CERIAS Tech Report 2009-27
A Privacy-Preserving Approach to Policy-Based Content Dissemination
by Ning Shang, Mohamed Nabeel, Federica Paci, Elisa Bertino
Center for Education and Research
Information Assurance and Security
Purdue University, West Lafayette, IN 47907-2086

A Privacy-Preserving Approach to Policy-Based Content Dissemination (Full Paper)

Ning Shang, Mohamed Nabeel, Federica Paci, Elisa Bertino
Purdue University,
West Lafayette, Indiana, USA
{nshang, nabeel, paci, bertino}@cs.purdue.edu

Abstract

We propose a novel scheme for selective distribution of content, encoded as documents, that preserves the privacy of the users to whom the documents are delivered and is based on an efficient and novel group key management scheme.

Our document broadcasting approach is based on access control policies specifying which users can access which documents, or subdocuments. Based on such policies, a broadcast document is segmented into multiple subdocuments, each encrypted with a different key. In line with modern attribute-based access control, policies are specified against identity attributes of users. However our broadcasting approach is privacy-preserving in that users are granted access to a specific document, or subdocument, according to the policies without the need of providing in clear information about their identity attributes to the document publisher. Under our approach, not only does the document publisher not learn the values of the identity attributes of users, but it also does not learn which policy conditions are verified by which users, thus inferences about the values of identity attributes are prevented. Moreover, our key management scheme on which the proposed broadcasting approach is based is efficient in that it does not require to send the decryption keys to the users along with the encrypted document. Users are able to reconstruct the keys to decrypt the authorized portions of a document based on subscription information they have received from the document publisher. The scheme also efficiently handles new subscription of users and revocation of subscriptions. Please note that this is an improved and extended version of our previous report [1].

Index Terms

Privacy, Identity, Document Broadcast, Policy, Key Management, Access Control

I. Introduction

The Internet and the Web have enabled tools and systems for quickly disseminating data, by posting on Web sites or broadcasting, to user communities in a large variety of application domains and for different purposes. However, because of legal requirements, organizational policies, or commercial reasons, selective access to data should be enforced in order to protect data from unauthorized accesses. Modern access control models, like XACML [2], allows one to specify access control policies that are expressed in terms of conditions concerning the protected objects against properties of subjects, referred to as *identity attributes*, characterizing the users accessing the protected data. Examples of identity attributes include the role that a user has in

his/her[1] organization, the age, and the country of origin. A user thus verifies a given access control policy, if its identity attributes verify the conditions of the policy. The use of such an approach is crucial to simplify access control administration and support high-level policies closer to organizational policies and is in line with current initiatives for digital identity management [3], [4], [5], [6], [7]. An approach to support fine-grained selective attribute-based access control when posting or broadcasting contents is to encrypt each content portion to which the same access control policy (or set of policies) applies with the same key, and then distributing this key to each user, satisfying the policy (or any policy in the set) associated with the content portion. A user would thus receive all the keys for the content portions the user can access [8], [9].

A critical issue in such a context is represented by the fact that very often identity attributes encode privacy-sensitive information and this information has to be protected, even from the party distributing the contents. Privacy is considered a key requirement in all solutions and initiatives for digital identity management. It is important to notice that because of the problem of insider threats, today recognized as a major source of data theft and privacy breaches, identity attributes should still be strongly protected even if the party distributing the contents and the content recipients belong to the same organization. To date the problem of disseminating contents to user groups by enforcing attribute-based access control while at the same time assuring the privacy of the user identity attributes has not been addressed.

To this extent, it is worth noting that a simplistic approach in which the content publisher encrypts different portions of a document with different keys, and then directly sends keys to corresponding users has some major drawbacks with respect to user privacy and key management. On one hand, user private information, encoded in the user identity attributes, is not protected in the simplistic approach. On the other hand, such a simplistic key management scheme does not scale well as the number of users becomes large and when multiple keys need to be distributed to multiple users. The goal of this paper is to develop an approach which does not have these shortcomings.

In the paper we develop an attribute-based access control mechanism whereby a user is able to decrypt the disseminated contents if and only if its identity attributes satisfy the content

[1]We shall use “it” and “its” to refer to a user and the user’s ownership, respectively, in the rest of the paper.

provider's policies, whereas the content provider learns nothing about user's identity attributes. The mechanism is fine-grained in that different policies can be associated with different content portions. A user can derive only the encryption keys associated with the portions the user is entitled to access. A crucial aspect of such an approach is key management. In order to acheive this goal, we propose a novel flexible key management scheme and integrate it with techniques for oblivious transfer of information. The proposed key management scheme satisfies the following requirements [10]:

- **Minimal trust** requires the key management scheme to place trust on a small number of entities.
- **Key indistinguishability** requires that for given public information, any element in the key space has the same probability of being the real key.
- **Key independence** requires that a leak of one key does not compromise other keys.
- **Forward secrecy** requires that a user who left the group should not be able to access any future keys.
- **Backward secrecy** requires that a newly joining user should not be able to access any old keys.
- **Collusion resistance** requires that colluding users can not obtain keys which they are not allowed to obtain individually.[2]
- **Bandwidth overhead** requires that the rekey of the group should not include a high number of transmitted messages.
- **Computational costs** should be acceptable at both the key server and users.
- **Storage requirements** should be minimal; high storage of keys or relevant data need be avoided in the key management scheme.

In summary, we propose a new protocol for content dissemination which assures policy-based access control, preserves users' privacy and satisfies all the above requirements. We formally analyze the protocol and carry on an extensive experimental evaluation to assess its efficiency and scalability. In the rest of the paper we will use the term documents to refer to contents and to subdocuments to refer to content portions.

The rest of the paper is organized as follows. Section II discusses the related work. Section III provides an overview of our scheme. Section IV introduces the basic notions on which our

[2]We assume the adversaries have access to any public information and information that users who left the group hold.

approach is based. Section V presents our new scheme for document broadcasting, and Section VI analyzes the our scheme in terms of security and efficiency. Section VII presents the result of our experiments. In Section VIII we further discuss issues such concerning scalability and optimization of the proposed scheme. Section IX concludes the paper and outlines future research directions.

II. Related Work

Approaches closely related to our work have been investigated in three different areas: selective publication and broadcast of documents, attribute-based security, and group key management.

The database and security communities have carried out extensive research concerning techniques for the selective dissemination of documents based on access control policies [8], [9], [11]. These approaches fall in the following two categories.

1) Encryption of different subdocuments with different keys, which are provided to users at the registration phase, and broadcasting the encrypted subdocuments to all users [8], [9].
2) Selective multicast of different subdocuments to different user groups [11], where all subdocuments are encrypted with one symmetric encryption key.

The latter approaches assume that the users are honest and do not try to access the subdocuments to which they do not have access authorization. Therefore, these approaches provide neither backward nor forward key secrecy. In the former approaches, users are able to decrypt the subdocuments for which they have the keys. However, such approaches require all [8] or some [9] keys be distributed in advance during user registration phase. This requirement makes it difficult to assure forward and backward key secrecy when user groups are dynamic with frequent join and leave operations. Further, the rekey process is not transparent, thus shifting the burden of acquiring new keys on existing users when others leave or join. In contrast, our approach makes rekey transparent to users by not distributing actual keys during the registration phase.

Attribute-Based Encryption (ABE) [12] is another approach for implementing encryption-based access control to documents. Under such an approach, users are able to decrypt subdocuments if they satisfy certain policies. ABE has two variations: associating encrypted documents with attributes and user keys with policies [13]; associating user keys with attributes and encrypted documents with policies [12]. In either cases the cost of key management is minimized

by using attributes that can be associated with users. However, these approaches require the attributes considered in the policies to be sent in clear. Having such clear texts reveals sensitive information about users during both registration and document distribution phases. In contrast, our approach preserves user privacy in both phases, in that users are not required to reveal the values of their identity attributes to the content distributor.

Group Key Management (GKM) is a widely investigated topic in the context of group-oriented multicast applications [10], [14]. Early work on GKM relied on a key server to share a secret with users to distribute keys to decrypt documents [15], [16]. Such approaches suffer from the drawback of sending $O(n)$ rekey information, where n is the number of users, in the event of join or leave to provide forward and backward secrecy. Hierarchical key management schemes [17], [18], where the key server hierarchically establishes secure channels with different sub-groups instead of with individual users, were introduced to reduce this overhead. However, they only reduce the size of the rekey information to $O(\log n)$, and furthermore each user needs to manage at worst $O(\log n)$ hierarchically organized redundant keys. Similar to the spirit of our approach, there have been efforts to make rekey a one-off process [19], [14]. The secure lock approach [19] based on the Chinese Remainder Theorem (CRT) performs a single broadcast to rekey. However, the proposed approach is inefficient for large n values as it requires performing CRT calculation involving n congruences each time a new document is sent. The access control polynomial approach [14] encodes secrets given to users at registration phase in a special polynomial of order at least n in such a way that users can derive the secret key from this polynomial. The special polynomials used in this approach represent only a small subset of domain of all the polynomials of order n, and the security of the approach is neither fully analyzed nor proven.

III. Overview

Our scheme for selective distribution of documents involves four main entities: the *Publisher* (Pub), the users referred to as *Subscribers* (Subs)[3], the *Identity Providers* (IdPs), and the *Identity Manager* (IdMgr). It is based on three main phases (see Figure 1): *identity token issuance*, *identity token registration*, and *document dissemination*.

[3]In what follows we use the term Sub; however in practice the steps are carried out by the client software transparently to the actual end user.

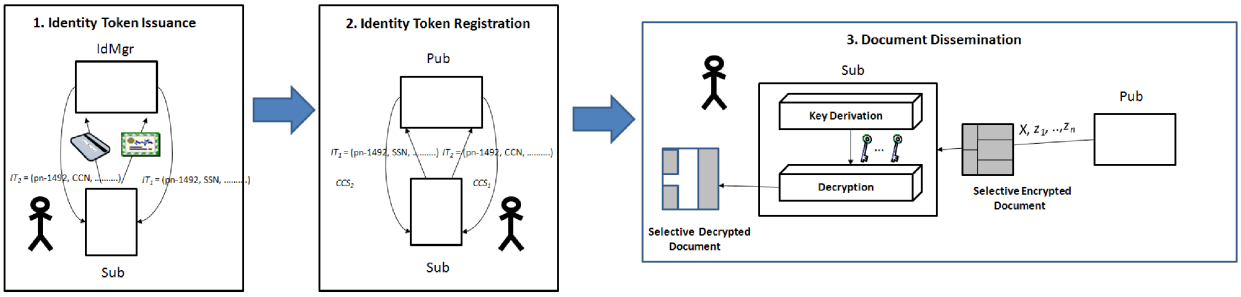

Fig. 1. Overview of our content dissemination scheme

1) Identity token issuance

IdPs issue certified identity attributes to Subs. Subs present their identity attributes to the IdMgr which is a trusted third party that issues *identity tokens* to Subs. An identity token is a Sub's identity in a specified electronic format in which the involved identity attribute value is represented by a semantically secure cryptographic *commitment*.[4] Identity tokens are used by Subs during the registration phase.

Note that the main functionality of the IdMgr is to generate a uniform electronic format for an identity attribute value, in the form of an "identity token." We adopt this notion for ease of the presentation. In general, a Sub can engage into a protocol with the IdPs, and request identity tokens directly from the IdPs, without needing to use the IdMgr. In this sense, our approach is suitable for both open and closed environments.

2) Identity token registration

In order to be able to decrypt the documents that will be received from the Pub, Subs have to register at the Pub. During the registration, each Sub presents its identity tokens and receives from the Pub a *conditional subscription secret* (CSS) for each identity attribute name in the Pub's access control policy condition matching the Sub's identity token tag. CSSs are used by Subs to derive the keys to decrypt the subdocuments for which they satisfy the access control policy and managed by the proposed GKM scheme. The Pub delivers the CSSs to the Subs using a privacy-preserving approach based on carrying out OCBE protocols [20] with the Subs.

[4]A cryptographic commitment allows a user to commit to a value while keeping it hidden and preserving the user's ability to reveal the committed value later.

The OCBE protocols ensure that a Sub can obtain a CSS if and only if the Sub's committed identity attribute value (within Sub's identity token) satisfies the matching condition in the Pub's access control policy, while the Pub learns nothing about the identity attribute value. Note that not only the Pub does not learn anything about the actual value of Subs' identity attributes but it also does not learn which policy conditions are verified by which Subs, thus the Pub cannot infer the values of Subs' identity attributes. Thus Subs' privacy is preserved in our scheme.

3) Document Dissemination

The Pub broadcasts selectively encrypted documents to Subs. The broadcast is based on access control policies that specify which documents or subdocuments Subs are entitled to access. Such policies specify conditions against Subs' identity attributes. Documents are divided in subdocuments based on the access control policies that apply to them. The policies apply to a subdocument form a *policy configuration*. For each policy configuration, the Pub generates a symmetric key K and encrypts all the subdocuments to which the configuration applies with the same symmetric key. To allow Subs to derive the key K for a given policy configuration using their CSSs in an efficient and secure manner, a new GKM scheme is developed and adopted in this paper. Unlike approaches such as hierarchical GKM [17], [18], our scheme does not require a secure communication channel for updating keys. Section V-C gives a detailed description of the scheme. With this scheme, our broadcasting system efficiently handles new subscriptions and revocations to provide backward and forward secrecy. The system design also ensures that access control policies can be flexibly updated and enforced at the Pub without changing any information stored at Subs.

IV. Background

In this section, we review some basic notions and the cryptographic and mathematical tools which are relevant to the construction of the scheme, to help the reader better understand it.

A. Discrete logarithm problem and computational Diffie-Hellman problem

Definition 1: Let G be a (multiplicatively written) cyclic group of order q and let g be a generator of G. The map $\varphi : \mathbb{Z} \rightarrow G, \varphi(n) = g^n$ is a group homomorphism with kernel $\mathbb{Z}_q$. The problem of computing the inverse map of φ is called the *discrete logarithm problem (DLP) to the base of* g.

Definition 2: For a cyclic group G (written multiplicatively) of order q, with a generator $g \in G$, the *Computational Diffie-Hellman problem (CDH)* is the following problem: Given g^a and g^b for randomly-chosen secret $a, b \in \{0, \ldots, q-1\}$, compute g^{ab}.

Note that CDH-hard is a stronger condition than DL-hard.

B. Pedersen commitment

First introduced in [21], the Pedersen Commitment scheme is an unconditionally hiding and computationally binding commitment scheme which is based on the intractability of the discrete logarithm problem. We describe how it works as follows.

Pedersen Commitment

Setup

A trusted third party T chooses a finite cyclic group G of large prime order p so that the computational Diffie-Hellman problem is hard in G. Write the group operation in G as multiplication. T chooses two generators g and h of G such that it is hard to find the discrete logarithm of h with respect to g, i.e., an integer α such that $h = g^\alpha$. Note that T may or may not know the number α. T publishes (G, p, g, h) as the system's parameters.

Commit

The domain of committed values is the finite field $\mathbb{F}_p$ of p elements, which can be implemented as the set of integers $\mathbb{F}_p = \{0, 1, \ldots, p-1\}$. For a party U to commit a value $x \in \mathbb{F}_p$, U chooses $r \in \mathbb{F}_p$ at random, and computes the commitment $c = g^x h^r \in G$.

Open

U shows the values x and r to open a commitment c. The verifier checks whether $c = g^x h^r$.

C. OCBE Protocols

The Oblivious Commitment-Based Envelope (OCBE) protocols, proposed by Li and Li [20], provide the capability of delivering information to qualified users in an oblivious way. There are three communications parties involved in OCBE protocols: a receiver R, a sender S, and a trusted third party T. The OCBE protocols make sure that the receiver R can decrypt a message sent by S if and only if R's committed value satisfies a condition given by a predicate in S's access control policy, while S learns nothing about the committed value. Note that S does not

even learn whether R is able to correctly decrypt the message or not. The supported predicates by OCBE are comparison predicates $>, \geq, <, \leq, =$ and $\neq$.

The OCBE protocols are built with several cryptographic primitives:

1) The Pedersen commitment scheme.
2) A semantically secure symmetric-key encryption algorithm $\mathcal{E}$, for example, AES, with key length k-bits. Let $\mathcal{E}_{\mathsf{Key}}[M]$ denote the encrypted message M under the encryption algorithm $\mathcal{E}$ with symmetric encryption key Key.
3) A cryptographic hash function $H(\cdot)$. When we write $H(\alpha)$ for an input α in a certain set, we adopt the convention that there is a canonical encoding which encodes α as a bit string, i.e., an element in $\{0,1\}^*$, without explicitly specifying the encoding.

Given the notation as above, we summarize the OCBE protocols for $=$ (EQ-OCBE) and $\geq$ (GE-OCBE) predicates as follows. The OCBE protocols for other predicates can be derived and described in a similar fashion. The protocols' descriptions are tailored to fit the presentation of this paper, and are stated in a slightly different way than in [20].

EQ-OCBE Protocol

Parameter generation

T runs a Pedersen commitment setup protocol to generate system parameters $\mathsf{Param} = \langle G, g, h \rangle$. T outputs the order of G, p, and $\mathcal{P} = \{\mathsf{EQ}_{x_0} : x_0 \in \mathbb{F}_p\}$, where

$$\mathsf{EQ}_{a_0} : \mathbb{F}_p \rightarrow \{\mathsf{true}, \mathsf{false}\}$$

is an equality predicate such that $\mathsf{EQ}_{x_0}(x)$ is true if and only if $x = x_0$.

Commitment

T first chooses an element $x \in \mathbb{F}_p$ for R to commit. T then randomly chooses $r \in \mathbb{F}_p$, and computes the Pedersen commitment $c = g^x h^r$. T sends x, r, c to R, and sends c to S.

Alternatively, in an offline version, T digitally signs c and sends x, r, c together with the signature of c to R. Then the validity of the commitment c can be ensured by verifying T's signature. In this way, after S obtains T's public key for signature verification, no further communication is needed between T and S.

Interaction

- R makes a data request to S.

- Based on this request, S sends an equality predicate $\mathsf{EQ}_{x_0} \in \mathcal{P}$.
- Upon receiving this predicate, R sends S a Pedersen commitment $c = g^x h^r$.
- S picks $y \in \mathbb{F}_p^*$ at random, computes $\sigma = (cg^{-x_0})^y$, and sends R a pair $\langle \eta = h^y, C = \mathcal{E}_{H(\sigma)}[M] \rangle$, where M is a message containing the requested data.

Open

Upon receiving $\langle \eta, C \rangle$ from S, R computes $\sigma' = \eta^r$, and decrypts C using $H(\sigma')$.

The **GE-OCBE** Protocol can be done in a similar way, but in a bit-by-bit fashion, for attribute values of at most ℓ bits long, where ℓ is a system parameter which specifies an upper bound for the bit length of attribute values such that $2^\ell < p/2$. The GE-OCBE protocol is more complex in terms of description and computation compared to EQ-OCBE. It works as follows.

GE-OCBE Protocol

Parameter generation

As in EQ-OCBE, T runs a Pedersen commitment setup protocol to generate system parameters $\mathsf{Param} = \langle G, g, h \rangle$, and outputs the order of G, p. In addition, T chooses another parameter ℓ, which specifies an upper bound for the length of attribute values, such that $2^\ell < p/2$. T outputs $\mathcal{V} = \{0, 1, \ldots, 2^\ell - 1\} \subset \mathbb{F}_p$, and $\mathcal{P} = \{\mathsf{GE}_{x_0} : x_0 \in \mathcal{V}\}$, where

$$GE_{x_0} : \mathcal{V} \to \{\mathsf{true}, \mathsf{false}\}$$

is a predicate such that $\mathsf{GE}_{x_0}(x)$ is true if and only if $x \geq x_0$.

Commitment

As in EQ-OCBE, T chooses an integer $x \in \mathcal{V}$ for R to commit. T then randomly chooses $r \in \mathbb{F}_p$, and computes the Pedersen commitment $c = g^x h^r$. T sends x, r, c to R, and sends c to S.

Similarly, an offline alternative also works here.

Interaction

- R makes a data request to S.
- Based on the request, S sends to R a predicate $\mathsf{GE}_{x_0} \in \mathcal{P}$.
- Upon receiving this predicate, R sends to S a Pedersen commitment $c = g^x h^r$.
- Let $d = (x - x_0) \pmod p$. R picks $r_1, \ldots, r_{\ell-1} \in \mathbb{F}_p$, and sets $r_0 = r - \sum_{i=1}^{\ell-1} 2^i r_i$. If $\mathsf{GE}_{x_0}(x)$ is true, let $d_{\ell-1} \ldots d_1 d_0$ be d's binary representation, with d_0 the lowest bit. Otherwise if GE_{x_0} is false, R randomly chooses $d_{\ell-1}, \ldots, d_1 \in \{0, 1\}$, and sets $d_0 = d - \sum_{i=1}^{\ell-1} 2^i d_i$

$(\mathrm{mod}\ p)$. R computes ℓ commitments $c_i = g^{d_i} h^{r_i}$ for $0 \leq i \leq \ell - 1$, and sends all of them to S.

- S checks that $cg^{-x_0} = \prod_{i=0}^{\ell-1} (c_i)^{2^i}$. S randomly chooses ℓ bit strings $k_0, \ldots, k_{\ell-1}$, and sets $k = H(k_0 \parallel \ldots \parallel k_{\ell-1})$. S picks $y \in \mathbb{F}_p^*$, and computes $\eta = h^y, C = \mathcal{E}_k[M]$, where M is the message containing requested data. For each $0 \leq i \leq \ell - 1$ and $j = 0, 1$, S computes $\sigma_i^j = (c_i g^{-j})^y, C_i^j = H(\sigma_i^j) \oplus k_i$. S sends to R the tuple

$$\langle \eta, C_0^0, C_0^1, \ldots, C_{\ell-1}^0, C_{\ell-1}^1, C \rangle.$$

Open

After R receives the tuple $\langle \eta, C_0^0, C_0^1, \ldots, C_{\ell-1}^0, C_{\ell-1}^1, C \rangle$ from S as above, R computes $\sigma_i' = \eta^{r_i}$, and $k_i' = H(\sigma_i') \oplus C_i^{d_i}$, for $0 \leq i \leq \ell - 1$. R then computes $k' = H(k_0' \parallel \ldots \parallel k_{\ell-1}')$, and decrypts C using key k'.

The OCBE protocol for the $\leq$ predicates (LE-OCBE) can be constructed in a similar way as GE-OCBE. Other OCBE protocols (for $\neq, <, >$ predicates) can be built on EQ-OCBE, GE-OCBE and LE-OCBE.

All these OCBE protocols guarantee that the receiver R can decrypt the message sent by S if and only if the corresponding predicate is evaluated as true at R's committed value, and that S does not learn anything about this committed value.

V. Proposed Scheme

In this section we describe in detail our data dissemination approach. We first introduce the phase of identity tokens issuance to Subs, followed by the phase in which the Pub generates and provides Subs proper subscription secrets. We then describe our group key management scheme. This section also includes an illustrative example.

A. Identity Token Issuance

The IdMgr runs a Pedersen commitment setup algorithm to generate system parameters Param $= \langle G, g, h \rangle$. The IdMgr publishes Param as well as the order p of the finite group G. The IdMgr also publishes its public key for the digital signature algorithm it is using. Such parameters are used by the IdMgr to issue *identity tokens* to Subs. We assume the Subs hold

identity attributes issued by one or more IdPs and present to the IdMgr such identity attributes to receive *identity tokens* as follows. For each identity attribute shown by a Sub, the IdMgr verifies its validity,[5] encodes the identity attribute value as $x \in \mathbb{F}_p$ in a standard way, and issues the Sub an identity token. An identity token is a tuple

$$\mathcal{IT} = (\mathsf{nym}, \mathsf{id\text{-}tag}, c, \sigma),$$

where nym is a pseudonym for uniquely identifying the Sub in the system, id-tag is the tag of the identity attribute under consideration, $c = g^x h^r$ is a Pedersen commitment for the value x, and σ is the IdMgr's digital signature for $\mathsf{nym}, \mathsf{id\text{-}tag}$ and c. The IdMgr passes values x and r to the Sub for the Sub's private use. We require that all identity tokens of the same Sub have the same nym,[6] so that the Sub and its identity tokens can be uniquely matched with a nym. Once the identity tokens are issued, they are used by Subs for proving the satisfiability of the Pub's access control policies; Subs keep their identity attribute values hidden, and never disclose them in clear during the interactions with other parties.

Example 1: Suppose a Sub Bob presents his driver's license to IdMgr to receive an identity token for his age. IdMgr assigns Bob a pseudonym pn-1492. IdMgr deduces from the birthdate on Bob's driver's license that Bob's age is $x = 28$. The IdMgr randomly chooses a value $r = 9270$, and computes a Pedersen commitment $c = g^x h^r$. The IdMgr then digitally signs the message containing Bob's pseudonym, a tag for "age" and the commitment c. The identity token Bob receives from the IdMgr may look like this:
$\mathcal{IT} = (\mathsf{pn\text{-}1492}, \mathsf{age}, 6267292101, 949148425702313975)$.

B. Privacy-Preserving Attribute-Based Conditional Subscription Secret Delivery

We assume that the Pub defines a set of access control policies denoted as $\mathcal{ACPB}$ that specifies which subdocuments Subs are authorized to access. Access control policies are formally defined as follows.

[5]The IdMgr can verify the validity of Sub's identity either in a traditional way, e.g., through a on-the-spot registration, or digitally over computer networks. We will not dive into the details of identity validity check in this paper.

[6]In practice, this can be achieved by requesting the Sub to present a strong identifier that correlates with the identity being registered. Again, we will not discuss this process in this paper.

Definition 3: (**Attribute Condition**).
An attribute condition cond is an expression of the form: "name_A op l", where name_A is the name of an identity attribute A, op is a comparison operator such as $=$, $<$, $>$, $\leq$, $\geq$, $\neq$, and l is a value that can be assumed by attribute A.

Definition 4: (**Access control policy**).
An access control policy acp is a tuple $(s, o, \mathcal{D})$ where: o denotes a set of portions (subdocuments) $\{\mathcal{D}_1, \ldots, \mathcal{D}_t\}$ of document $\mathcal{D}$; and s is a conjunction of attribute conditions $\mathsf{cond}_1 \wedge \ldots \wedge \mathsf{cond}_n$ that must be satisfied by a Sub to have access to o. [7]

Example 2: The access control policy

$$(\text{“}\mathsf{level} \geq 58\text{”} \wedge \text{“}\mathsf{role} = \text{nurse}\text{”}, \{\text{physical exam}, \text{treatment plan}\}, \text{“}\mathsf{EHR.xml}\text{”})$$

states that a Sub of level no lower than 58 and holding a nurse position has access to the subdocuments "physical exam" and "treatment plan" of document EHR.xml.

Different access control policies can apply to the same subdocuments because such subdocuments may have to be accessed by different categories of Subs. We denote the set of access control policies that apply to a subdocument as *policy configuration*.

Definition 5: (**Policy configuration**).
A policy configuration Pc for a subdocument $\mathcal{D}_1$ of a document $\mathcal{D}$ is a set of policies $\{\mathsf{acp}_1, \ldots, \mathsf{acp}_k\}$ where $\mathsf{acp}_i, i = 1, \ldots, k$ is an access control policy $(s, o, \mathcal{D})$ such that $\mathcal{D}_1 \in o$.

There can be multiple subdocuments in $\mathcal{D}$ which have the same policy configuration. For each policy configuration of $\mathcal{D}$, the Pub generates a key K for a symmetric key encryption algorithm (e.g, AES), and uses K to encrypt all subdocuments associated with this policy configuration. Therefore, if a Sub satisfies access control policies $\mathsf{acp}_1, \ldots, \mathsf{acp}_m$, Pub must make sure that the Sub can derive all the symmetric keys to decrypt those subdocuments to which a policy configuration containing at least one access control policies $\mathsf{acp}_i (i = 1, \ldots, m)$ applies.

As in our scheme the actual symmetric keys are not delivered along with the encrypted documents, a Sub has to register its identity tokens at the Pub in order to derive the symmetric encryption key from the disseminated data. During the registration, a Sub receives a set of

[7] In what follow we use the dot notation to denote the different components of an access control policy.

conditional subscription secrets (CSSs), based on the identity attribute names corresponding to the attribute names in the identity tokens. Note that CSSs are generated by the Pub only based on the names of identity attributes and not on their values. So a Sub may receive an encrypted CSS corresponding to a condition which has a value that the Sub' identity attribute does not satisfy. However, in this case, the Sub will not be able to extract the CSS from the message delivering it. Proper CSSs are later used by a Sub to compute symmetric decryption keys for particular subdocuments of broadcast encrypted documents, as discussed in Section V-C. The delivery of CSSs are performed in such a way that the Sub can correctly receive an CSS if and only if the Sub has an identity token whose committed identity attribute value satisfies an attribute condition in Pub's access control policy, while the Pub does not learn any information about the Sub's identity attribute value and does not learn whether Sub has been able to obtain the CSS.

To enable Subs registration, the Pub first chooses an ℓ'-bit prime number q, a cryptographic hash function $H(\cdot)$ whose output bit length is no shorter than ℓ', and a semantically secure symmetric-key encryption algorithm with key length ℓ' bits. The Pub publishes these parameters. Then for an access control policy acp in $\mathcal{ACPB}$ that a subscriber Sub_i under pseudonym nym_i wants to satisfy, it selects and registers an identity token $\mathcal{IT} = (\mathsf{nym}_i, \mathsf{id\text{-}tag}, c, \sigma)$ with respect to each attribute condition cond_j in acp. Note that Sub_i does not register only for the attribute condition which the Sub_i's identity token satisfies; to assure privacy, Sub_i registers its identity token for any attribute condition whose identity attribute name matches the id-tag contained in the identity token. In this way, the Pub cannot infer from Sub_i's registration which condition Sub_i is actually interested in.

The Pub checks if id-tag matches the name of the identity attribute in cond_j, and verifies the IdMgr's signature σ using the IdMgr's public key. If either of the above steps fails, the Pub aborts the interaction. Otherwise, the Pub generates a κ-bit random value $r_{i,j} \in \mathbb{F}_q$, where κ is a security parameter chosen by the Pub. $r_{i,j}$ is the conditional subscription secret. The Pub then starts an OCBE session as a sender (S) to obliviously transfer $r_{i,j}$ to Sub_i who acts as a receiver (R). The Pub maintains a table $\mathcal{T}$ storing all the delivered $r_{i,j}$ along with the associated Sub's pseudonym nym_i and policy condition cond_j. Upon the completion of the OCBE session the Pub performs the following actions:

- If nym_i does not exist in the table, it first creates a row for it.

- It saves $r_{i,j}$ as a record in $\mathcal{T}$ with respect to nym_i and cond_j. An old CSS is overridden by the new CSS $r_{i,j}$ if it already exists. This will allow a Sub to update the Pub with its updated identity tokens.

We remark that all CSSs are independent, so the above CSS delivery process can be executed in parallel. Table $\mathcal{T}$ is used by the Pub to create public information for access control of broadcast documents, and should be protected.

Example 3: Table I shows an example of table $\mathcal{T}$. A Sub under pseudonym pn-0012 who has an identity token with respect to identity tag role registers for all attribute conditions ("role $=$ doc" and "role $=$ nur" are shown in Table I) involving identity attribute role. This Sub does not register for attribute conditions "level ≥ 59", "YoS ≥ 5" [8] and "YoS < 5", either because it does not hold an identity token with identity tag level or YoS, thus cannot register, or because it chooses not to register as it only needs to access subdocuments whose associated access control policy does not require conditions for these attributes. A drawback of registering only for the conditions required is that it may allow an attacker to infer certain attributes about the Sub with high confidence. To protect against such attacks the Sub may choose to register for all conditions. Note that the Sub under pn-0829 registers for both conditions YoS ≥ 5 and YoS < 5, which are mutually exclusive and thus cannot both be satisfied by any Sub. The registration for both conditions is crucial for privacy in that it prevents the Pub from inferring from the Sub's registration behavior which condition the Sub is actually interested in. A Sub under pn-1492 registers for all five attribute conditions.

TABLE I

A TABLE OF CSSS MAINTAINED BY THE PUB

nym	level ≥ 59	YoS ≥ 5	YoS < 5	role $=$ doc	role $=$ nur	...
pn-0012	—	—	—	86571	96875	...
pn-0829	47785	56456	87534	—	—	...
pn-1492	11109	4578	10491	13011	60987	...
...	...	...	...	...	...	...

[8] YoS means "years of service".

C. Group Key Management Scheme

A trivial approach to key management is to deliver all needed keys to qualified Subs. However, this approach suffers from various shortcomings. First, it is a Sub-to-Sub process, as the Pub must delivery the keys to each Sub individually. Second, key maintenance is expensive: a Sub may have to keep track of a high number of keys; whenever an encryption key is changed, every involved Sub needs to be notified and provided with the new keys.

In this section, we propose a new group key management scheme which enables any registered Sub whose identity attributes satisfy at least one of the access control policies applicable to a subdocument to compute the encryption/decryption key, thus to view the content of the subdocument.

1) Basic construction: The Pub generates policy configurations for all subdocuments of $\mathcal{D}$. For each policy configuration, the Pub identifies all the subdocuments to which the policy configuration applies, each of which will then be encrypted with the same symmetric encryption key. Without loss of generality, we will focus on one subdocument, referred to as $\mathcal{D}_1$, when introducing the scheme.

Let $\mathcal{D}_1$'s associated policy configuration be $\mathsf{Pc} = \{\mathsf{acp}_1, \ldots, \mathsf{acp}_\alpha\}$, where each acp_k is a conjunction of conditions $\mathsf{cond}_1^{(k)} \wedge \ldots \wedge \mathsf{cond}_{m_k}^{(k)}$.

For each acp_k, the Pub searches the database table $\mathcal{T}$ to get a list of pseudonyms $U_k = \{\mathsf{nym}_1^{(k)}, \ldots, \mathsf{nym}_{n_k}^{(k)}\}$ whose CSS records corresponding to the attribute conditions in acp_k are in $\mathcal{T}$. The Pub chooses a suitable value

$$N \geq \sum_{k=1}^{\alpha} \#U_k, \tag{1}$$

where $\#U_k$ denotes the cardinality of the set U_k.

Let $r_{i,j}^{(k)} \in \mathbb{F}_q$ be the CSS of a subscriber with $\mathsf{nym}_i^{(k)}$ for $\mathsf{cond}_j^{(k)}$.

For Pc (or equivalently, $\mathcal{D}_1$), the Pub chooses an encryption key K randomly from $\mathbb{F}_q^\times$, and N random τ-bit values $z_1, \ldots, z_N$, where τ is chosen such that $\tau \cdot N$ is larger than 160. This choice of the parameter will ensure the z_i sequence from different sessions will be different with

high probability, and with the effect of "birthday paradox" being considered.[9] Pub lets

$$A = \left(\begin{array}{ccccc} 1 & a_{1,1}^{(1)} & a_{1,2}^{(1)} & \dots & a_{1,N}^{(1)} \\ 1 & a_{2,1}^{(1)} & a_{2,2}^{(1)} & \dots & a_{2,N}^{(1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & a_{n_1,1}^{(1)} & a_{n_1,2}^{(1)} & \dots & a_{n_1,N}^{(1)} \\ \hline \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \hline 1 & a_{1,1}^{(\alpha)} & a_{1,2}^{(\alpha)} & \dots & a_{1,N}^{(\alpha)} \\ 1 & a_{2,1}^{(n')} & a_{2,2}^{(n')} & \dots & a_{2,N}^{(n')} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & a_{n_\alpha,1}^{(\alpha)} & a_{n_\alpha,2}^{(\alpha)} & \dots & a_{n_\alpha,N}^{(\alpha)} \end{array}\right),$$

where

$$a_{i,j}^{(k)} = H(r_{i,1}^{(k)} || r_{i,2}^{(k)} || \dots || r_{i,m_k}^{(k)} || z_j), \tag{2}$$

where $||$ is the string concatenation operation. The Pub solves for a nonzero $(N+1)$-dimensional column vector Y such that $AY = 0$. Note that such a nontrivial Y always exists, because the number of rows of matrix A is less than or equal to N by (1), thus the null space of A is guaranteed nontrivial. We call such a vector Y an *access control vector (ACV)*.

Document Broadcasting:

The Pub sets the vector

$$X = (K, 0, 0, \dots, 0)^T + Y,$$

where v^T is the transpose of vector v and K is the encryption key for $\mathcal{D}_1$. The Pub broadcasts the subdocument $\mathcal{D}_1$ encrypted with K together with the values X, $z_1, \dots, z_N$, as part of the entire document $\mathcal{D}$.

Decryption Key Derivation:

If a Sub with nym_i wants to view the subdocument $\mathcal{D}_1$, it picks an access control policy acp_k it satisfies, and computes

$$K' = (1, a_{i,1}^{(k)}, a_{i,2}^{(k)}, \dots, a_{i,N}^{(k)}) \cdot X,$$

[9] A more detailed analysis on the choice of parameters will be made in a later version of the paper.

where $a_{i,j}^k$ are computed as in (2). We call any $(N+1)$-dimensional vector ν whose first entry is 1 such that $\nu Y = 0$ a *key extraction vector (KEV)* with respect to K and X.

New Subscription:

When a new subscriber Sub′ registers at the Pub, the Pub delivers corresponding CSSs to Sub′, and updates the table $\mathcal{T}$. The Pub then performs a rekey process for all involved subdocuments (or equivalently, policy configurations). When Pub broadcasts new documents, it also publishes the updated X and z_i.

Credential Revocation

The conditions under which a Sub needs to be revoked is out of the scope of this paper. We assume that the Pub will be notified when a Sub with a pseudonym nym_i is revoked from those who may satisfy cond_j. In this case, the Pub simply removes the value $r_{i,j}$ from table $\mathcal{T}$, and performs a rekey process for all involved subdocuments. Allowing particular CSSs to be deleted from $\mathcal{T}$ enables a fine-tuned user management.

Credential Update

A Sub's credentials may have to be updated over time for various reasons such as promotions, change of responsibilities, etc. In this case, the Sub with a pseudonym nym_i submits updated credential $cond_j$ to Pub. Pub simply replace the existing $r_{i,j}$ in the table $\mathcal{T}$, and performs a rekey process only for the subdocuments involved.

Subscription Revocation

When a Sub with a pseudonym nym_i needs to be removed, the Pub removes the row corresponding to nym_i from the table $\mathcal{T}$, and performs a rekey process only for the subdocuments involved.

Note that in all cases of new subscription, credential revocation, credential update and subscription revocation, the rekey process does not introduce any cost to Subs in that except for those whose identity attributes are added, updated or revoked, no Sub needs to directly communicate with the Pub to update CSSs–new encryption/decryption keys can be derived by using the original CSSs and updated public values published by the Pub. The ability to derive the secret encryption/decryption keys using public values is a key point to achieve transparency in subscription handling. Most of the existing GKM schme fails to achieve this objective.

2) An example: We now illustrate how our group key management scheme works through a simplified example in a healthcare scenario. This discussion is based on the information available

at [22].

Example 4: A hospital's data center Pub has to broadcast an XML file "EHR.xml" which contains the electronic health record (EHR) of a patient to the hospital's employees.

– EHR.xml –

```
<PatientRecord>
  <ContactInfo>
    ... ...
  </ContactInfo>
  <BillingInfo>
    ... ...
  </BillingInfo>
  <ClinicalRecord>
    <HistoryOfPresentIllness>
      ... ...
    </HistoryOfPresentIllness>
    <PastMedicalHistory>
      ... ...
    </PastMedicalHistory>
    <Medication>
      // This has the current prescription
      ... ...
    <Medication>
    <AlergiesAndAdverseReactions>
      ... ...
    </AlergiesAndAdverseReactions>
    <FamilyHistory>
      ... ...
    </FamilyHistory>
    <SocialHistory>
      // Things like smoking, drinking, etc.
      ... ...
```

```
    <SocialHistory>
    <PhysicalExams>
      // Weight, body temperature, skin tests, etc.
      ... ...
    </PhysicalExams>
    <LabRecords>
      // X-rays, etc.
      ... ...
    </LabRecords>
    <Plan>
      // What needs to be done, etc.
      ... ...
    </Plan>
  </ClinicalRecord>
</PatientRecord>
```

The subdocuments of "EHR.xml", marked with different XML tags, need to be accessed by different employees based on their roles and other identity attributes. Suppose the roles for the hospital's employees are: receptionist (rec), cashier (cas), doctor (doc), nurse (nur), data analyst (dat), and pharmacist (pha). The involved access control policies for "EHR.xml" are

1) $\mathsf{acp}_1 = (\text{“}\mathsf{role} = \text{rec}\text{”}, \{\langle \text{ContactInfo} \rangle\}, \text{“}\mathsf{EHR.xml}\text{”})$
2) $\mathsf{acp}_2 = (\text{“}\mathsf{role} = \text{cas}\text{”}, \{\langle \text{BillingInfo} \rangle\}, \text{“}\mathsf{EHR.xml}\text{”})$
3) $\mathsf{acp}_3 = (\text{“}\mathsf{role} = \text{doc}\text{”}, \{\langle \text{ClinicalRecord} \rangle\}, \text{“}\mathsf{EHR.xml}\text{”})$
4) $\mathsf{acp}_4 = (\text{“}\mathsf{role} = \text{nur} \wedge \mathsf{level} \geq 59\text{”}, \{\langle \text{ContactInfo} \rangle, \langle \text{Medication} \rangle, \langle \text{PhysicalExams} \rangle, \langle \text{LabRecords} \rangle, \langle \text{Plan} \rangle\}, \text{“}\mathsf{EHR.xml}\text{”})$
5) $\mathsf{acp}_5 = (\text{“}\mathsf{role} = \text{dat}\text{”}, \{\langle \text{ContactInfo} \rangle, \langle \text{LabRecords} \rangle\}, \text{“}\mathsf{EHR.xml}\text{”})$
6) $\mathsf{acp}_6 = (\text{“}\mathsf{role} = \text{pha}\text{”}, \{\langle \text{BillingInfo} \rangle, \langle \text{Medication} \rangle\}, \text{“}\mathsf{EHR.xml}\text{”})$

"EHR.xml" is divided into subdocuments based on these access control policies:

- $\langle \text{ContactInfo} \rangle$: acp_1, acp_4, acp_5
- $\langle \text{BillingInfo} \rangle$: acp_2, acp_6
- $\langle \text{Medication} \rangle$: acp_3, acp_4, acp_6

- ⟨PhysicalExams⟩: acp_3, acp_4
- ⟨LabReports⟩: acp_3, acp_4, acp_5
- ⟨Plan⟩: acp_3, acp_4
- Other stuff: none

The policy configurations and their associated subdocuments are:

$\mathsf{Pc}_1 = \{\mathsf{acp}_1, \mathsf{acp}_4, \mathsf{acp}_5\} \leftrightarrow$ ⟨ContactInfo⟩

$\mathsf{Pc}_2 = \{\mathsf{acp}_2, \mathsf{acp}_6\} \leftrightarrow$ ⟨BillingInfo⟩

$\mathsf{Pc}_3 = \{\mathsf{acp}_3, \mathsf{acp}_4, \mathsf{acp}_6\} \leftrightarrow$ ⟨Medication⟩

$\mathsf{Pc}_4 = \{\mathsf{acp}_3, \mathsf{acp}_4\} \leftrightarrow$ ⟨PhysicalExams⟩, ⟨Plan⟩

$\mathsf{Pc}_5 = \{\mathsf{acp}_3, \mathsf{acp}_4, \mathsf{acp}_5\} \leftrightarrow$ ⟨LabReports⟩

$\mathsf{Pc}_6 = \{\} \leftrightarrow$ Other XML tags

Assume that involved hospital employees have already obtained their identity tokens and have received their CSSs through the delivery phase described in Section V-B, and that the CSS table $\mathcal{T}$ has been created by Pub. Pub chooses an encryption key K_i for each policy configuration Pc_i to encrypt the associated subdocuments.

Without loss of generality, we focus on the case of $\mathsf{Pc}_4 = \{\mathsf{acp}_3, \mathsf{acp}_4\}$ and use the visible records in Table I for demonstration. An SQL-styled database query

SELECT * FROM $\mathcal{T}$ WHERE ‘$\mathsf{role} = \text{doc}'$ <> NULL

returns two rows containing pseudonyms pn-0012 and pn-1492, corresponding to the employees which can potentially access subdocuments to which acp_3 applies. Similarly, it can be easily seen that an employee under pn-1492 is the only one who may satisfy acp_4. The Pub then chooses $N = 3$, and random values z_1, z_2, z_3. For the employee under pn-0012 whose CSS for the attribute condition ““$\mathsf{role} = \text{doc}$” is 86571, the Pub computes values

$$a_{1,1} = H(86571||z_1), a_{1,2} = H(86571||z_2), a_{1,3} = H(86571||z_3).$$

The Pub executes a similar computation for the user under pn-1492 thus obtaining the values

$$a_{2,1} = H(13011||z_1), a_{2,2} = H(13011||z_2), a_{2,3} = H(13011||z_3).$$

By now the Pub has computed both required rows of matrix A for acp_3, and will process acp_4. In this case, for pn-1492 whose CSSs corresponding to the two conditions “$\mathsf{role} = \text{nur}$” and “$\mathsf{level} \geq 59$” are $r_{3,1}$ and $r_{3,2}$, respectively, the Pub computes

$a_{3,1} = H(11109||60987||z_1), a_{3,2} = H(11109||60987||z_2),$
$a_{3,3} = H(11109||60987||z_3).$

For simplicity and illustration purpose, assume $q = 17$, and the resulting matrix over $\mathbb{F}_{17}$

$$A = \begin{pmatrix} 1 & 15 & 3 & 4 \\ 1 & 4 & 13 & 3 \\ 1 & 12 & 5 & 6 \end{pmatrix}.$$

The Pub solves $AY = 0$ to for a non-trivial $Y = (4, 4, 3, 3)^T$. Let $K_4 = 11$. The Pub sets

$$X = Y + (K_4, 0, 0, 0)^T = (15, 4, 3, 3)^T.$$

The Pub publishes X, z_1, z_2, z_3 with the associated subdocuments $\langle$PhysicalExams$\rangle, \langle$Plan$\rangle$, which are encrypted with a symmetric encryption key $K_4 = 11$.

Suppose that the employee under pn-0012 is a doctor, thus satisfies acp_3 and has correctly received the CSS during the delivery process. To obtain the decryption key K_4, the doctor computes $a_{1,1} = 15$, $a_{1,2} = 3$ and $a_{1,3} = 4$ as the Pub did, then calculates

$$K_4 = (1, a_{1,1}, a_{1,2}, a_{1,3}) \cdot X = (1, 15, 3, 4) \cdot (15, 4, 3, 3)^T = 11.$$

The doctor can now use this key to decrypt the subdocuments $\langle$PhysicalExams$\rangle, \langle$Plan$\rangle$.

Suppose that the employee under pn-1492 is a nurse of level 58. Then it satisfies neither acp_3 nor acp_4; therefore it cannot receive the CSSs 11109 or 13001. Although this nurse has the correct CSS 60987 for attribute condition "role $=$ nur", it is not able to compute any of $a_{2,i}$ or $a_{3,i}$, $i = 1, 2, 3$, and thus is not able to obtain a KEV to derive the decryption key K_4. Hence it cannot access the subdocuments $\langle$PhysicalExams$\rangle, \langle$Plan$\rangle$.

The process is similar for the other policy configurations. It is worth remarking, though, that for the policy configuration Pc_6, which is an empty set, the Pub can just encrypt the associated subdocuments with an encryption key K_6 without the need of publishing X or z_i, because in this case no employee is authorized to access this portion of data.

VI. Analysis

In this section we first analyze the security of our techniques. We then discuss relevant performance issues of our techniques.

A. CSS Delivery Security

Two security requirements need be satisfied in the delivery phase of the CSS values $r_{i,j}$ for Sub_i and cond_j:

1) Access control. The CSS value $r_{i,j}$ can be correctly delivered to the user Sub_i if and only if Sub_i has an identity token whose committed identity attribute value satisfies cond_j.
2) User privacy. The Pub learns nothing about the value of the Sub's identity attribute.

The use of OCBE protocols guarantees that both requirements are satisfied. In order to prevent the Pub from inferring any additional information about a Sub's identity attribute value, for such an attribute, the Sub may and shall choose to register its identity token for all conditions involving this attribute. For example, a Sub who holds an identity token whose tag is role and committed value is "nurse" registers the identity token for all attribute conditions associated with role, so that the Pub will not know which condition the Sub is actually interested in, thus successfully guess its real role. Note that the Sub in order to request any CSS corresponding to an attribute condition involving a given attribute, must have an identity token with a tag equal to the name of this attribute. An extension of our approach allows the Sub to further hide the attributes it is interested in, even though the Sub may not have proofs of these identities from the IdP, by obtaining from the IdMgr identity tokens for such attributes whose committed values, set by the IdMgr, lie out of the "normal" range of values.

B. Group Key Management Scheme Analysis

In this section, we focus on the security of our newly proposed group key management scheme. In our analysis, we will model a cryptographic hash function as a random oracle,[10] and base the discussion on requirements listed in Section I.

The security analysis is based on the following lemma, whose proof is straightforward.

Lemma 1: Let $F = \mathbb{F}_q$ be a finite field with q elements. Let V be an n-dimensional F-vector space. Let $v_1, \ldots, v_m$ be m independently uniformly randomly chosen vectors in V, where

[10]Intuitively, a random oracle is a mathematical function that maps every query to a uniformly randomly chosen response from its output domain.

$m \leq n$. Then the probability that $v_1, \ldots, v_m$ are linearly independent is

$$\prod_{i=1}^{m} \left(1 - 1/q^{n-i+1}\right). \tag{3}$$

1) Soundness of the scheme: We say the group key management scheme is *sound* if a qualified Sub can always correctly derive the decryption key.

Let K be an encryption key for a subdocument, and X be the vector published with the encrypted document. The ACV is $Y = X - (K, 0, \ldots, 0)^T$. Recall that for any KEV ν with respect to K and X, we always have $\nu Y = 0$. By definition ν has 1 as its first entry, so it is clear that $\nu X = K$.

The soundness of the proposed key management scheme follows from the fact that each valid Sub can compute a row of the matrix A which is a KEV with respect to K and X, then use this KEV to extract the encryption key.

2) Security: **Minimal trust.** The Pub is the only entity in the key management scheme which is responsible for generating and distributing the encryption/decryption keys.

Key indistinguishability and key independence. Given the public vector X, any element $K \in \mathbb{F}_q$ has the same probability of being the designated encryption key for a policy configuration. Indeed, for this K, let $\nu = (1, a_1, \ldots, a_N)$ be an $(N+1)$-dimensional row vector such that $\nu Y = 0$, where $Y = X - (K, 0, \ldots, 0)^T$, then we have $\nu X = K$.[11] With the hash function $H(\cdot)$ modeled as a random oracle, it follows that it is not possible to distinguish the real encryption key from any value in the key space $\mathbb{F}_q$ by having only knowledge of the public values $X, z_1, \ldots, z_N$. The independence of the encryption keys corresponding to different policy configurations and sessions is a direct consequence.

Forward secrecy. When a Sub is no longer allowed to access the subdocument corresponding to a policy configuration, a rekey takes place.[12] A new encryption key K' is chosen and a new set of values $X, z_1, \ldots, z_N$ is published by the Pub. With the hash function $H(\cdot)$ being modeled as a random oracle, the updated vectors that correspond to the Subs' key extraction vectors from the previous session can be viewed as chosen independently uniformly at random. Since

[11]Such a ν with 1 as its first entry can almost always be found. The only exception happens when X has its first entry followed all 0s. An X of this form can easily be identified by the Pub and excluded from consideration.

[12]Forward secrecy is relevant in our context when documents are updated and the policies associated with the updated documents change. We discuss it for completeness.

the total number of Subs is no more than N, by Lemma 1, we conclude that all these updated vectors are linearly independent with a probability greater than or equal to

$$\prod_{i=1}^{N}\left(1-1/q^{N-i+1}\right) \geq \prod_{i=1}^{\infty}\left(1-1/q^{i}\right) \approx 1,$$ [13]

when q is large. Therefore, by construction all key extraction vectors ν such that $\nu X = K'$ spans an N-dimensional $\mathbb{F}_q$-subspace W. The updated vector $\widetilde{\nu}$ for Sub is an $(N+1)$-dimensional row vector with 1 as its first entry. It can be easily shown that the probability that $\widetilde{\nu}$ is in W is $1/q$. When q is large, the probability is negligible. Therefore in practice any revoked Sub cannot correctly compute the updated encryption keys by following the key derivation procedure.

Backward secrecy. Similar to the discussion of forward secrecy, it can be easily seen that a newly joined Sub can retrieve an earlier encryption key only with a negligible probability.

Collusion resistance. With $H(\cdot)$ modeled as a random oracle, external or revoked adversaries have only knowledge of independent random vectors. Colluding adversaries do not have advantages compared to an individual attacker who tries to use these independent information pieces. When q is large, the probability that the decryption key can be retrieved by colluding adversaries who follow the key extraction procedure is negligible.

3) Other requirements: **Bandwidth overhead.** Once a Sub's CSSs are delivered via the delivery phase, they are stable for the Sub and no further direct communication is required between Pub and Sub. Each time the dynamics of the set of subscribers or documents changes (e.g., encryption key update, a Sub joining or leaving the set of subscribers), the values $X, z_1, \ldots, z_N$ are broadcast with the encrypted documents. Such a broadcast has $O(\ell' N)$-bit bandwidth overhead, where ℓ' is the bit length of the size of the underlying finite field $\mathbb{F}_q$, for transmitting these values. As we will see in Section VII, this is not a problem in practice.

Computational costs. A Sub only needs to conduct $N+1$ hashing operations, compute an inner product of two $(N+1)$-dimensional $\mathbb{F}_q$-vectors to extract the encryption key, and perform a symmetric-key decryption for a document. As shown by the experiments in Section VII, this computation is light-weight.

[13]The formula on the left hand side is forumula (3) with $n = N$ and $m = n$. This is because all vectors under consideration have 1 as the value of their first entries. If we ignore all their first entries, we are left with N-dimensional q-vectors. A necessary condition for all these N-dimensional vectors to be linearly independent is that all original $(N+1)$-dimensional vectors are linearly independent.

However, each time when a new encryption key and an access control vector need be generated, the Pub has to solve a linear system of size N, over a large finite field which can be computationally costly as N becomes large. Experiments in Section VII evaluate the performance of the scheme in terms of the size of the matrix A.

Storage requirements. Nowadays we are less worried about the storage requirements on both the Pub and the Subs' sides in general. Users as mobile clients may have special space limitation to consider. However, a Sub only needs $O(\ell' N)$ bits to store the needed information (e.g., the CSSs, the KEV, information about the finite fields) when deriving a decryption key. The space requirement can be easily satisfied for a reasonable number of Subs and a finite field of suitable size.

VII. Experimental Results

In this section, we present experimental results for various parameters in our system. We have built a fully functioning system in C/C++ that incorporates our techniques for privacy preserving CSS delivery based on the OCBE protocols, and efficient key management.

The experiments were performed on a machine running GNU/Linux kernel version 2.6.27 with an Intel® Core™ 2 Duo CPU T9300 2.50GHz and 4 Gbytes memory. Only one processor was used for computation. The code is built with 64-bit gcc version 4.3.2, optimization flag -O2. The code is built over the G2HEC C++ library [23], which implements the arithmetic operations in the Jacobian groups of genus 2 curves. For the CSS delivery and group key management phases, we use V. Shoup's NTL library [24] version 5.4.2 for finite field arithmetic, and SHA-1 implementation of OpenSSL [25] version 0.9.8 for cryptographic hashing.

A. CSS Delivery

The CSS delivery phase uses the OCBE protocols, which consist of three major steps: 1) extra commitments generation (OCBE for inequality conditions only) at the Sub, 2) envelope composition at the Pub, and 3) envelope opening at the Sub.[14] In this section, we evaluate the performance of these three steps for both EQ- and GE-OCBE protocols.

[14]Interested readers may refer to [20], [7] for details.

We choose the group G to be the rational points of the Jacobian variety (aka. Jacobian group) of a genus 2 curve

$$C : y^2 = x^5 + 2682810822839355644900736x^3$$
$$+22659135529599310290211 6x^2 + 254767471595292971789991 8x$$
$$+4797309959708489673059350$$

over the prime field $\mathbb{F}_q$, with $q = 5 \cdot 10^{24} + 8503491$ (83 bits). The Jacobian group of this curve has a prime order
$p =$2499999999999413043860099940220946396619751607569 9 (164 bits).[15]

The OCBE parameter generation program chooses non-unit points g and h in the Jacobian group as the base points for constructing the Pedersen commitments.

We use attribute values that satisfy the attribute conditions in the policy. We expect a similar running time if the attribute values do not satisfy the attribute conditions in the policy. For GE-OCBE, we vary the value of the ℓ parameter, which controls the range of the difference between the committed value x and the value x_0 specified in the policy, from 5 to 40, and performed evaluation accordingly. In this experiment, we run both EQ- and GE-OCBE protocols for randomly chosen data, for 50 rounds, and take the average values. Figure 2 and Table II report the average running time of one round of the GE-OCBE protocol and the EQ-OCBE protocol, respectively.

The experimental results show that the overall computation takes at most a few seconds for the privacy preserving subscription through the OCBE protocols when all possible identity attribute values lie within an interval of width up to 2^{40}. Because of the impact of the values of ℓ on the performance of the CSS delivery, it is important to choose ℓ as small as possible, while at the same time large enough to upper-bound the attribute values. For example, the identity attribute "age" (in years) usually has values from 0 to 200 and can be represented using 8 bits. In this case, it is sufficient to choose ℓ to be 8. We expect other OCBE protocols for inequality predicates to have a performance similar to that of GE-OCBE, because the design and operations are similar.

[15]The data is taken from [26].

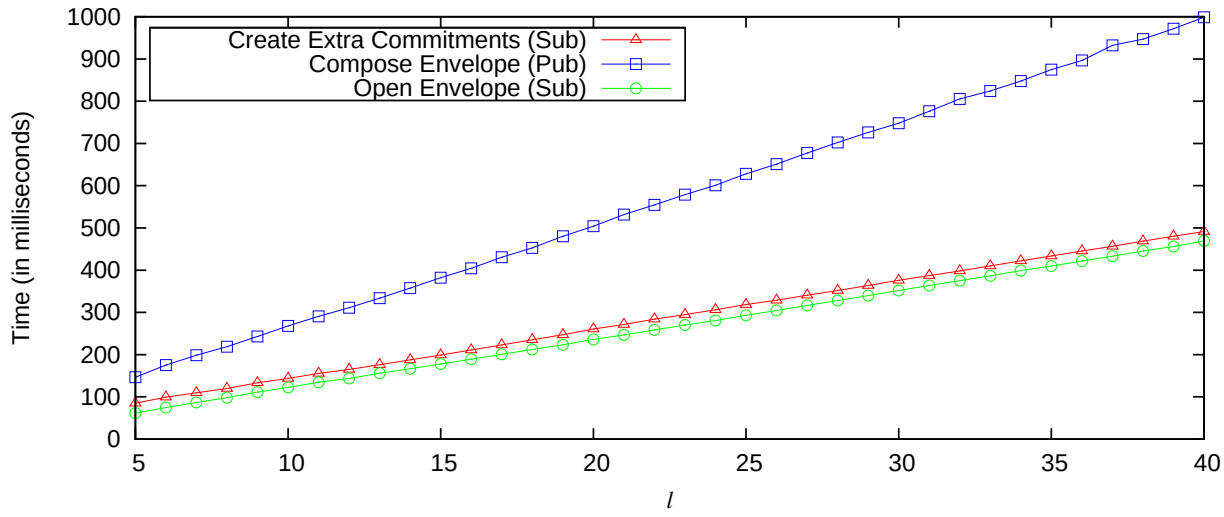

Fig. 2. Average computation time for running one round of GE-OCBE protocol

TABLE II

AVERAGE COMPUTATION TIME FOR RUNNING ONE ROUND OF THE EQ-OCBE PROTOCOL

Computation	Time (in ms)
Create Extra Commitments (Sub)	0.00
Open Envelope (Sub)	35.25
Compose Envelope (Pub)	11.80

B. Group Key Management

In this section we perform experiments to evaluate the performance of generation of the ACVs at the Pub and the key derivation from the ACVs at the Sub, and the size of the ACVs for different system parameters including the number of maximum users and the number of attribute conditions. All finite field arithmetic operations are performed in an 80-bit prime field.

The following experiments are performed with different *user configurations*. A user configuration indicates the number of current Subs and the maximum user limit N. For example, the configuration '25% Subs' with N = 1000, has 250 Subs. We use 25 policies, each on average containing two conditions. Each Sub satisfies the policy in the policy configuration under consideration. We illustrate the experiments for one subdocument, as computations related

to different subdocuments are independent and similar, and thus can be performed in parallel.

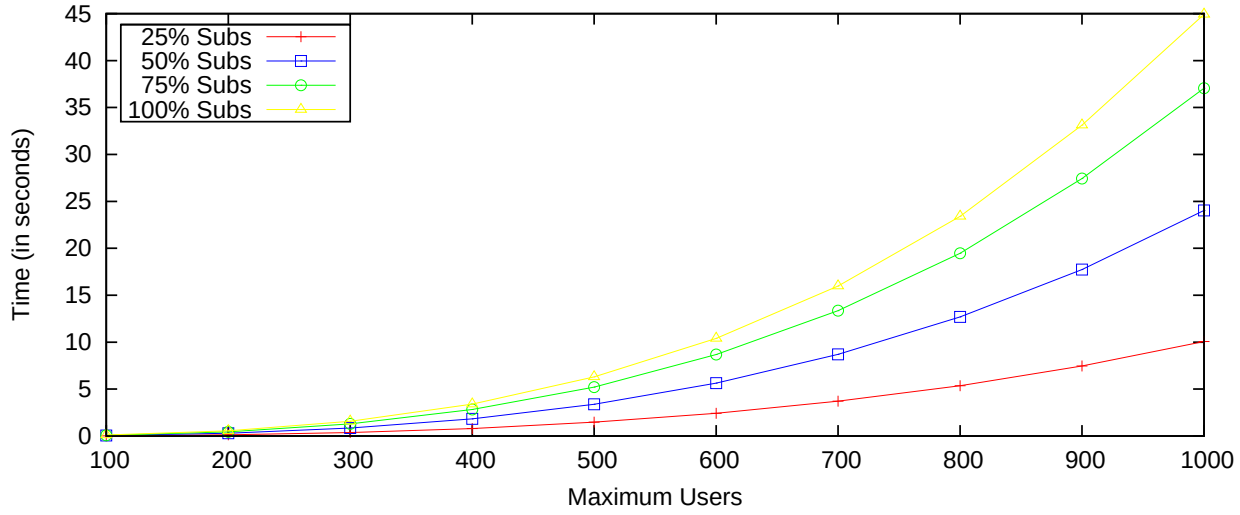

Fig. 3. Time to generate an ACV for different user configurations

Figure 3 reports the average time spent in computing an ACV corresponding to the matrix A for different user configurations. An ACV is a random vector in the null space of matrix A. We generate an ACV by first computing a basis of the null space of A, then choosing the ACV as a random linear combination of the basis vectors. For a given N, the ACV computation time increases with the number of current users. This is consistent with the fact that as the number of current users increases, the number of rows in the matrix A (consequently the rank of A) increases, requiring an increasing amount of elementary matrix operations to compute the null space for the linear solver of NTL. As shown in Figure 3, this computation is efficient (less than 45 seconds on a personal computer) for reasonably large N values.

Figure 4 reports the average time for Subs to derive the symmetric keys from ACVs and KEVs for different user configurations. Key derivation is performed by Subs whose computational capabilities may be limited. Therefore, an efficient decryption key derivation process is desired. As Figure 4 shows it not only incurs minimal computational costs (a few milliseconds), but also increases only linearly with N.

Figure 5 shows the average size of ACVs for different user configurations. Another design goal of our approach is to keep the additional communication overhead minimum. In order to

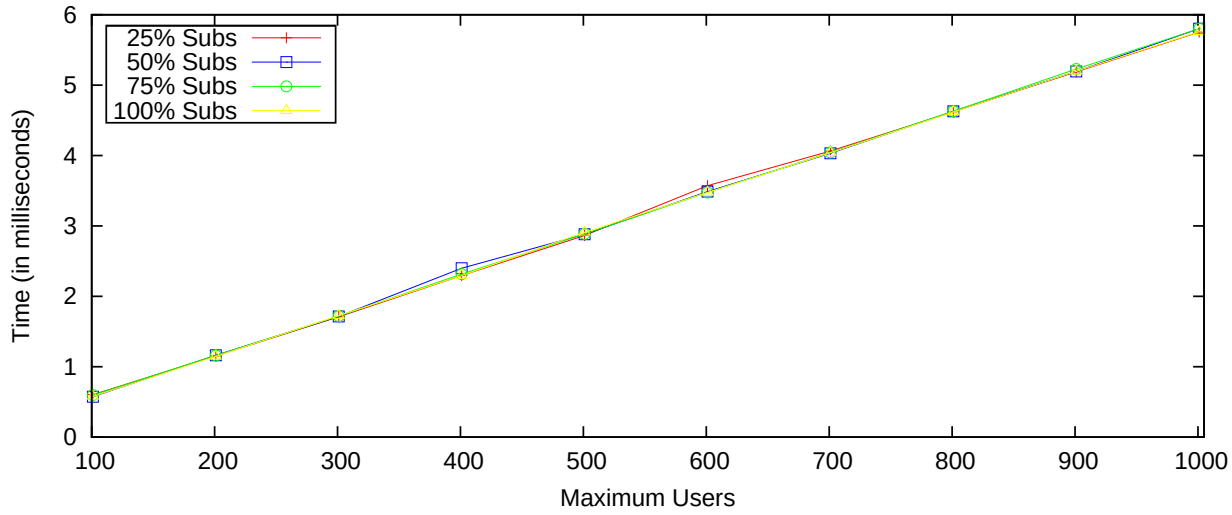

Fig. 4. Key derivation time for different user configurations

achieve this goal, the Pub compresses the ACVs before broadcasting them with the encrypted documents. As Figure 5 indicates, our approach only requires a few kilobytes to transmit these vectors, and the size increases only linearly with N.

In the following experiment, we measure the time for ACV generation (at Pub) and key derivation (at Sub) by varying the average number of attribute conditions per policy, and keeping the number of policies and the maximum number of users fixed at 25 and 500, respectively.

Figure 6 shows the average running time for ACVs generation at Pub and symmetric decryption key derivation at Sub, for different number of conditions per policy. As the number of conditions per policy increases, the key derivation time remains almost constant but the ACV generation time slightly increases (by less than 100 milliseconds).

VIII. Further Discussions

In this section, we further discuss some relevant features of our scheme and also compare it with another possible approach.

A. Hierarchical Key Management

It can be easily seen that our proposed group key management scheme automatically supports a hierarchical access control, which means that if a Sub can retrieve the encryption/decryption key

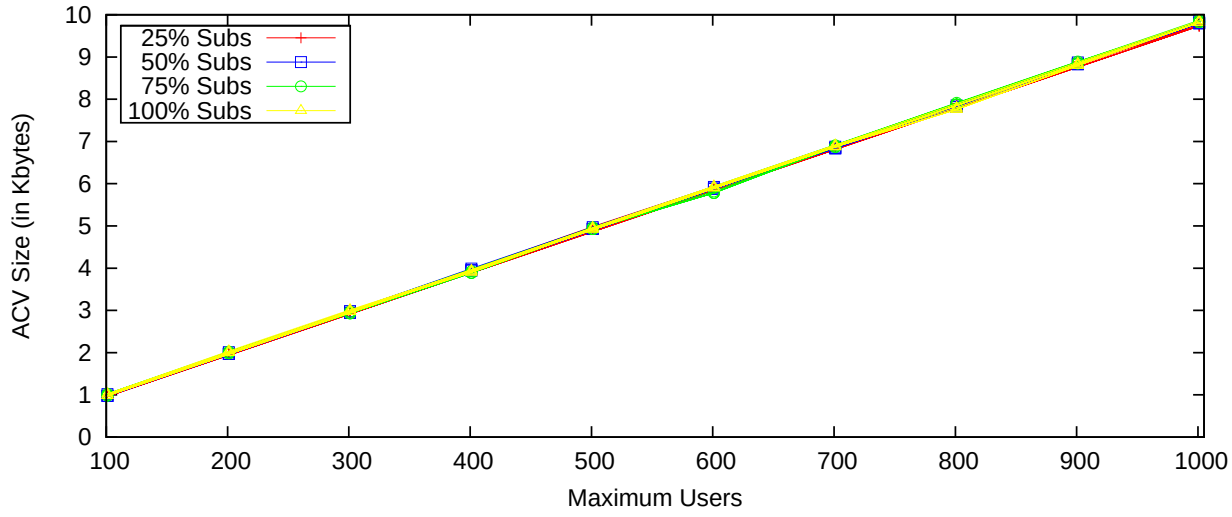

Fig. 5. Size of ACV for different user configurations

corresponding to a policy configuration Pc, then it can retrieve keys for all policy configurations that are dominated by Pc, where the notion of *dominance relation* between policy configurations is defined as follows.

Definition 6: (**Dominance relation**).
Let Pc_i and Pc_j be two policy configurations that apply to a document $\mathcal{D}$. We say that Pc_i dominates Pc_j if and only and if $\mathsf{Pc}_i \subseteq \mathsf{Pc}_j$.
Indeed, when the Sub satisfies an access control policy $\mathsf{acp} \in \mathsf{Pc}_i$ and Pc_i dominates Pc_j, then automatically $\mathsf{acp} \in \mathsf{Pc}_j$. Therefore the Sub can use the same set of CSSs that are used to derive the decryption key for Pc_i to construction that for Pc_j. Our approach can be further optimized by eliminating reduntant calculations at Pub by taking advantage of dominance relationships.

B. Advantages over a Simplistic Approach

A simplistic approach to privacy-preserving policy-based content distribution is to obliviously deliver (via the OCBE protocols) the encryption keys to a Sub for all broadcast contents. However, this approach requires quite a large amount of communications between the Pub and the Subs, and an individual Sub may need to maintain a high number of keys, one per policy configuration the Sub satisfies. Moreover, when any encryption key is changed, e.g., when a new Sub joins or a subscription revocation takes place, the Pub has to communicate directly

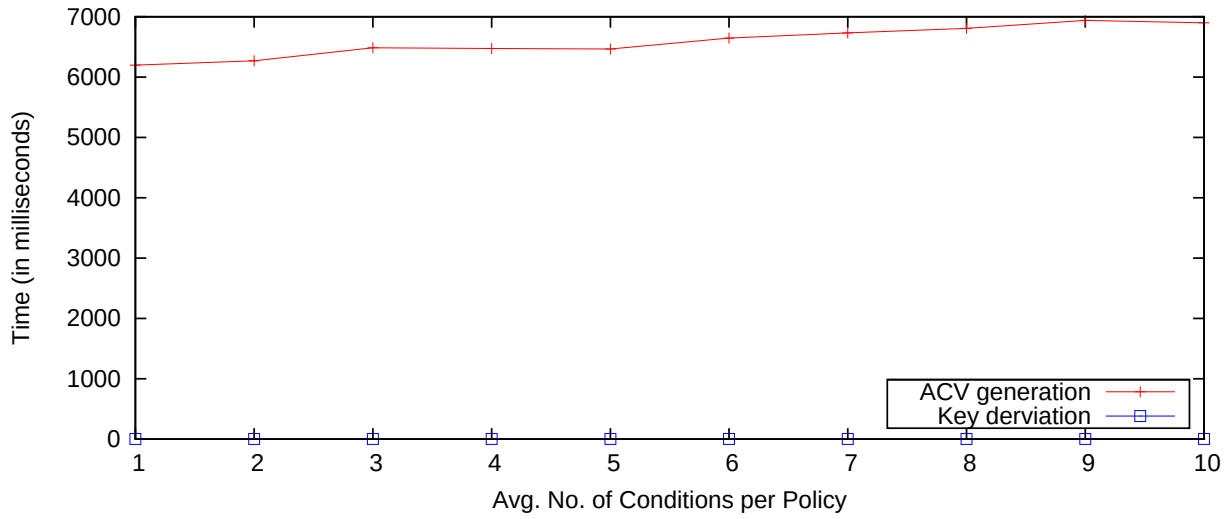

Fig. 6. ACV generation and key derivation for different number of conditions per policy

with all Subs in order to update them with the new keys. This approach thus results in high costs for the Pub, and is inconvenient for both the Pub and the Subs.

In contrast, our approach only requires the Pub to directly communicate with Subs during the identity token registration phase to deliver the CSSs. Subs only need to maintain a list of CSSs. All the CSSs are stable, in that they do not change after registration, unless an update of identity attribute happens and the Sub registers its new identity token. When a rekey process takes place, involved Subs just need to perform local computations to derive the new keys based on updated information published by Pub and their old CSSs, without establishing direct communications with Pub. Furthermore, in our scheme, the number of CSSs a Sub needs to manage is always bounded by the total number $\mathcal{N}$ of attribute conditions involved in the access control policies, whereas the simplistic approach requires a Sub to manage one key for each policy configuration, and the total number of policy configurations can be $2^{2^{\mathcal{N}}}$ in the worst case. Our approach is efficient in terms of communication and computation, and is easy to use and maintain for the Pub and the Subs.

C. Scalability

The experimental results in Section VII have shown that the proposed key management scheme works efficiently even when there are thousands of subscribers for a subdocument. However, as

the upper bound N of the number of involved subscribers gets large, solving the linear system $AY = 0$ over a large finite field $\mathbb{F}_q$ becomes the most computationally expensive operation in our proposed key management scheme. Solving this linear system with the method of Gaussian-Jordan elimination [27] takes $O(N^3)$ time. Although this computation is executed at the Pub, which is usually capable of carrying on computationally expensive operations, when N is very large, e.g., $N = 1,000,000$, the resulting costs may be too high for the Pub. In this case, the Pub can divide all the involved Subs into multiple groups of a suitable size (e.g., 1000 each), compute a different ACV Y for each group, and broadcast it to the corresponding group, while the subdocument is still encrypted with one uniform key. In practice, the grouping criterion can be based on access control policies, subscribers' physical locations, and so forth. The computation of the ACV for each group is independent, thus can be performed in parallel.

Note also that a different and interesting group key management was suggested by one of the anonymous referees when this paper was being peer-reviewed. We discuss and compare it with our scheme below.

D. A New Group Key Management Scheme

The new method of group key management proposed by one of the reviewers is the following one.

Pub first chooses a "well-known marker" m that is long enough to avoid collision. Pub then chooses a symmetric encryption key k for a (sub)document D associated with access control policies $\mathsf{acp}_1, \ldots, \mathsf{acp}_\alpha$. Let H be a random oracle. Pub chooses a (long enough) random value

z, and then publishes encrypted D together with z and N values

$$
\begin{aligned}
&(k||m) \oplus H(r_{1,1}^{(1)}||r_{1,2}^{(1)}|| \ldots ||r_{1,m_1}^{(1)}||z),\\
&(k||m) \oplus H(r_{2,1}^{(1)}||r_{2,2}^{(1)}|| \ldots ||r_{2,m_1}^{(1)}||z),\\
&\qquad \vdots\\
&(k||m) \oplus H(r_{n_1,1}^{(1)}||r_{n_1,2}^{(1)}|| \ldots ||r_{n_1,m_1}^{(1)}||z),\\
&\qquad \vdots\\
&(k||m) \oplus H(r_{1,1}^{(\alpha)}||r_{1,2}^{(\alpha)}|| \ldots ||r_{1,m_\alpha}^{(\alpha)}||z),\\
&(k||m) \oplus H(r_{2,1}^{(\alpha)}||r_{2,2}^{(\alpha)}|| \ldots ||r_{2,m_\alpha}^{(\alpha)}||z),\\
&\qquad \vdots\\
&(k||m) \oplus H(r_{n_\alpha,1}^{(a)}||r_{n_\alpha,2}^{(a)}|| \ldots ||r_{n_\alpha,m_\alpha}^{(\alpha)}||z),
\end{aligned}
$$

where $\oplus$ denotes the bitwise exclusive-or operation.

To decrypt the message, a Sub finds which access control policy is satisfies for the document, and then computes the key k using H and z, as follows. If CSSs r_1,..,r_w satisfy an access control policy, then the Sub computes $H(r_1||...|r_w||z)$. It then tries to decrypt all above N values by XORing with $H(r_1||...|r_w||z))$. If any of the values contains the well-known marker m, then it removes m from the decrypted text to obtain k.

This solution will require $O(N)$ computation at Pub, $O(N)$ ciphertext size, and $O(N)$ computation at Sub.

The new approach reduces the load at the Pub and does seem to satisfy our security requirements assuming the random oracle model. However, for an instantiation (with cryptographic hash functions), one restriction of this approach is that the length of the key must be strictly less than that of the hash output, whereas the scheme proposed in our paper allows the key size (in bits) to be as large as twice of that of the hash output. Although a technique with ”multiple markers” can be used to split a long key into shorter segments, this will also increase the load on Subs. The choice of practical parameters (e.g. key size, hash output size, length of m) for this new approach also seems to be a subtle issue. In the case of our scheme this is relatively easy, and a security analysis is clearly provided.

Another advantage of our scheme over the new approach can be seen in the following

demonstrative case. Suppose two different documents/subdocuments satisfying the same policy configuration are to be encrypted with different keys k_1 and k_2 (e.g., data are preprocessed then assigned policies afterwards, and broadcast on a daily basis). Since the two documents share the same policy configuration, thus also the same user base, it is reasonable that in deployment they share the same z_i values. For our scheme in this case, for both documents, the Pub can simply compute one matrix and its null space (once for both documents), then choose two linearly independent ACVs and associate with them two different keys to compose the public vectors to broadcast. From the Sub's point of view, once a Sub receives all z_i's (say, for the day), the Sub can compute the hash values and cache the resultant vector for future use to retrieve documents associated with the same policy. Suppose an outside attacker knows one of the keys, say k_1. Then this knowledge alone does not help the attacker to learn any information about k_2. However, for the newly proposed scheme in this case, if the same z value is assigned to the two documents associated with the same policy configuration, for k_1, the public values will be something like $X_1 = (k_1||m) \oplus H(r||z)$, and for k_2, the values will be like $X_2 = (k_2||m) \oplus H(r||z)$. Although a valid Sub can still use the cached hash value $H(r||z)$ to retrieve both keys k_1 and k_2, an attacker who knows k_1 can immediately obtain k_2 by computing $(X_1 \oplus X_2 \oplus (k_1||0\text{-padding}))$ and extracting the first bits. In this sense, our scheme is more flexible and more secure to allow fine-grained control of encrypted data.

More importantly, our scheme is provably secure (see [28]), whereas the formal security proof of the newly proposed scheme is not yet available. However, provided that the security of the new scheme is formally defined and analyzed, it might as well be suitable for adoption in a variety of applications, including the one under this paper's consideration.

IX. Conclusions and Future Work

We have proposed an approach to support attribute-based access control while preserving privacy of users' identity attributes in a document broadcasting setting. Our approach is supported by a new group key management scheme which is secure and allows qualified subscribers to efficiently extract decryption keys for the portions of documents they are allowed to access, based on the subscription information they have received from the document publisher. The scheme efficiently handles joining and leaving of subscribers, with guaranteed security. Experimental results show that subscribers efficiently derive decryption keys, and that a rekey process at

the publisher takes less than one minute for up to a thousand subscribers even on a personal computer.

Our further research will focus on scalability and optimization issues. We will develop proper criteria for clustering subscribers depending on different requirements of broadcasting. We have also devised optimization strategies to reduced the size of the matrix A based on a partial order among the set of access control policies.[16] In our current implementation we use the kernel() function of V. Shoup's NTL library as the linear solver, and perform computations on the CPU. We plan to further improve the performance of our scheme by extending the techniques [29], [30], [31] which implement fast linear algebra operations with floating-point arithmetic or over finite fields of various sizes, based on cache-aware CPU approaches and GPU architectures like Nvidia CUDA [32].

ACKNOWLEDGEMENTS

The work reported in this paper has been partially supported by the NSF grant 0712846 "IPS: Security Services for Healthcare Applications," and the MURI award FA9550-08-1-0265 from the Air Force Office of Scientific Research. We would also like to thank the anonymous reviewers for their valuable comments.

REFERENCES

[1] N. Shang, F. Paci, M. Nabeel, and E. Bertino, "A privacy-preserving approach to policy-based content dissemination," Purdue University, Tech. Rep. CERIAS TR 2009-14, 2009.

[2] "Extensible Access Control Markup Language (XACML)," http://xml.coverpages.org/xacml.html.

[3] "Liberty Alliance," http://www.projectliberty.org/.

[4] "OpenID," http://openid.net/.

[5] "Windows CardSpace," http://msdn.microsoft.com/en-us/library/aa480189.aspx.

[6] "Higgins Open Source Identity Framework," http://www.eclipse.org/higgins/.

[7] F. Paci, N. Shang, E. Bertino, K. Steuer Jr., and J. Woo, "Secure transactions' receipts management on mobile devices," in *Symposium on Identity and Trust on the Internet (IDtrust Symposiums)*, NIST, Gaithersburg, MD, USA, April 2009.

[8] E. Bertino and E. Ferrari, "Secure and selective dissemination of XML documents," *ACM Trans. Inf. Syst. Secur.*, vol. 5, no. 3, pp. 290–331, 2002.

[9] G. Miklau and D. Suciu, "Controlling access to published data using cryptography," in *VLDB '2003: Proceedings of the 29th international conference on Very large data bases*. VLDB Endowment, 2003, pp. 898–909.

[16] This approach will be discussed in a later version of this technical report.

[10] Y. Challal and H. Seba, "Group key management protocols: A novel taxonomy," *International Journal of Information Technology*, vol. 2, no. 2, pp. 105–118, 2006.

[11] A. Kundu and E. Bertino, "Structural signatures for tree data structures," *Proc. VLDB Endow.*, vol. 1, no. 1, pp. 138–150, 2008.

[12] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Eurocrypt 2005, LNCS 3494*. Springer-Verlag, 2005, pp. 457–473.

[13] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "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*. New York, NY, USA: ACM, 2006, pp. 89–98.

[14] X. Zou, Y. Dai, and E. Bertino, "A practical and flexible key management mechanism for trusted collaborative computing," *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pp. 538–546, April 2008.

[15] H. Harney and C. Muckenhirn, "Group key management protocol (gkmp) specification," Network Working Group, United States, Tech. Rep., 1997.

[16] H. Chu, L. Qiao, K. Nahrstedt, H. Wang, and R. Jain, "A secure multicast protocol with copyright protection," *SIGCOMM Comput. Commun. Rev.*, vol. 32, no. 2, pp. 42–60, 2002.

[17] C. Wong and S. Lam, "Keystone: a group key management service," in *International Conference on Telecommunications, ICT*, 2000.

[18] A. Sherman and D. McGrew, "Key establishment in large dynamic groups using one-way function trees," *Software Engineering, IEEE Transactions on*, vol. 29, no. 5, pp. 444–458, May 2003.

[19] G. Chiou and W. Chen, "Secure broadcasting using the secure lock," *Software Engineering, IEEE Transactions on*, vol. 15, no. 8, pp. 929–934, Aug 1989.

[20] J. Li and N. Li, "OACerts: Oblivious attribute certificates," *IEEE Transactions on Dependable and Secure Computing*, vol. 3, no. 4, pp. 340–352, 2006.

[21] T. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *CRYPTO '91: Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology*. London, UK: Springer-Verlag, 1992, pp. 129–140.

[22] "XML in clinical research and healthcare industries," http://xml.coverpages.org/healthcare.html.

[23] N. Shang, "G2HEC: A Genus 2 Crypto C++ Library," http://www.math.purdue.edu/~nshang/libg2hec.html.

[24] V. Shoup, "NTL library for doing number theory," http://www.shoup.net/ntl/.

[25] "OpenSSL the open source toolkit for SSL/TLS," http://www.openssl.org/.

[26] P. Gaudry and É. Schost, "Construction of secure random curves of genus 2 over prime fields," in *Advances in Cryptology – EUROCRYPT 2004*, ser. LNCS, vol. 3027. Springer-Verlag, 2004, pp. 239–256.

[27] D. Dummit and R. Foote, "Gaussian-Jordan elimination," in *Abstract Algebra*, 2nd ed. Wiley, 1999, p. 404.

[28] N. Shang, E. Bertino, and X. Zou, "A broadcast approach to group key management with access control vectors," 2009, preprint.

[29] "The PALO ALTO Project," http://ljk.imag.fr/membres/Laurent.Fousse/palo-alto/.

[30] M. Abshoff and C. Pernet, "Efficient exact linear algebra over GPU," SAGE Day 9 talk, available http://membres-liglab.imag.fr/pernet/Publications/SD9_mabshoff_cpernet.pdf, Aug. 2008.

[31] "FFLAS-FFPACK: Finite field linear algebra subroutines/package," http://ljk.imag.fr/membres/Jean-Guillaume.Dumas/FFLAS/.

[32] “Nvidia CUDA,” http://www.nvidia.com/object/cuda_home.html.