to ones from compromised ABM server. The attacker has access to all the messages on MTA and can forward them to anybody but as the message content are protected end-to-end using ABE confidentiality is retained. Additionally, the attacker will be able to send a message as any user or from any "from" address. But ABM is protected from such an attack as all the messages have to be signed by the sender using S/MIME. Another problem with MTA compromise is the attacker's ability to drop messages. But also dropping messages makes the attacker more visible thus aid intrusion detection.

AA Compromise. If an agent is able to compromise the AA it can generate any attribute key for any user. Thus it can violate encryption policy and therefore confidentiality of all messages that it can get access to. But to gain access to all the messages it also has to compromise the ABM server or the MTA. It can also stop generating keys or generate wrong keys for the legitimate users and make their ABM messages unreadable.

PDP Compromise. If an agent is able to compromise the PDP it can clearly violate the address authorization policy by giving arbitrary or false responses to policy decision requests. But, the PDP in itself does not send or deliver messages. The agent also needs to compromise a sender to be able to send arbitrary messages to arbitrary groups.

PSS Compromise. If an agent is able to compromise the PSS it can mislead the senders by giving them incorrect specialized policies. But senders will not be able to send messages that violate the address authorization policy as verification by the PDP at step AR2 (see Figure 3) would fail for such messages. However, the adversary is able to frustrate users by letting them send messages that will be denied delivery.

User Collusion. If an agent is able to compromise multiple users it may attempt to violate the encryption policy by combining attribute private keys of those users. However, this attempt will fail as ABM derives its collusion-resistance from CP-ABE. For example, if an email is encrypted for all graduate students taking the course CS563 then neither an undergrad student taking CS563 nor a graduate student individually has enough attributes to satisfy the policy. If the agent attempts to violate this encryption policy by compromising two students that when combined have the necessary attributes, he will fail. This is because as discussed in Section 5, every attribute key in CP-ABE is seeded with a user specific component r_u. As a result keys from two users cannot be combined to decrypt such an email.

Denial-of-Service. Whenever an agent compromises a service-providing component such as the ABM server, MTA, AA, PDP or PSS the system can become unavailable. The attacker can stop the regular services of the component and bring the ABM system to a halt until the compromise is detected and recovered from.

8. IMPLEMENTATION AND EXPERIMENTAL EVALUATION

To test that the architectural framework presented in Section 6 satisfies the requirements for ABM, we implemented a prototype ABM system. We used this prototype implementation as a test bed for experimental evaluation. This section provides details on the prototype implementation, experimental setup, and performance results.

8.1 Implementation

We implemented components of an ABM system and used them as a test bed for experimental evaluation. We had to make a number of decisions on the technologies to use for the major components of our proposed architecture. These decisions and the rationale behind them are briefly discussed in this section.

PDP. We chose to use XACML and Sun's standards-compliant implementation of its policy engine for our implementation. There are several advantages to XACML for our practical demonstration. First, XACML lends itself very well for ABAC policy specification as the framework supports attributes. Second, the XACML standard has widespread support from industry and standards bodies and this may support adoption. Third, its successful integration in several commercial products (`http://docs.oasis-open.org/xacml/xacmlRefs.html`) as well as research projects [Lorch et al. 2003] indicates the confidence in its deployability and effectiveness. An XACML policy file is stored in conjunction with the PDP. This policy file contains the address authorization rules for sending messages based on address literals. The implementation supports numerical, enumerated and boolean attributes.

Database. The implementation uses two different database representations, relational and native XML. We included an XML database representation in our evaluation as we envision data abstracted from heterogeneous enterprise databases to be in XML format. The queries submitted to the XML database are XQueries, and the queries for the relational database are expressed in SQL. We had to chose a database management system with support for XML and XQuery as well as SQL. We used Microsoft SQL Server 2005 (Standard Edition), which provides support for all the above mentioned data models and query languages.

ABM Server. The ABM server is associated with an enterprise MTA. The ABM Server gets automatically invoked when the MTA receives an ABM message targeted for the inbox associated with the ABM Server. This enabled us to use our domain MTA without any modification. We used C# to implement the ABM Server, and used the University of Illinois MTA as the enterprise MTA.

AA/Encryption/Decryption. We encrypt messages using CP-ABE with ABM address as the encryption policy. Messages are decrypted using attribute private keys issued by an AA. We used the CP-ABE toolkit (available at `http://acsc.csl.sri.com/cpabe/`) to implement an AA and to encrypt and decrypt messages. It uses 160-bit elliptic curve group based on the supersingular curve $y^2 = x^3 + x$, over a 512-bit field. While the CP-ABE scheme we used can be extended to be secure against a Chosen Ciphertext Adversary (CCA) using techniques like Fujisaki-Okamoto transformation [Fujisaki and Okamoto 1999] the implemented version is only secure against a Chosen Plaintext Adversary (CPA). However, since such a transformation adds minimal overhead, a couple of hash computations, the resulting overheads can safely be ignored.

8.2 Test Bed

Studying the components in Figure 1, we anticipated that the major resource consuming components would be the database, PDP, and the AA. Based on this assumption, we decided to place them on different machines on the network. The SQL Server database was running on a Windows 2003 Server with dual Intel Xeon 3.2GHz processors and 1 GB of memory. PDP, Web server and ABM Server were running on 2.8 GHz Pentium 4s with 1GB of memory and Windows XP Pro operating system. The encryption, decryption, and key generation experiments are performed on a Dell Poweredge 2950, with two Intel Xeon 2.66Ghz CPUs and 4GB of RAM running Linux.

8.3 Experimental Setup

The goals of the experiments are to evaluate the performance of our ABM system both with and without security. These goals enabled us to demonstrate the feasibility of the system as well as determine the additional costs imposed by security. To evaluate the performance without security we study the performance on the messaging path and address resolution path but without the authorization check and confidentiality protection. To evaluate the performance with security we study the performance on the messaging, address resolution and attribute keying paths described in Figures 3 and 4.

Since we use the University of Illinois MTA we ignore the message delivery latency on the messaging path as it would be the same for ABM messages as it is for regular email. Since S/MIME is built into most commonly used email clients and its performance is well understood, we do not characterize it in this evaluation. It suffices to say that S/MIME signatures do not unduly burden end users in terms of performance. Furthermore, since the costs of confidentiality protection are mostly at the client side (for encryption and decryption during messaging) and clients can choose not to encrypt messages, we study them separately from access control costs. Therefore, we study (1) the performance on the address resolution path to characterize access control overhead and (2) the performance on attribute keying path as well as encryption and decryption costs to characterize confidentiality protection overhead. We also evaluate the performance of our policy specialization service described in Figure 2.

To carry out the above described evaluation we vary the following parameters: (1) the complexity and number of address authorization rules, (2) the number of users and their assignment to a varying number of attributes in the database, (3) the complexity of ABM addresses and (4) the length of the window of vulnerability.

Address Authorization Policy Generation. The complexity and number of address authorization rules affects the latency of the policy specialization path and the authorization check on the address resolution path. We wrote a probabilistic XACML policy generator using Java, which created uniformly random authorization rules of varying complexity in a disjunctive normal form, which satisfies the proposed grammar in Table I. The condition in each rule consists of a number of *terms* combined with the *or* operand. Each term consists of a number of literals (as defined in the grammar) combined with the *and* operand. Specifically, the number of terms and the number of literals in each term were uniformly drawn between one and five, creating relatively simple to reasonably complex conditions. The number of rules depends on the number of address literals and we varied the number of attributes between 25 and 125 with an average of 5 values (or value ranges) per attribute resulting in 143 to 674 address authorization rules.

Database Population. The distribution of attributes in the user population affects the number of recipients a given ABM address resolves to, which in turn affects the latency of the address resolution path. The number and type of attributes a user has also affects the attribute-key generation time. Users were assigned an attribute based on the incidence probability of that attribute. For example, if an attribute has an incidence probability of 0.1 then 10% of the user population is assigned that attribute. For our test database, most of the attributes (80%), had a probability of incidence that ranged from 0.0001 to 0.01, 10% had a probability of incidence that was between 0.5 and 0.9 and the remaining 10% had the probability close to 1. This distribution allowed a big range in the number of recipients per message, and, intuitively, this distribution also reflects organizations where all the users have some common attributes and rest of the attributes are sparsely distributed in the population.

ABM Address Generation. The complexity of an ABM address affects the performance on the address resolution path by affecting both the number of recipients it resolves to and the database query resolution time. It also affects the encryption and decryption latencies as ABM addresses are also used as encryption policies after appropriate translation by our tool. Similar to our approach for address authorization policy generation we varied the number of terms for a given address query between one and five (chosen randomly) and the number of literals in each term between one and three (also chosen randomly). Each literal was randomly assigned an attribute from the routable list of attributes of the message sender.

8.4 Performance Measurements

Address Resolution Path. The performance on this path is translated to the latency between the time an ABM message is received by the ABM Server until the time the message is sent out to the MTA for distribution.

For the case with access control this latency includes the time for: (1) verifying the signature on the message to authenticate the sender (2) consulting the PDP for authorization (3) retrieving the list of the recipients specified by the ABM address from the database, and (4) re-composing the message with the list of recipients. For the case without access control only the third and fourth latency components were included.

We performed our tests using databases of user size ranging from 15,000 to 60,000. Each of the experiments was performed on a sample of 100 users chosen uniformly at random from the corresponding databases. Table II summarizes our results. The Average List Size field in the table refers to the average number of recipients that the ABM addresses resolved to. The ABM addresses used had 2.5 terms on average and each term had 2.5 literals on average. There were 100 attributes in the system and 568 address authorization rules. There were 2.5 terms on average per rule and 2.5 literals on average per term. It is worth mentioning that since the databases were probabilistically filled, users were randomly selected, and the queries were also probabilistically generated, we had no direct control on the average list sizes.

II: Address Resolution Times for Relational and XML Databases.

DB Size (No. of Users)	Relational Database			XMLDatabase		
	Avg. List Size	Address Resolution Time Mean 95% Conf. Interval (ms)		Avg. List Size	Address Resolution Time Mean 95% Conf. Interval (ms)	
		With Access Control	Without Access Control		With Access Control	Without Access Control
60K	422	(367, 522)	(83, 244)	745	(4882, 6262)	(4628, 5970)
45K	302	(334, 494)	(65, 206)	472	(4169, 4911)	(3599, 4436)
30K	220	(317, 379)	(61, 116)	317	(2840, 3417)	(2581, 3151)
15K	145	(315, 347)	(50, 72)	171	(2541, 3057)	(2067, 2624)

Messaging and Attribute Keying Paths. As discussed earlier we measure, (1) encryption and decryption costs on the messaging path and (2) key generation costs on the attribute keying path to characterize end-to-end confidentiality protection overheads. We encrypted each message using the associated ABM address, generated as described above, after translating it into a valid encryption policy and measured the encryption time. The literals in these encryption policies can be classified into the following two types:

—Equality: Literals containing an equality condition for a boolean, enumerated or numerical attribute, *e.g.*, *Position* = *Faculty*, which are converted to unique strings, *e.g.*, *"Position_val_Faculty"* in the encryption policy.

—Relational: Literals including one of the following relational conditions $>$, $<$, $\geq$, or $\leq$ for a numerical attribute (*Salary* $\geq$ 60000).

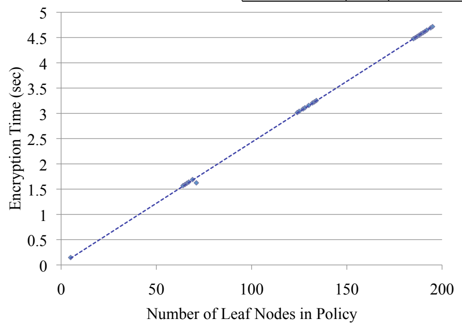
Table III summarizes the encryption times of our policies based on the number and types of their constituent literals. For example, encryption with a policy containing two relational literals and two equality literals takes 1.57 seconds. As an alternative view on the encryption times, we measure encryption time as a function of the number of leaf nodes in the expanded (*i.e.*, numerical comparisons other than equality are expanded as explained in Section 5) policy tree. Figure 5a summarizes the results. A trendline is used to show the linearity of encryption time w.r.t. number of leaf nodes. Note that the same underlying data is used in both Table III and Figure 5a and only the representation is different.

For each policy, a representative set of keys that satisfy the policy are generated and used for decryption. Specifically, (1) a key is generated for each term in the policy such that it satisfies the term and (2) a key is generated for each combination of term in the policy such that the key satisfies all the terms in the combination. For each case, the average decryption time is recorded. The results are shown in Figure 5b.

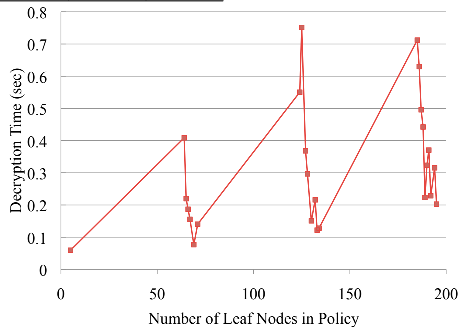
Both encryption and decryption times reported above were for a file size of 1KB. To gauge the effect of file size on encryption and decryption we varied the file sizes from 1KB to 57MB for a fixed policy. Figure 6a summarizes the results for a sample 124-leaf

III: Encryption time (in seconds) for policies containing different combinations of equality and relational literals. The values in table cells shows average time in seconds.

		Number of Relational Literals			
		0	2	4	6
Number of Equality Literals	0		1.53s	3.00s	4.49s
	1	0.05s	1.55s	3.05s	4.56s
	2	0.07s	1.57s	3.08s	4.56s
	3	0.09s	1.59s	3.09s	4.60s
	4	0.12s	1.61s	3.12s	4.61s
	5	0.14s	1.65s	3.16s	4.64s
	6	0.17s	1.66s	3.17s	4.63s



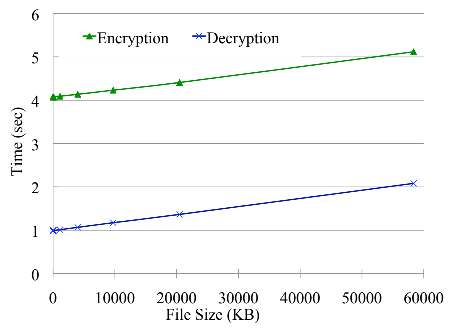
(a) Encrytion Time.



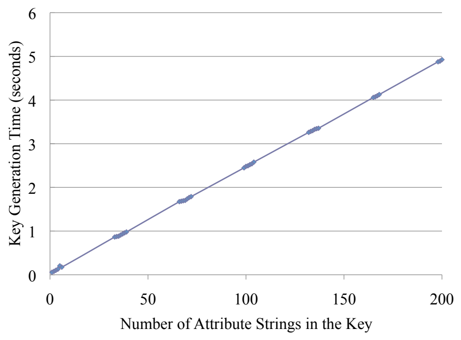
(b) Decryption Time.

5: Encryption and decryption times as a function of the number of leaf nodes in the expanded policy tree.

policy tree. Both encryption and decryption times increased roughly by 20ms for every 1MB increase in file size irrespective of the policy size. However, since in most enterprises e-mail sizes are capped at 10MB, increases in encryption and decryption overheads due to file size can be capped at 200ms.



(a) Encryption and Decryption Time vs. File size.



(b) Key Generation Time vs. No. of Attribute Strings.

6: (a) Encryption and decryption times as a function of file size for a 124-node policy tree. (b) Key generation time as a function of the total number of attribute strings in the key

We measure the time to generate all the necessary keys to decrypt messages. The attributes used for key generation can be classified into the following two types:

—String: Enumerated or boolean attributes that are represented as unique strings (*e.g.*, $\langle$*Position, Faculty*$\rangle$ represented as *"Position_val_Faculty"*).

—Numerical: Numerical attributes (*e.g.*, $\langle$*Salary,* 60000$\rangle$)

Table IV summarizes the key generation times for users with different mixes of attributes. For example, key generation time for a set of attributes containing two numerical attributes and three string attributes took 1.7 seconds. Since each numerical attribute is internally represented using 33 (32 strings, one for each bit of the value, plus 1 unique string representing the value) attribute strings, we can alternatively represent the data as a graph that depicts key-generation time as a function of the number of attribute strings in the key (see Figure 6b).

IV: Key generation time (in seconds) as a function of number of string and numerical attributes. The values in table cells include average time. Empty cells represent combinations for which we did not have enough data.

		Number of String Attributes						
		0	1	2	3	4	5	6
Number of Numerical Attributes	0		0.05s	0.07s	0.10s	0.12s	0.20s	0.17s
	1	0.86s	0.87s	0.88s	0.90s	0.93s	0.95s	0.97s
	2	1.67s	1.68s	1.69s	1.70s	1.73s	1.76s	1.78s
	3	2.44s	2.48s	2.49s	2.52s	2.54s	2.57s	
	4	3.26s	3.28s	3.29s	3.32s	3.34s	3.35s	
	5	4.05s	4.07s	4.09s	4.12s			
	6	4.87s	4.89s	4.92s				

Policy Specialization Path. The performance in this path is translated to the latency a user would see from the time she attempts to log in to the system until the time her specialized policy is revealed to her. This time includes: (1) a database lookup for retrieving a user's attributes and (2) a policy decision time for determining the user's routable attributes.

We studied the policy specialization time with regard to complexity of the address authorization rules and the results capturing the latencies are summarized in Figure 7a. Each rule had 2.5 terms on average and each term 2.5 literals on average. Each of the experiments was averaged over 100 runs. The database used for these experiments was a relational database with 60,000 users, which was filled using the distribution described above. In each of the runs the policy specialization is performed with respect to a user chosen uniformly at random from the database.

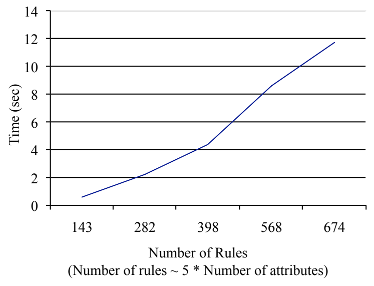
8.5 Analysis

Feasibility Without Security. As shown in Table II, the average latency added to an email message by the ABM system (address resolution latency) without access control is under $250ms$ using a relational database. It is under 6 seconds using an XML database. The implemented system thus can process at least 240 requests per minute using a relational database and 10 requests per minute using an XML database. Though the address resolution takes longer when using an XML database, we can expect that to decrease in the future as XML technology matures.

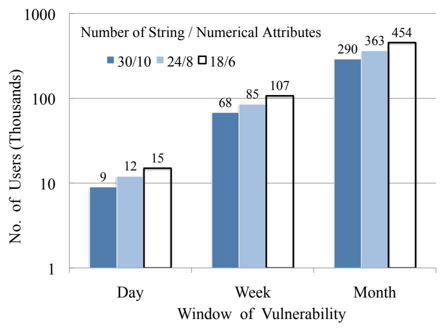
Feasibility With Access Control. As shown in Table II, the average latency added to an email message by the ABM system (address resolution latency) with access control is under $530ms$ when using a relational database and under 6.3 seconds when using an XML database. With access control, the ABM system can process at least 110 requests per minute using a relational database and 9 requests per minute using an XML database. Latency added by access control when using an XML database is higher than when using a relational databases due to the fact that access control involves one database look up apart from user authentication and authorization and on average an XML database look up took $360ms$ more than a relational database lookup. However the percentage overhead added by access control is higher in the relational case than the XML case since latency in the XML case is much higher to begin with.

Feasibility of End-to-end Confidentiality. Based on Table III we see that the total encryption time for a given encryption policy depends on the number and type of literals used in the policy. Specifically, encryption time for a policy can be approximated as $Ax + By$ where A is the number of relational literals in the encryption policy, B is the number of equality literals in the encryption policy, x is the encryption time when using an encryption policy with just one relational literal and y is the encryption time when using an encryption policy with just one equality literal. From Table III the values of x and y are approximately $0.75s$ and $0.05s$ respectively. Relational literals (numerical comparisons) are more expensive when compared to equality literals in terms of encryption time since they are expanded into subtrees with constituent string literals as discussed in Section 5. Note that, literals containing an equality for a numerical attribute are as efficient as those for boolean or enumerated attributes as the CP-ABE implementation we used optimized this case by representing the numerical attribute as a unique string internally in addition to the *bag of bits* representation. From Figure 5a we see that the encryption time is linear in the number of leaf nodes in the expanded policy tree.

Decryption times depend on both the access tree structure, *i.e.*, the encryption policy structure, and the attributes of the user decrypting the message. For a given message, users with different keys will take different amounts of time to decrypt and hence the large variation in decryption times for a given policy size as seen in Figure 5b. In general, we observed decryption times are much smaller than the encryption times for a given message. The decryption times varied from a minimum of about 50ms to a maximum of about 800ms. By taking a conservative approach of adding the worst-case times for encryption and decryption from Table III and Figure 5b, we conclude that for the encryption policies, *i.e.*, for ABM addresses used in our experiments, confidentiality adds at most $4.6 + 0.8 = 5.4$ seconds of latency for a message.



(a) Policy Specialization Time.



(b) No. of Users an Attribute Authority Can Support.

7: (a) Policy specialization time vs. number of rules (b) Number of users (in thousands, logarithmic scale) that an AA can support for a given vulnerability window. 30/10 represents the case that on average each user has 30 string and 10 numerical attributes. The 24/8, and 18/6 data series are defined likewise.

Similar to encryption time, key generation time for a given set of attributes can be approximated as $A.t_1 + B.t_2$ where A is the number of numerical attributes in the set, B is the number of string attributes, t_1 is the key generation time for an attribute set with just one numerical attribute and t_2 is the key generation time for an attribute set with just one string attribute . From Table IV we can estimate t_1 and t_2 to be $800ms$ and $30ms$ respectively. Numerical attributes are more expensive when compared to string attributes in terms of key generation time since numerical attributes are represented using 32 strings (numbers are represented as 32 bit integers in the CP-ABE implementation we used) as opposed to one string for a string attribute. Figure 6b shows that the key generation time is proportional to the total number of attribute strings in a user key.

Given the key generation times, we estimate the number of users a single attribute authority (AA) can support. The AA is responsible for issuing keys to the users. Authors of [Bethencourt et al. 2007] propose adding a numerical attribute to the set of user attributes that denotes the expiration date for the set of attributes. For example, a user will have the numerical attribute $\langle$*Expiry, 31/12/2007*$\rangle$, which denotes that all the user's attributes expire at the end of 2007. The expiration date should be chosen based on the vulnerability window that the application is willing to tolerate for the most sensitive attribute, *i.e.*, the attribute with the smallest tolerable vulnerability window, in the user's attribute set. We simulate the use of this scheme for the key generation process at the AA. When the vulnerability window is a day, attribute private keys issued to users expire within a day after issuing the key. Thus the AA has to re-issue new keys to all users once every day.

Given the parameters we used for populating our databases, on average, each user has 20 attributes from which 10 are numerical and 10 are enumerated. Each user on average possesses 3 values for each of his enumerated attributes (*e.g.*, $\langle$*Courses,* (*CS219, CS230, CS301*)$\rangle$). We represent each enumerated attribute value pair as a unique string attribute for key generation (*e.g.*, *"Courses_val_CS219", "Courses_val_CS230", "Courses_val_CS301"*). So, on average, each user has 10 numerical and 30 string attributes. Given these numbers, we estimate the number of users our AA can support when the vulnerability window ranges from a day to a month. The results are depicted in Figure 7a. For example, for users having 30 string attributes and 10 numerical attributes on an average our instantiation of an AA could support 68000 users with a vulnerability window of one week. While optimizations in current version of CP-ABE toolkit, version 0.7 (we used 0.5), reduce the encryption time for relational literals by almost a third, they increased the key generation time for numerical attributes by almost 50%. Since key generation is centralized at the AA while encryption is distributed among clients we find that the older version is better suited for ABM.

Feasibility of Policy Specialization. As expected, Figure 7a shows that the policy specialization time increases with the number of address authorization rules in the system. The number of authorization rules in the system is directly proportional to the number of attributes in the system. In particular, it is equal to *number of attributes* $\times$ *average number of values (sub-ranges) per attribute*. The number of values/sub-ranges per attribute was randomly drawn between 1 and 10. So we can conclude that the policy specialization time is directly proportional to the number of attributes in the system. Our experiments showed that for policy specialization, database access time remains virtually constant regardless of the number of attributes in the system. This value is about $40ms$ for relational and $400ms$ for XML databases. This is due to the fact that each policy specialization includes a single lookup on the primary key of the database. So the observed increase in the policy specialization is due to the increase in the policy evaluation time, not the database lookup time.

Arguably, the latencies of 12 seconds might be beyond the level of patience of some users and also impact the scalability of the system. However, we have to keep in mind that specialized policy need not be computed every time a user wants to send a message. The ABM system could periodically, say once a week or once a month, compute the specialized policy for all users and cache it. Re-computation between the periods will only be necessary if there is a change in the sending policy or users' attributes.

Scalability. Here, we aim to characterize the scalability of our realization of ABM. First, in terms of access control, it is shown that when using a relational database of 60K users our system can support up to 110 messages per minute. This amounts to about $158,000$ messages per day. It is important to note that ABM system will be used only for a fraction of messages that are mass targeted in the enterprise, and is not involved with traditional emails addressed to individuals or sent via legacy mailing-lists. Therefore, we envision this throughput to be fairly generous. For example, at the University of Illinois at Urbana-Champaign (UIUC) campus which has about $60,000$ users, the email filters process about 1.8 million emails daily [3] of which $650,000$ are marked clean. $158,000$ is about 24.3% of clean messages currently delivered at UIUC. We believe it is reasonable to assume not more than 25% of UIUC's emails would be mass-targeted ABM-type messages. Second, we showed that encryption time for our randomly generated encryption policies is at most 4.63 seconds. While, this time may increase as the encryption policy complexity increases, this operation is distributed, *i.e.*, performed at user client, and thus does not affect scalability adversely. Furthermore, as long as this operation can be performed in the background, after the user clicks to send the message, it will not test the patience of users. Decryption is also a distributed operation and decryption times of at most 800ms are within users' patience limits. Third, in terms of key generation we showed that with a conservative vulnerability window of one week, 68000 users could be supported. Fourth, assuming the worst-case timings, the policy specialization engine can support up to 86000 users per day. Such operation needs to be done for each user once per day and the results would be cached. Therefore, our realization could easily support enterprises with $60,000$ user and we conclude that even with security the performance of our realization of ABM system remains reasonable. However, without better lifetime management schemes for attribute-keys, key generation for ABE has the potential to become a bottleneck.

9. DEPLOYABILITY

ABM architecture proposed in this work is designed with an aim towards deployability. Given a database with user attributes and an e-mail enterprise messaging system, ABM can be deployed by introducing three servers, namely, an ABM server, a PDP and an authentication server, without needing any changes to client software. However, providing end-to-end confidentiality does require encryption and decryption functionality at the clients. This functionality is currently available as a command line tool and could be integrated into user clients as a plug-in. While ABM is deployable from an architectural perspective there are other considerations that effect its deployability and usability. In the rest of the section we discuss some of these issues.

Policy Administration. Specifying and managing an address authorization policy can be a significant burden in the deployment of an ABAC-based ABM system. Even having only one address authorization rule, for each $\langle$*attribute, value*$\rangle$ pair might lead to a unwieldy

[3]It is estimated, that about 7.5 to 15 million spam messages are blocked even before reaching the filters.

set of address authorization rules. Indeed, attributes like *Address* surely cannot be enumerated with a permission for each individual value. In practice, however, attributes do not need a separate authorization rule for every possible value. For example, a rule like $(\mathit{CourseTaken} = x) \leftarrow \mathit{CourseTeaching} = x$ **and** $\mathit{Position} = \mathit{Faculty}$ could cover many individual cases. A rule for a user to send a message to students in a given course might be that the user must be teaching the course. So there is no need to write a separate rule for each $\langle$*CourseTaken, value*$\rangle$ pair as rules for all values of attribute *CourseTaken* follow the same pattern and hence can be written as one rule. The rule is validated at the PDP with the sender's attributes. Some attributes in an enterprise might need only one access rule for each disjoint subset of possible values. For example an attribute like *Age,* whose possible values are from $(17, 120)$, might need a rule only for disjoint sub-ranges like $(17, 30]$, $(30, 65]$ and $(65, 120)$. In general, we observe that any attribute whose values cannot be grouped together in any meaningful way will have only one rule while any attribute that divides the population into disjoint sets might need a rule for every $\langle$*attribute, value*$\rangle$ pair. We analyzed attributes in three units of University of Illinois with the above observations in mind found that only 20% of them need a unique rule for each value while for 50% of them a single rule per attribute is sufficient.

Furthermore, a single enterprise policy administrator does not necessarily need to specify and manage rules for all attributes in an enterprise. Policy administrators in each unit can be responsible for specifying and managing rules for attributes originating from their unit, thereby enabling distributed administration of access rules.

User Interface. End users cannot be expected to write database queries or logical expressions. An effective user interface for composing ABM addresses is crucial for the ABM system to be adopted. Similarly, policy administrators will benefit from a user interface for specifying address authorization rules. Though we do not address these needs in this work, user interfaces that closely satisfy the requirements are those found in web directories and catalog searches. Moreover, recent advances in natural language query interfaces such as NaLix [Li et al. 2006; 2007], that enable translation of queries in English into queries in XQuery can further improve the usability of ABM system.

Key Management. While CP-ABE scheme provides end-to-end confidentiality even when the sender does not know the recipients, like IBE schemes, it suffers from lack of a revocation mechanism. Recently, Boldyreva *et al.* , [Boldyreva et al. 2008] showed how to integrate a binary key-tree with Key-Policy ABE (KP-ABE), a flavor of ABE where policies are associated with keys and attributes with data, to obtain a KP-ABE scheme with an efficient key update process. However this technique does not extend to CP-ABE.

In this work we adopt an approach suggested in [Bethencourt et al. 2007] and append an expiration date to the attribute keys issued. However, as shown in Section 8 CP-ABE does not scale for smaller vulnerability windows, such as when keys have to be re-issued every day. ABE is a new paradigm and we expect that ABE schemes that perform better will emerge, but there is at least one observation that can be used to improve the situation immediately: it is often reasonable to let the delivery policy be a (possibly small) subset of the encryption policy. In particular, some attributes in a system need not have associated keys and are only used for targeting messages. For example, a message regarding insurance policy changes targeted for faculty who are above the age of 60 could sufficiently be encrypted without using age as an attribute. A more compelling example is email sent to the primary care physician and ordering physician of a given patient. Since medical information systems often allow any doctor access to the records of any patient, it is essential that such a message is encrypted only under the doctor attribute key. Reducing the number of attribute keys each user needs and will improve the performance of the AA. If only 80% of attributes are associated with keys, average number of attributes of a user for which keys need to be issued reduces from $30/10$ to $24/8$ in Figure 7b and the number of users an AA can support increases from 68000 to 85000 for a tolerable vulnerability window of one week. The idea of letting the delivery policy be a subset of the encryption policy requires extra mechanisms in policy specification, however, so a systematic approach would probably require some automated support such as a rule set for deriving encryption keys from addresses.

Another issue that needs to be considered is the key management and storage at the end user. End users might have to store their old attribute keys until all the messages in their mailbox encrypted with those keys are deleted. This could mean a lot of key storage and management. To estimate this storage we consider a scenario where users need to store keys to be able to access messages that are up to a year old. If the tolerable vulnerability window is one day then users have to store 365 keys in the scenario above. On the other hand if the tolerable vulnerability window is one month then the users have to store only 12 keys. The size of a CP-ABE key with 30 string and 10 numerical attributes is found to be 110KB with the version of the CP-ABE toolkit we used. Thus, the key storage needed for such keys reduces from 39.2MB to 5.6MB to 1.3MB, as the tolerable vulnerability window increases from one day to one week to one month respectively. Similarly, for CP-ABE keys with 18 string and 6 numerical attributes, whose size is found to be 66KB, the key storage needed reduces from 23.5MB to 3.4MB to 0.8MB as the tolerable vulnerability window increases from one day to one week to one month respectively. While storage is very cheap on desktops, secure storage and its management may not be so. One way to avoid this storage is to re-encrypt all messages using a password after they are read. Since the messages are encrypted using hybrid encryption, the computational cost of re-encrypting is one symmetric encryption, of a 128 bit message encryption key, which is very low.

Privacy Considerations. Privacy concerns posed by a system like ABM in different environments could be of varying magnitude. A user's privacy is said to be violated whenever an unauthorized entity associates the user's identity with his attributes. In an ABM system entities may be allowed to target messages using an ABM address even though they might not otherwise be authorized to associate recipient identities with the attributes in the ABM address. Thus in most scenarios the list of recipients needs to be kept secret from the sender. Thus the current design and implementation are limited to do-not-reply notifications and cannot be used for discussions unless the sender is authorized to know the recipient identities.

Another privacy related questions is whether the recipients of an ABM message should be allowed to know the ABM address used to target the message. Letting the recipients of a message know the ABM address used to target the message might violate privacy if they could learn who else received the message. Currently, when we employ CP-ABE to protect confidentiality of the message, the ABM address is revealed to the recipient as the encrypted message contains the policy used to encrypt the message which is the same as the ABM address. ABM without an option for encryption can hide the policy from the recipients. There has been work on hiding policies both for two-party interactions [Ateniese et al. 2007] and for CP-ABE schemes [Katz et al. 2008; Nishide et al. 2008]. However none of the existing schemes for hiding policy in CP-ABE are practical. The scheme of [Nishide et al. 2008] only supports policies with single AND gates. The size of policies is bounded in the scheme of [Katz et al. 2008], *i.e.*, it is a bounded ciphertext scheme. For a situation where recipients need to know the ABM address used to target the message but should remain private from each other the ABM server can use blind-carbon-copy (bcc) option while forwarding the message. However, this also works only for do-not-reply notifications. Furthermore there might be leakage of user attribute information outside the system, for example, at a water cooler conversation. Thus more work needs to be done to address privacy concerns before ABM system can be used for discussions by a wide number of users (a limited number of privileged users may be able to use the current system for discussions).

10. RELATED WORK

We discuss five areas of related work: targeted messaging systems, secure role-based messaging, WSEmail, attribute based access control and attribute based encryption.

Perhaps the technology most similar to ABM was that described in a patent application [Hofmann and Hurley 1999] that proposed Attribute-Based Addressing (ABA) of messages. A key difference between ABA and our independently proposed ABM work is that we address security challenges for access control and end-to-end confidentiality. Another technology similar to ABM arises in Customer Relationship Management (CRM) systems. CRMs help enterprises target customers by isolating specific buying patterns and using this to customize the communication with them. The key difference between CRMs and ABM is that in CRMs the communication is from the enterprise to the customer group and so there is no need for access control. Where as in ABM messages are sent by users to other users after access is determined by the attributes of the sender. In other words, CRM generally uses a monolithic permission given to the owner of the system, whereas ABM provides diverse permissions to a broad user group. Traditional list servers also provide a way to send email messages to a certain group of people. One can imagine driving membership in lists from a database of attributes to provide a form of ABM. For example, SendMail (a popular MTA) can be integrated with LDAP but it lacks a mechanism to control the use of such mailing lists. A key difference between ABM and list servers is the fact that ABM has the potential to route on 'involuntary' attributes of recipients rather than relying solely or mainly on voluntary subscriptions. A good potential use of ABM is to provide a way for users to subscribe to lists automatically and voluntarily by collecting a user profile of interests.

Secure role-based messaging uses RBAC for authorizing access to sensitive email content [Chadwick et al. 2004; Mont et al. 2003]. In this area [Chadwick et al. 2004] allows users to send messages to a given role identified by a special mailbox. Users that are assigned to that role can then provide their role membership credentials and access the email. Message security is provided by traditional public-key cryptography and a trusted entity decrypts the message for the user after ensuring his role-membership and thus is not end-to-end protection. Furthermore, this work does not consider targeting messages to a conjunction of multiple roles. Using a slightly different approach [Mont et al. 2003] employs Identity-Based Encryption (IBE) for encrypting messages to recipients. On receipt of every message, recipients must interact with a trusted authority which verifies whether the recipient satisfies the encryption policy before releasing the message encryption key, in order to decrypt the message. The only exception is when the user has fetched the key previously for the same policy. Furthermore, the sender has to send each message individually to all the receivers that he thinks might satisfy his policy. These two approaches further differ from ABM by focusing only on the access control rules for recipients, whereas we focused on access control rules for both senders and recipients. Of course, they also differ in the use of roles rather than attributes as a foundation for policies.

WSEmail is the idea of building messaging systems over a web services foundation. A prototype [Lux et al. 2005] of such a system demonstrated messages that could be routed with addresses that are determined dynamically as the message passes through WSEmail MTAs. However, this system does not explore the idea of routing based recipient attributes. A WSEmail-based design [Afandi et al. 2006] shows how to adapt to recipient policies as part of messaging, but this design does not deal with multiple recipients. Secure RSS systems such as Shibboleth RSS [Gioachin et al. 2007] may use attributes to decide who receives an RSS feed. This concept has a number of things in common with ABM, but focuses on subscription access rules in an inter-domain context.

While the concept of ABAC has been around (introduced as early as 1996 in ISO 10181-3) it has gained prominence in research literature with its use in trust negotiation and credential-based access control in a distributed system with multiple administrative domains [Bonatti and Samarati 2002; Wang et al. 2004; Yuan and Tong 2005; Yu et al. 2003]. Our ABM study shows how ABAC is also valuable for enterprise applications and uses attributes assimilated from backend databases. Also, access control in ABM is different from access control in traditional systems and services because the resource (ı.e., an ABM address) is somewhat different than a resource in these traditional systems. Most of the research on ABAC provides insights on theory and expressiveness for applications but does not discuss implementation of the proposed designs and practical studies on applications. Some works [Lorch et al. 2003; Yuan and Tong 2005; Yu et al. 2003] have led to implementations, but no performance data is available. At the same time performance of access control systems in becoming important in recent application such as location based access control [Borders et al. 2005]. In this work we demonstrate the practicality of ABAC for a novel enterprise application (ABM) in a mid-size enterprise as evidenced by our performance evaluation. In a related work [Olson et al. 2008], Olson *et al.* provide a formal framework for using attributes from a database to control access to the database itself.

While some of the roots of Attribute-Based Encryption (ABE) can be traced back to Identity-Based Encryption [Boneh and Franklin 2003; Cocks 2001], which could be considered a special case of ABE, the first ABE scheme was proposed in [Sahai and Waters 2005] and supported only encryption policies with a single threshold gate. Goyal *et al.*, [Goyal et al. 2006] first defined the two complimentary forms of ABE, namely, Key Policy ABE (KP-ABE) and Ciphertext Policy ABE (CP-ABE), and provided a construction for KP-ABE. Bethencourt *et al.*, [Bethencourt et al. 2007] gave the first construction for a CP-ABE scheme albeit in the generic group model. Both these schemes supported monotonic boolean encryption policies. Many ABE schemes with varying properties have been proposed since then. For example, schemes that supported non-monotonic boolean encryption policies (*e.g.*, [Ostrovsky et al. 2007]), schemes that supported policy privacy (*e.g.*, [Nishide et al. 2008; Katz et al. 2008]) and schemes that supported multiple attribute authorities (*e.g.*, [Muller et al. 2009; Chase and Chow 2009]) were all proposed.

However, at the time of this work, only ABE schemes of [Sahai and Waters 2005; Bethencourt et al. 2007] have been implemented. Pirretti *et al.* [Pirretti et al. 2006] implemented the ABE scheme proposed in [Sahai and Waters 2005] and proposed a security architecture based on it. Traynor *et al.*, [Traynor et al. 2008] demonstrated that this ABE scheme can be used to secure IPtv media that is being broadcast to millions of viewers. In this work we demonstrate the use of CP-ABE scheme proposed and implemented in [Bethencourt et al. 2007] for a novel application and demonstrate its practicality. Since this work, [Waters 2008] has proposed a more efficient CP-ABE scheme with security in the standard model, but at the cost of imposing some restrictions on flexibility of policies. In [Bobba et al. 2009], an "attribute-set based encryption" scheme is developed which admits significant improvments in handling attribute revocations in a system like ours.

One could imagine employing proxy re-encryption techniques such as those in [Mambo and Okamoto 1997; Khurana et al. 2006] at the ABM server, where senders encrypt their messages with a well know public key, say ABM server's public key, and the ABM servers re-encrypt (using proxy re-encryption) the message to all the users who are valid recipients. This approach incurs per-message per-user overheads at the ABM server which adversely impacts the throughput of the ABM system. Similarly, one could imagine an approach where the senders associate an encryption policy with their messages and encrypt them with a well known system public key. The ABM server simply forwards the encrypted messages to valid recipients. The recipients then contact another server, what we call a Key Distribution Center (KDC), to obtain the symmetric key used to encrypt the message. The KDC verifies that the recipient indeed satisfies the encryption policy before releasing the message key. Examples of technology that can be used to implement this approach are traditional IBE [Boneh and Franklin 2003; Cocks 2001] as it was done in [Mont et al. 2003] and the policy based encryption system of [Bobba et al. 2009]. This approach incurs per-message per-user overheads at the KDC and forces recipients to contact the KDC on receipt of every message. However, it still might do better than CP-ABE in certain situations owing to the expensive key generation and refresh operations when using CP-ABE (as shown in Section 8). Specifically, it might be able to support more users than CP-ABE as long as the recipient message volume, that is, the number of messages times the average number of recipients per message, is low. But it becomes unattractive when recipient message volume is high and is therefore not a good fit for ABM.

11. CONCLUSION

We proposed an architecture, 1) with a simple and manageable access control model for ABM based on ABAC that accommodates a useful collection of ABM applications and 2) with ABE to provide end-to-end confidentiality in ABM. We have shown that this architecture can be implemented efficiently for mid-size enterprises and we have given a profile of policy parameters that affect its efficiency.

There are a number of interesting questions and open opportunities for ABM. Three of these will particularly interest us for future research: inter-domain operation of ABM, more expressive ABAC policy languages and key management for ABE. While we have shown how to architect and deploy ABM for enterprises, it is much trickier to do this when multiple enterprises are involved. For example, suppose we wish to send a message to all of the doctors in a given county. This cannot be done with a single database or even the collection of databases of a single enterprise. There is some need to map the attribute ‘doctor’ across multiple domains. This problem arises with virtually any inter domain authorization challenge so the problem is only illustrative, but it is perhaps more tractable for ABM than for inter domain authorization in general. Clearly some techniques are required to map attributes. We have a design for such a system assuming such a mapping is possible, but it needs to be developed and studied in the way we have approached the enterprise systems in this paper. Our ABAC address authorization policy language (implemented as a subset of XACML) is rudimentary. We choose it because it was clearly useful and yielded non-trivial questions about processing and performance. However, one can certainly imagine ABAC based ABM systems benefiting from a more theoretical analysis of policy language expressibility such as that undertaken by [Li et al. 2002; Wang et al. 2004] for distributed systems. At the same time, it is not clear how complex a policy language should be; perhaps expressiveness is less important than the ease of maintaining policies. After all, existing systems do not offer ABM at all, so even basic functions are a step forward. Complex policies that lead to unintentional user errors would dampen enthusiasm for deployment. Nevertheless, there are a variety of interesting theoretical questions that can be considered in this area. Lack of revocation support in ABE calls for the design of schemes to manage the life cycle of ABE keys. While schemes proposed in [Pirretti et al. 2006; Bethencourt et al. 2007; Boldyreva et al. 2008] are a step in this direction there is a need to further improve them.

Acknowledgements. We would like to thank Noam Artz, Mike Berry, John Bethencourt, Brent Waters and anonymous reviewers for their helpful comments. We would also like to thank Tracy Tolliver, Mark Nye and Bradley Horn for providing us with UIUC e-mail processing statistics. This work was supported in part by NSF CNS 07-16626, NSF CNS 07-16421, NSF CNS 05-24695, ONR N00014-08-1-0248, NSF CNS 05-24516, DHS 2006-CS-001-000001, ONR N00014-04-1-0562, ONR N00014-06-1-1108, N00014-07-1-1173 and grants from the MacArthur Foundation, the Sohaib and Sara Abbasi Fellowship, and Boeing Corporation. The views expressed are those of the authors only.

REFERENCES

AFANDI, R., ZHANG, J., HAFIZ, M., AND GUNTER, C. A. 2006. AMPol: Adaptive messaging policy. In *ECOWS ’06: Proceedings of the European Conference on Web Services*. IEEE, Zurich, Switzerland, 53–64.

ATENIESE, G., KIRSCH, J., AND BLANTON, M. 2007. Secret handshakes with dynamic and fuzzy matching. In *NDSS ’07: Proceedings of The 14th Annual Network and Distributed System Security Symposium*. The Internet Society, San Diego, California.

BETHENCOURT, J., SAHAI, A., AND WATERS, B. 2007. Ciphertext-policy attribute-based encryption. In *SP ’07: Proceedings of the IEEE Symposium on Security and Privacy*. IEEE, Berkeley/Oakland, California, 321–334.

BOBBA, R., KHURANA, H., ALTURKI, M., AND ASHRAF, F. 2009. PBES: A policy based encryption system with application to data sharing in the power grid. In *ASIACCS ’09: Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*. ACM, New York, NY, USA.

BOBBA, R., KHURANA, H., AND PRABHAKARAN, M. 2009. Attribute-sets: A practically motivated enhancement to attribute-based encryption. In *ESORICS*, M. Backes and P. Ning, Eds. Lecture Notes in Computer Science, vol. 5789. Springer, 587–604.

BOLDYREVA, A., GOYAL, V., AND KUMAR, V. 2008. Identity-based encryption with efficient revocation. In *CCS ’08: Proceedings of the 15th ACM Conference on Computer and Communications Security*. ACM, New York, NY, USA, 417–426.

BONATTI, P. A. AND SAMARATI, P. 2002. A uniform framework for regulating service access and information release on the web. *Journal of Compututer Security 10,* 3, 241–271.

BONEH, D. AND FRANKLIN, M. 2003. Identity-based encryption from the weil pairing. *SIAM J. Comput. 32,* 3, 586–615.

BORDERS, K., ZHAO, X., AND PRAKASH, A. 2005. CPOL: High-performance policy evaluation. In *CCS '05: Proceedings of the 12th ACM Conference on Computer and Communications Security*. ACM, Alexandria, Virginia, 147–157.

CHADWICK, D., LUNT, G., AND ZHAO, G. 2004. Secure role-based messaging. In *CMS '04: Eighth IFIP TC-6 TC-11 Conference on Communications and Multimedia Security*. Springer, Windermere, UK, 263–275.

CHASE, M. AND CHOW, S. S. 2009. Improving privacy and security in multi-authority attribute-based encryption. In *CCS '09: Proceedings of the 16th ACM conference on Computer and communications security*. ACM, New York, NY, USA, 121–130.

COCKS, C. 2001. An identity based encryption scheme based on quadratic residues. In *Proceedings of the 8th IMA International Conference on Cryptography and Coding*, B. Honary, Ed. LNCS, vol. 2260. Springer-Verlag, London, UK, 360–363.

DAMIANI, E., DI VIMERCATI, S. D. C., AND SAMARATI, P. 2005. New paradigms for access control in open environments. In *5th IEEE International Symposium on Signal Processing and Information*. IEEE, Athens, Greece.

FERRAIOLO, D., KUHN, D., AND R.CHANDRAMOULI. 2003. *Role Based Access Control*. Artech House, Norwood, MA, USA.

FUJISAKI, E. AND OKAMOTO, T. 1999. Secure integration of asymmetric and symmetric encryption schemes. In *CRYPTO '99: Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*. Springer, London, UK, 537–554.

GIOACHIN, F., SHANKESI, R., MAY, M. J., GUNTER, C. A., AND SHIN, W. 2007. Emergency alerts as RSS feeds with interdomain authorization. In *ICIMP '07: Proceedings of the Second International Conference on Internet Monitoring and Protection*. IEEE, Washington, DC, USA, 13.

GOYAL, V., PANDEY, O., SAHAI, A., AND WATERS, B. 2006. Attribute-based encryption for fine-grained access control of encrypted data. In *CCS '06: Proceedings of the 13th ACM Conference on Computer and Communications Security*. ACM, Alexandria, Virginia, 89–98.

HOFMANN, W. D. AND HURLEY, P. E. 1999. Method and apparatus for attribute-based addressing of messages in a networked system. Tech. Rep. WO/1999/039271, Aveo Inc. May.

KATZ, J., SAHAI, A., AND WATERS, B. 2008. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008.*, N. P. Smart, Ed. LNCS, vol. 4965. Springer, Berlin, Heidelberg, 146–162.

KHURANA, H., HEO, J., AND PANT, M. 2006. From proxy encryption primitives to a deployable secure-mailing-list solution. In *Information and Communications Security, 8th International Conference, ICICS 2006, Raleigh, NC, USA, December 4-7, 2006, Proceedings*, P. Ning, S. Qing, and N. Li, Eds. LNCS, vol. 4307. Springer, Berlin, Heidelberg, 260–281.

LI, N., MITCHELL, J. C., AND WINSBOROUGH, W. H. 2002. Design of a role-based trust-management framework. In *SP '02: Proceedings of the IEEE Symposium on Security and Privacy*. IEEE, Oakland, California, 114.

LI, Y., YANG, H., AND JAGADISH, H. V. 2006. Constructing a generic natural language interface for an XML database. In *EDBT*, Y. E. Ioannidis, M. H. Scholl, J. W. Schmidt, F. Matthes, M. Hatzopoulos, K. Böhm, A. Kemper, T. Grust, and C. Böhm, Eds. LNCS, vol. 3896. Springer, Berlin, Heidelberg, 737–754.

LI, Y., YANG, H., AND JAGADISH, H. V. 2007. NaLIX: A generic natural language search environment for XML data. *ACM Transactions on Database Systems 32,* 4, 30.

LORCH, M., PROCTOR, S., LEPRO, R., KAFURA, D., AND SHAH, S. 2003. First experiences using XACML for access control in distributed systems. In *XMLSEC '03: Proceedings of the ACM Workshop on XML Security*. ACM, Fairfax, Virginia, 25–37.

LUX, K. D., MAY, M. J., BHATTAD, N. L., AND GUNTER, C. A. 2005. WSEmail: Secure internet messaging based on web services. In *ICWS '05: Proceedings of the IEEE International Conference on Web Services*. Vol. 0. IEEE, Orlando, Florida, 75–82.

MAMBO, M. AND OKAMOTO, E. 1997. Proxy cryptosystem: Delegation of the power to decrypt ciphertexts. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences E80,* A(1), 54–63.

MONT, M. C., BRAMHALL, P., AND HARRISON, K. 2003. A flexible role-based secure messaging service: Exploiting IBE technology for privacy in health care. In *DEXA '03: Proceedings of the 14th International Workshop on Database and Expert Systems Applications*. IEEE, Prague, Czech Republic, 432.

MULLER, S., KATZENBEISSER, S., AND ECKERT, C. 2009. On multi-authority ciphertext-policy attribute-based encryption. *Bulletin of the Korean Mathematical Society 46,* 4 (July), 803–819.

MYERS, J. 1999. SMTP service extension for authentication. RFC 2554, Internet Engineering Task Force. Mar.

NISHIDE, T., YONEYAMA, K., AND OHTA, K. 2008. Attribute-based encryption with partially hidden encryptor-specified access structures. In *Applied Cryptography and Network Security, 6th International Conference, ACNS 2008, New York, NY, USA, June 3-6, 2008. Proceedings*, S. M. Bellovin, R. Gennaro, A. D. Keromytis, and M. Yung, Eds. LNCS, vol. 5037. Springer, Berlin, Heidelberg, 111–129.

OLSON, L. E., GUNTER, C. A., AND MADHUSUDAN, P. 2008. A formal framework for reflective database access control policies. In *CCS '08: Proceedings of the 15th ACM conference on Computer and communications security*. ACM, New York, NY, USA, 289–298.

OSTROVSKY, R., SAHAI, A., AND WATERS, B. 2007. Attribute-based encryption with non-monotonic access structures. In *CCS '07: Proceedings of the 14th ACM Conference on Computer and Communications Security*. ACM, Alexandria, Virginia, 195–203.

PIRRETTI, M., TRAYNOR, P., MCDANIEL, P., AND WATERS, B. 2006. Secure attribute-based systems. In *CCS '06: Proceedings of the 13th ACM Conference on Computer and Communications Security*. ACM, Alexandria, Virginia, 99–112.

RAMSDELL, E. 2004a. Secure/Multipurpose internet mail extensions (S/MIME) version 3.1 certificate handling. RFC 3850, Internet Engineering Task Force. July.

RAMSDELL, E. 2004b. Secure/Multipurpose internet mail extensions (S/MIME) version 3.1 message specification. RFC 3851, Internet Engineering Task Force. July.

SAHAI, A. AND WATERS, B. 2005. Fuzzy identity-based encryption. In *Advances in Cryptology - EUROCRYPT '05, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. LNCS, vol. 3494. Springer, Aarhus, Denmark, 457–473.

SALTZER, J. AND SCHROEDER, M. Sept. 1975. The protection of information in computer systems. *IEEE 63,* 9, 1278–1308.

TRAYNOR, P., BUTLER, K., ENCK, W., AND MCDANIEL, P. 2008. Realizing massive-scale conditional access systems through attribute-based cryptosystems. In *NDSS '08: Proceedings of the 15th Annual Network and Distributed System Security Symposium*. Internet Society, San Diego, CA.

WANG, L., WIJESEKERA, D., AND JAJODIA, S. 2004. A logic-based framework for attribute based access control. In *FMSE '04: Proceedings of the ACM Workshop on Formal Methods in Security Engineering*. ACM, Washington, DC, 45–55.

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

YU, T., WINSLETT, M., AND SEAMONS, K. E. 2003. Supporting structured credentials and sensitive policies through interoperable strategies for automated trust negotiation. *ACM Transactions on Information System Security 6,* 1, 1–42.

YUAN, E. AND TONG, J. 2005. Attribute based access control (ABAC) for web services. In *ICWS '05: Proceedings of the IEEE International Conference on Web Services*. IEEE, Orlando, Florida, 561–569.