THE PUBLISHING HOUSE
OF THE ROMANIAN ACADEMY

PROCEEDINGS OF THE ROMANIAN ACADEMY, Series A,
Volume 18, Special Issue 2017, pp. 413-430

CRYPTOGRAPHIC ACCESS CONTROL FOR MANDATORY SECURITY POLICIES USING ATTRIBUTE-BASED ENCRYPTION

Daniel PLECAN

"Alexandru Ioan Cuza" University of Iasi, Department of Computer Science, Romania
E-mail: daniel.plecan@gmail.com

Abstract. This paper proposes a solution for enforcing mandatory access control policies through cryptography as an alternative to the continuous mediation of access operations by a reference monitor. Attribute-based encryption is used to model the lattice structures underlying this type of policies through techniques of constructing and distributing access trees and public key parameters.

Key words: Cryptographic access control, attribute-based encryption, mandatory policies, access trees, lattice structures.

1. INTRODUCTION

Any *secured* information system must offer, among others, protection against unauthorized disclosure and malicious alterations of computing *resources*, whilst ensuring that *legitimate* agents can easily and safely access them. These three security properties are also known as *privacy*, *integrity* and respectively *availability*.

Access control is the process of enforcing these properties and it requires that each access procedure of a system and its resources is preceded, directly or indirectly, by an access control decision which makes sure that only authorized operations take place.

The direct precedence is strongly related to the *reference monitor* notion defined by Lampson in [1], on which many of the contemporary access control mechanisms are based. A reference monitor is a trusted party that intercepts all the operations meant to be performed on resources and allows only the ones considered legitimate in relation to the initiator of the respective operation and the underlying security policy.

The security evaluation happens indirectly when mediation of every operation is not necessary and access control is a consequence of the mechanisms used to distribute the data. For example, in the setup phase, a trusted component of the system can encrypt all the resources and then share appropriate decryption keys to agents based on their privileges as defined by the desired security policy. We can then rely on the security of the employed cryptographic scheme and the correctness of the key distribution to ensure that agents can only access resources for which they have the required authorizations. More specifically, the agent's access rights will be given by the keys in his possession with which he will be able to read and/or manipulate certain resources. Given that access operations imply cryptographic evaluations such as encryption and decryption, it is no longer required for each one to be intercepted and evaluated by a reference monitor. This approach is known in literature as *cryptographic access control* and it represents an important step towards a more distributed way of dealing with access control.

Mandatory policies rely on rules defined and administered by a trusted party with which the participants have to strictly comply. Data protection focuses on its sensitivity and is achieved through a rigorous control of the flow of information which can be formally bounded. During the setup phase of a mandatory system, the trusted party assigns security clearances to agents and security labels to resources. The two are matched up against each other when access is requested and it is granted only if the initiating agent possesses the appropriate clearance required by the resource in question. This kind of policies lends itself well to environments where data leaks and corruptions are critical such as the military ones.

Besides relying on a trusted party for the definition of access rules, it is most often than not the case that mandatory policies also make use of a central authority for access control enforcement. The authority is active during normal operation of the system and intercepts all the access requests in order to allow only the legitimate ones. This means that the proper functioning of this mediating component is vital to the usability of the system, which makes it a potential single point of failure. Any malfunctions or performance bottlenecks at its level will severely incapacitate the protected system. Coupled with the fact that the data is usually stored in clear and is thus susceptible to theft in case the storage is compromised, it is clear that another approach is desirable.

The main focus of this paper is to propose a solution for the enforcement of mandatory policies that uses cryptography as an additional layer of protection for data, as well as a means of avoiding continuous mediation of access operations. The type of cryptographic schemes that we are going to use is attribute-based encryption (ABE) because it fits very well with our needs as it allows for fine-grained control over encrypted data. In the key-policy approach, ciphertexts are associated with one or more attributes that describe the nature of the encrypted information and the private keys of the users include access structures that have to be satisfied by the set of attributes belonging to a ciphertext in order for decryption to be possible. Since mandatory policies are formally underlined by lattice structures that govern the information flow, we will show how we can use ABE to model and enforce these lattices in terms of *read* and *write* operations.

We use the ABE scheme for monotonic access structures proposed by Goyal et al. in [2] to enforce general mandatory policies based on the information flow model defined by Denning in [4]. We also tackle the Biba [5] and Bell-LaPadula [6] policies which make use of richer lattice structures obtained from the Cartesian product between a set of security levels and a powerset of security categories to ensure data integrity and confidentiality, respectively. The same monotonic scheme is used for the Biba policy while for Bell-LaPadula the scheme for non-monotonic access trees proposed by Ostrovsky et al. in [3] is more appropriate because we need to be able to express negations. We will make use of a combination between the two ABE schemes in order for filtering out of public parameters to be possible as well.

Security of both read and write operations will be proven, as well as collusion resistance. We will make use of the results obtained by the authors of the ABE schemes and also of the decisional Diffie-Hellman and Bilinear Diffie-Hellman assumptions.

Finally, we will highlight the limitations concerning the enforcement of write operations at the category level in the case of the Bell-LaPadula policy. Forcing encryption with a set of attributes requires significant changes to either of the ABE schemes and we will leave it as an open problem.

2. PRELIMINARIES

We will first start by describing some introductory notions regarding mandatory policies and attribute-based encryption which we will later use to define our solution for cryptographically enforcing mandatory policies.

2.1. Mandatory access control policies

A predictable and strictly controlled flow of information is crucial when it is required to ensure properties such as data confidentiality or integrity. For this reason, mandatory policies are usually designed alongside a lattice such that the information flow will mirror the structure of the lattice. These lattice-based access control policies assign security classes (clearances and labels) to objects and subjects. The flow of information will then be regulated according to this association.

Dorothy Denning has formally defined the information-flow model in [4] as being the tuple $FM = < N, P, SC, \oplus, \rightarrow >$ where:

- $N = \{a, b, ...\}$ is the set of objects in the system. It includes the active objects represented by the users.
- $P = \{p, q, ...\}$ i s the set of processes running on behalf of the users.
- $SC = \{A, B, ...\}$ is the finite set of security classes. Each object $o \in N$ and each process $p \in P$ are associated with a security class from SC.

- $\oplus$: $SC \times SC \rightarrow SC$ is a binary operator for combining security classes that is commutative and associative. Its application to any two security classes A and B results in a security class which contains information obtained from both A and/or B.
- $\rightarrow$ is defined over the elements of the Cartesian product $SC \times SC$ and represents the flow relation. $A \rightarrow B$ indicates that information from objects associated with the A security class can be transferred to objects whose security class is B. Flow policies are defined in terms of the $\rightarrow$ relation and any information flow model is considered secure if the execution of any chain of operations results in a flow which adheres to the policy's definitions.

Denning has also defined a set of axioms for which the tuple $< SC, \rightarrow, \oplus, \otimes >$ forms a bounded lattice. Such a lattice comprises of a finite partially ordered set that has both a lower bound and an upper bound relative to the flow relation $\rightarrow$.

The principles are the following:

1. $< SC, \rightarrow >$ is a partially ordered set, otherwise the flow relation $\rightarrow$ is inconsistent:
   - *reflexivity*: information is permitted to flow from an object to itself.
   - *transitivity*: transferring information from a class A to a class C through an intermediary class B is equivalent to transferring information from A directly to C.
   - *anti-symmetry*: if information can flow between two security classes A and B in both directions then the classes are redundant and thus the same.
2. SC is a finite set. This is a reasonable requirement for any practically implemented system.
3. The class combining operator $\oplus$ joins any two classes $A, B \in SC$ into their least upper bound and is totally defined over SC. These two properties imply that $\oplus$ can be used to compute the upper bound of the entire lattice denoted H.
4. The application of the $\otimes$ operator to any two classes $A, B \in SC$ results in their greatest lower bound. It can be used to obtain the lower bound of the lattice denoted L, which is equivalent to the publicly available information in a system or the empty set $\emptyset$ if no such information is contained within the system in question.

Richer and more expressive information flow policies can be obtained by combining two or more lattices using the Cartesian product.

Any policy defined over the resulting lattice structure will permit the flow of information only if the corresponding flows are permitted in all of the individual lattices.

In other words, the overall flow relation is described as a logical *AND* over the flow relations of the lattices that were part of the Cartesian product: $\rightarrow = \rightarrow_1 \wedge \rightarrow_2 \wedge \cdots \wedge \rightarrow_n$.

The most common use of such combinations are mandatory policies defined over a set of security *levels* together with a set of security *categories*. The security classes defined by Denning are now formed from an authorization level and a subset of categories and the relation between them is usually called *dominance*. The information flow relation is defined as follows:

$$A \rightarrow B \Longleftrightarrow level(A) \leq level(B) \text{ AND } categories(A) \subseteq categories(B)$$

Formally, the security levels correspond to a lattice over a totally ordered set, while the power 3 set of categories along with the include operator $\subseteq$ form a lattice over a partially ordered set.

The Biba [5] and Bell-LaPadula [6] policies make use of such Cartesian products in order to ensure data integrity and respectively confidentiality.

2.2. Attribute-based encryption

This section reviews some of the formal definitions introduced and used by the authors of [2] and [3] to construct and prove the security of ABE schemes for both monotonic and non-monotonic access structures.

2.2.1. Access structures

Definition 2.1. If $\{P_1, P_2, ..., P_n\}$ is a set of participants, then an access structure is a collection $A \subseteq 2^{\{P_1, P_2, ..., P_n\}} \setminus \{\emptyset\}$ of non-empty subsets of $\{P_1, P_2, ..., P_n\}$. The access structure is considered monotone if: $B \in A$ *and* $B \subseteq C \Rightarrow C \in A$, $\forall B, C$.

The sets in A are called the authorized sets of participants, while those not in A are the unauthorized sets of participants.

In the context of ABE, the attributes will play the role of the participants and this means that A will encompass the authorized sets of attributes.

2.2.2. Algorithms of attribute-based encryption schemes

A key-policy attribute-based encryption scheme as designed by the authors of [2] is composed from the following four algorithms:

- **Setup**: A non-deterministic algorithm that takes as input the security parameter and outputs the public parameters *PK* and the master key *MK*.
- **Encryption**: A non-deterministic algorithm that, given a message *m*, a set of attributes γ and the public parameters *PK*, computes the ciphertext *E* of *m*.
- **Key generation**: A non-deterministic algorithm that takes as input an access structure A, the master key *MK* and the public parameters *PK* and outputs a decryption key *D* that includes the access structure *A*.
- **Decryption**: A deterministic algorithm that given as input a ciphertext *E* of a message *m* encrypted with the set of attributes γ, a decryption key *D* associated with an access control structure A and the public parameters *PK*, outputs *m* if $\gamma \in A$, otherwise an error is thrown.

2.2.3. The selective-set model for ABE

The security against chosen plaintext attacks of an attribute-based encryption scheme is proven using the **selective-set model for ABE**, comprised of the following steps:

- **Init**: The adversary A declares a set of attributes γ upon which he will be challenged.
- **Setup**: The challenger runs the Setup algorithm of ABE and transfers the public parameters *PK* to the adversary.
- **Phase 1**: The adversary can issue requests for private keys for a set of access structures A_i, where $\gamma \notin A_i$, $\forall i$.
- **Challenge**: The adversary submits two messages m_0 and m_1 such that $|m_0| = |m_1|$. The challenger randomly chooses a bit $b = \{0, 1\}$ and encrypts m_b with γ. The resulting ciphertext is given to the adversary.
- **Phase 2**: Phase 1 is repeated.
- **Guess**: The adversary outputs a guess b^0 of b.

The advantage of A in this game is given by the probability $Pr[b' = b] - \frac{1}{2}$.

The selective-set model can also handle chosen ciphertext attacks if decryption queries are allowed in Phase 1 and Phase 2.

Definition 2.2. An attribute-based encryption scheme is secure in the selective-set model of security if all polynomial-time adversaries have at most a negligible advantage in the selective-set game.

2.2.4. Bilinear maps

If we consider $\mathbb{G}_1$ and $\mathbb{G}_2$ to be two multiplicative cyclic groups of prime order p, g a generator of $\mathbb{G}_1$ and e: $\mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ a bilinear map, then e exhibits the following characteristics:

- Bilinearity: $e(u^a, v^b) = e(u, v)^{ab}, \forall u, v \in \mathbb{G}_1$ and $a, b \in Z_p$.
- Non-degeneracy: $e(g, g) \neq 1$.
- Symmetry: $e(g^a, g^b) = e(g, g)^{ab} = e(g^b, g^a), \forall a, b \in Z_p$.

$\mathbb{G}_1$ is said to be a bilinear group if its corresponding group operation and the bilinear map e are both efficiently computable.

2.2.5. The decisional Bilinear Diffie-Helman (BDH) assumption

If a, b, c, z are randomly extracted from $\mathbb{Z}_p$ and g is a generator of $\mathbb{G}_1$, then the decisional BDH assumption states that there exists no probabilistic polynomial-time algorithm $\mathcal{B}$ that can distinguish the tuple $(A = g^a, B = g^b, C = g^c, e(g, g)^{abc})$ from the tuple $(A = g^a, B = g^b, C = g^c, e(g, g)^z)$ with more than a negligible advantage. The advantage of $\mathcal{B}$ is:

$$| Pr[\mathcal{B}(A,B,C,e(g, g)^{abc}) = 0] - Pr[\mathcal{B}(A,B,C,e(g, g)^z) = 0] |$$

where the probability is considered over the random choice of the generator g, the random choice of $a, b, c, z \in \mathbb{Z}_p$ and the random bits consumed by $\mathcal{B}$.

3. OUR SOLUTION

The central authority that defines the rules of mandatory policies also intervenes in both the setup phase of the system by assigning security classes to users and objects and also during normal operation for access control mediation and enforcement, very much like a reference monitor. This latter fact is disadvantageous because it requires users and/or their associated subjects to authenticate themselves before each access operation. Also, more importantly, the availability of the central authority is critical for the proper functioning of the system, which inevitably makes it a potential performance bottleneck and single point of failure. Additionally, even though such systems usually have their storage components placed in a protected area, the data is persisted in clear and that means it is still susceptible to theft in case the containers are compromised.

A more distributed approach to mandatory policies is therefore desirable so as to make possible access control enforcement without the continuous intervention of a reference monitor during the normal operation of the system. Having the data secured by an additional layer of protection such as encryption is also required. Both goals can be achieved by using attribute-based encryption: data can be encrypted and the private keys and the public encryption parameters can be generated and distributed in a manner that precisely enforces the lattice structures that underline these policies. As such, the central authority will only be used to initialize the system and to add users and resources. Access operations will no longer require mediation since access control will be a consequence of the employed encryption scheme, with subjects being able to access the storage containers directly.

The rest of the section describes how we can carry out this modeling, both for the general case of policies based on the information flow model as defined by Denning and, more specifically, for the Biba and Bell-LaPadula policies where the Cartesian product between security levels and security categories is used to obtain richer lattice structures.

3.1. Cryptographic enforcement of general mandatory policies

Let the tuple $FM = < S, O, SC, \oplus, \rightarrow >$ be an information flow model and the tuple $L = < SC, \rightarrow, \oplus, \otimes >$ a bounded lattice as defined by Denning, where O is the set of objects, S is the set of subjects acting on behalf of users, SC is the set of security classes, $\oplus$ is the class combining operator, $\otimes$ is the lower bound operator and $\rightarrow$ is the flow relation. We also consider $c: S \cup O \rightarrow SC$ a function that returns the security class associated with a subject or an object. The bounds imposed on the flow of information by a general lattice-based mandatory policy can then be defined in terms of *read* and *write* operations as follows:

- *read* operations: a subject $s \in S$ can read an object $o \in O$ if $c(o) \rightarrow c(s)$. The lattice structure must allow for information to flow from the security class of o to the security class of s.

- *write* operations: a subject $s \in S$ can write into an object $o \in O$ if $c(s) \rightarrow c(o)$. The lattice structure must allow for information to flow from the security class of s to the security class of o.

The semantics of the flow relation $\rightarrow$ will be given by the protection goals the policy sets out to achieve such as confidentiality in the case of the Bell-LaPadula model or integrity in the case of the Biba model. The general policy can accommodate any kind of restrictions over the information flow.

At a high level, general mandatory policies are cryptographically enforced through ABE by considering the security classes from SC as attributes. Each data object is encrypted with an attribute corresponding to its security class. Private keys are generated such that a user can decrypt an object only if information can flow from the security class of the object to the security class of the user. This is done by constructing each access tree to capture the security class of the corresponding user and the ones below it in the lattice structure. In other words, the tree will only be satisfied by security classes that flow into the class of the user, hence enforcing read access.

As far as write operations are concerned, if enforcement no longer relies on mediation then a malicious user can always choose to bypass encryption or use other communication channels to send information to unauthorized parties. However, we will only focus on preventing users from generating valid encryptions that would violate the bounds placed on the information flow by a policy. Thus, they should be able to encrypt data only with security classes that are above theirs in the lattice structure so that the flow of information is permitted from their class to the destination class.

We will use the ABE scheme for monotonic access structures defined by Goyal et. al in [2] for modeling general mandatory policies. Let $SC = \{sc_1, sc_2, ..., sc_n\}$ be a set of security classes and $U = \{1,2,...,n\}$ a universe of attributes. We consider a one-to-one mapping between them such that $i \in U$ is the attribute associated with the security class $sc_i \in SC$. We can now formally define two functions that given a security class construct an access tree and respectively generate encryption parameters for ABE for that class:

- **Access Tree Generation** (sc, L, U): Given a bounded lattice $L = < SC, \rightarrow, \oplus, \otimes >$, a security class $sc \in SC$ and a universe of attributes U, the function constructs an access tree T that will be satisfied by any security class equal to or below sc in the lattice structure. T will be used by the **Key Generation** from the attribute-based encryption scheme to produce a key that allows a user to decrypt an object only if information can flow from the security class of the object to the security class of the user.

  T will have a leaf node for each attribute $i \in U$ such that $sc_i \rightarrow sc, sc_i \in SC$. The leaves are then connected to an *OR* gate, which serves as the root node. Since the private key of any user belonging to the security class sc will not be satisfied by classes above sc in the hierarchy, users can not perform decryption operations that would result in a violation of the flow relation $\rightarrow$.

- **Encryption Parameters Generation** (*sc*, L, *PK*): Given a bounded lattice L = < *SC*, →, ⊕, ⊗ >, a security class *sc* ∈ *SC* and a set of public parameters generated by the **Setup** algorithm of the ABE scheme $PK = \{T_1, T_2, ..., T_n, Y\}$, the function filters out parameters such that encryption is only possible with attributes corresponding to security classes above *sc* in the lattice structure. The remaining parameters are randomized in order to prevent collusion and then given to a user associated with *sc*. He will be able to encrypt data with security classes to which information can flow from *sc*.

  If *x* is a value extracted uniformly at random from Z_p, the returned parameters will be:

$$EP_{sc} = \{Y^x, \{T_i^x \mid sc \rightarrow sc_i, sc_i \in SC\}\}$$

  Note that raising the public parameters to the power of a random value *x* is precisely what happens during the **Encryption** algorithm of the monotonic attribute-based encryption scheme. This guarantees that the randomization process does not affect the correctness of the cryptosystem: the random *s* value will simply be replaced by $x \cdot s$, nothing else changes. The users will be able to utilize the scheme in the same way as before.

  Since any user belonging to the class *sc* will not possess encryption parameters associated with the attributes for which the security classes are below *sc* in the hierarchy, he cannot encrypt with any attributes that would violate the flow relation →.
  Let L = < *SC*, →, ⊕, ⊗ > be a lattice where $SC = \{SC_1, SC_2, SC_3, SC_4\}$ such that
  $SC_1 \rightarrow SC_2 \rightarrow SC_3 \rightarrow SC_4$, as illustrated on the right.
  Let U = {1, 2, 3, 4} be a universe of attributes such that *i* is the attribute associated with the security class SC_i. Then the public key generated by the **Setup** algorithm of ABE is $PK = \{T_1, T_2, T_3, T_4, Y\}$.
  Encryption Parameters Generation (SC_3, L, *PK*) returns the encryption parameters
  $EP_{SC_3} = \{T_3^x, T_4^x, Y^x\}$ where *x* is a random value from Z_p.

  The tree generated by the **Access Tree Generation** (SC_3, L, U) function is illustrated on the right.

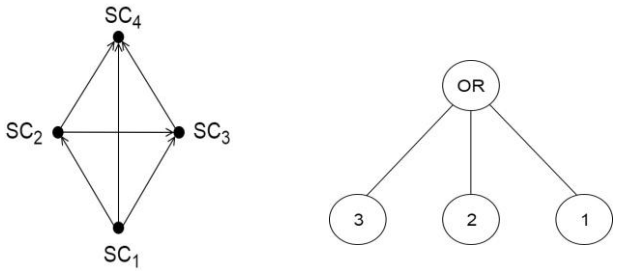

Fig. 1 – Example of access tree and encryption parameters generation for general mandatory policies.

Figure 1 illustrates an example of how an access tree is generated and how the public parameters are filtered out.

We will now describe how these two functions are used in a system that cryptographically enforces a general mandatory policy. During the setup phase, a trusted component associates each object and each user with a security class. The attribute-based cryptosystem is also initialized through the **Setup** algorithm using the security classes set as attributes. Data objects are then encrypted using ABE with the attributes that correspond to their security classes. For each user, his security class is used to generate an appropriate access tree with the **Access Tree Generation** function which will in turn be used to issue a private key for the attribute-based scheme. The security class is also used to filter out forbidden public parameters and randomize the remaining ones with the help of the **Encryption Parameters Generation** function. The private key and the remaining encryption parameters are then handed out to the user. At this point the initialization step is considered complete and users can perform read and write operations through decryption and encryption,

respectively, without the intervention of a reference monitor. The central authority will only perform actions when users and objects are added into the system.

We will now discuss the security of the proposed solution. We will prove that if the read operations lead to a violation of the information flow bounds then a simulator can be constructed that has a non-negligible advantage in the selective-set model security game for attribute-based encryption which, as proven in [2], ultimately leads to a contradiction of the decisional BDH assumption. We also show that if write operations that do not follow the policy can occur, we can build a simulator that can play the simple decisional Diffie-Hellman (DDH) game with a non-negligible advantage, which would contradict the DDH assumption.

Security of read operations. Failure to properly enforce read activities implies that a subject $s \in S$ associated with a security class $c(s)$ can decrypt an object $o \in O$ associated with a security class $c(o)$ in spite of the fact that $c(o) \rightarrow c(s)$ does not hold in the lattice L. In ABE terms, this means that decryption is possible even if the attribute with which the ciphertext is encrypted ($c(o)$) does not satisfy the access tree T generated by calling the **Access Tree Generation** function with the $c(s)$ security class, tree based on which the private key was generated.

Let A be an adversary that can perform such decryptions with a non-negligible probability. We show how we can build a simulator S that uses A to attack the ABE scheme in the selective-set model with non-negligible advantage.

- **Init**: The challenger C generates the groups G_1 and G_2 with an efficient bilinear map $e: G_1 \times G_1 \rightarrow G_2$, a generator g and the universe of attributes U. The simulator S sets a bounded lattice $L = < SC, \rightarrow, \oplus, \otimes >$ such that each $i \in U$ is the attribute associated with the security class $sc_i \in SC$.

  S then declares $\gamma = \{x\}$, $x \in$ U the set of attributes it wants to be challenged upon such that $\exists\, sc_i \in SC$ with $sc_x \neq sc_i$ for which $sc_x \rightarrow sc_i$ does not hold. In other words, γ contains an attribute corresponding to a security for which there exists at least one other class below it in the lattice structure. If we assume that the set of security class SC is not trivial and contains more than one element then at least such one element is guaranteed to exist because of the anti-symmetry property of the partial order relation $\rightarrow$. An example of a security class that satisfies this property is the upper bound of the lattice obtained using the class combining operator: $SC_1 \oplus \cdots \oplus SC_n$

- **Setup**: The challenger initializes the ABE cryptosystem to obtain the master key $MK = \{t_1, t_2, ..., t_n, y\}$ and the public key $PK = \{T_1 = g^{t_1}, T_2 = g^{t_2}, ..., T_n = g^{t_n}, Y = e(g, g)^y\}$. The simulator receives PK, which in turn gives it to the adversary, together with the set γ and the security class sc_x.

- **Phase 1**: S issues queries to the challenger for private keys associated with access trees generated by calls to the function **Access Tree Generation** with security classes $sc_i \in SC$ such that $sc_x \nrightarrow sc_i$. The fashion in which the attribute x was chosen in the **Init** step assures us that SC contains at least one such class. The selection criteria for sc_i and the definition of the **Access Tree Generation** function lead to the conclusion that γ will not satisfy any of the generated trees.

  All the user keys that the simulator receives will be given to A.

- **Challenge**: The simulator S generates two challenge messages m_0 and m_1 and sends them to C. The challenger extracts uniformly at random a bit $b \in \{0, 1\}$ and encrypts m_b with γ. The resulting ciphertext E is given to S which then sends it to the adversary A.

- **Phase 2**: **Phase 1** is repeated.

- **Guess**: A will try to decrypt the ciphertext E despite the fact the access trees associated with the keys he obtained during **Phase 1** and **Phase 2** are not satisfied by the set of attributes γ. If decryption is successful, the adversary will transfer the obtained message m to S which compares it to m_0 and m_1 and gives the challenger a guess b^0 of b such that $m_{b^0} = m$. In case the adversary fails to decrypt E he sends $\perp$

to S. The simulator then extracts the guess b^0 uniformly at random from {0, 1} and gives it to the challenger.

A has, by definition, the probability to decrypt E. If decryption is successful then the simulator can tell for sure which challenge message was encrypted. Thus, the probability of guessing b in this case is equal to ϵ. On the other hand, if decryption is unsuccessful then the probability of guessing b is $\frac{1}{2}$ since the simulator extracts the guess b^0 uniformly at random from {0, 1}.

The overall probability for S to guess b is $[b' = b] = \epsilon + \frac{1}{2}$. The selective-set model for attribute-based encryption specifies that the advantages of the simulator S is $Pr[b' = b] - \frac{1}{2} = \epsilon + \frac{1}{2} - \frac{1}{2} = \epsilon$.

Goyal et al. have proven that S can be used to build another simulator that plays the decisional BDH game with advantage $\frac{\epsilon}{2}$. The fact that is non-negligible leads to a contradiction of the decisional BDH assumption.

Security of write operations. Erroneous enforcement of write activities means that a subject $s \in S$ associated with a security class $c(s)$ can encrypt data with an attribute k corresponding to a security class sc_k in spite of the fact that $c(s) \rightarrow sc_k$ does not hold in the lattice L. Considering the definition of the **Encryption Parameters Generation**, this scenario is equivalent to saying that a user can create a valid encryption in the ABE scheme with an attribute for which he does not possess the corresponding encryption parameter. This means that the user can somehow determine $T_k^x = g^{x \cdot tk}$ despite the fact that $T_k^x \notin EP_{c(s)} = \{Y^x = e(g, g)^{x \cdot y}, \{T_i^x = g^{x \cdot ti} \mid c(s) \rightarrow sc_i, sc_i \in SC\}\}$.

Since the value of the generator g is publicly known, the user is left to find out the value $x \cdot t_k$. We will focus on the probability of computing t_k. Note that the values x, y and $t_i, \forall i \in$ U are chosen uniformly at random from Z_p and are thus independent of one another. This means that the user gains no information whatsoever about t_k from the encryption parameters already in his possession. In this case, the only way t_k can be determined is by randomly guessing it from the entire domain Z_p. This is made infeasible by choosing an appropriate security parameter κ so that p is sufficiently large and thus ensure security. Should the probability to guess t_k still be non-negligible then the other elements of the private key MK from the ABE scheme can also be determined since they are no different from t_k from a statistical point of view. This would allow the user to generate private keys that can decrypt any kind of ciphertext, irrespective of the attributes they are encrypted with. We have already seen that such decryptions lead to a contradiction of the decisional BDH assumption.

There is, however, a scenario in which the user/subject can own a value that is dependent on t_k. This happens when $sc_k \rightarrow c(s)$ because, based on the construction of the **Access Tree Generation** function, the attribute k will be part of the user's private key. Thus, the user will possess the value $D_x = g^{\frac{q_x(0)}{t_k}}$ where x is the leaf node associated with k: $k = att(x)$. q_x is the polynomial computed for the leaf node by the **Key Generation** algorithm from the ABE scheme. Taking into account that the points of the polynomial and y are chose uniformly at random from Z_p we conclude that the value $q_x(0)$ is also random. We now prove that if there exists an adversary A that can determine t_k with non-negligible probability if it possesses $g^{\frac{q_x(0)}{t_k}}$, then a simulator can be constructed that uses A to play the decisional Diffie-Hellman game with a non-negligible advantage.

The decisional Diffie-Hellman (DDH) assumption states that, given a multiplicative cyclic group G of order p and g a generator for G, there exists no probabilistic polynomial-time algorithm that can distinguish the tuple $(A = g^a, B = g^b, g^{ab})$ from the tuple $(A = g^a, B = g^b, g^c)$ with more than a negligible advantage, where a, b and c are chosen randomly and independently from Z_p.

Let S be a simulator that uses A to play the DDH game as follows:

- The challenger C determines the group G of prime order p and with generator g. It then extracts uniformly at random from Z_p the values a, b and c. A bit $u \in \{0, 1\}$ is also randomly chosen. If $u = 0$, C

sets the tuple ($A = g^a, B = g^b, Z = g^{ab}$), otherwise it sets ($A = g^a, B = g^b, Z = g^c$). The tuple ($A, B, Z$) is then given to the simulator S.

- S gives A from the tuple to the adversary A.

- $A = g^a$ can be seen as $g^{\frac{1}{\frac{1}{a}}}$. We know that A has, by definition, the probability of determining the denominator in such a scenario. It gives the simulator S the value $\frac{1}{a}$ if the computation was successful or $\perp$ to indicate that it failed.

- If the simulator receives $\frac{1}{a}$ from A it then computes $B' = Z^{\frac{1}{a}}$ and compares the result to B. If $B' = B$ it sends to the challenger C a guess $u' = 0$ of u and $u' = 1$ otherwise. If it instead receives $\perp$ from the adversary, S extracts u' uniformly at random from {0, 1} and gives it to C.

When $Z = g^{ab}$, if the adversary determined $\frac{1}{a}$ then $B' = Z^{\frac{1}{a}} = g^{ab\frac{1}{a}} = g^b = B$. This means that in this case, the probability for the simulator to guess u is the probability of A to determine the denominator.

If the simulator receives $\perp$ from the adversary, then the probability to guess u is $\frac{1}{2}$ due to the random choice of u'.

The overall probability of S to guess u is $Pr[u' = u] = \epsilon + \frac{1}{2}$ which results in the advantage of the simulator in the DDH game to be: $Pr[u' = u] - \frac{1}{2} = \epsilon + \frac{1}{2} - \frac{1}{2} = \epsilon$.

Thus, the existence of such an adversary A contradicts the decisional Diffie-Hellman assumption.

It is also desirable that read and write operations are secure against collusions. If two or more users come together with their respective private keys and encryption parameters, they should not be able to decrypt messages or encrypt with attributes for which it would be impossible for them to do so on their own. Read operations rely on the fact that the private keys of the attribute-based encryption scheme are collusion resistant themselves.

As far as write operations are concerned, it should not be possible for encryption parameters belonging to different users to be used together to create valid encryptions, unless the parameters are associated with the same attributes. This is ensured by the randomization step of the **Encryption Parameters Generation** function. The encryption parameters will be different from user to user and combinations between them will make the interpolation at the exponent during decryption impossible. Note that such parameters can be combined if they are associated with the same attribute but this is not an issue since it implies that the colluding users can already encrypt with the attribute in question, meaning that no advantage is gained.

Formally, we consider two users u_1 and u_2 with their respective encryption parameters $EP_1 = \{Y^{x_1}, \{T_i^{x_1} | c(u_1) \rightarrow sc_i, sc_i \in SC\}\}$ and $EP_2 = \{Y^{x_2}, \{T_j^{x_2} | c(u_2) \rightarrow sc_j, sc_j \in SC\}\}$.

Collusion between them implies that they can compute the triple (Y^z, T_i^z, T_j^z) where $i \neq j$, $T_j^{x_1} \notin EP_1, T_i^{x_2} \notin EP_2, z \in \mathbb{Z}_p$ and thus they can encrypt with both attributes i and j despite them not being able to do so on their own.

We consider the case when u_1 and u_2 have one encryption parameter each for different attributes. We prove that if there exists an adversary A that has a non-negligible probability to use the tuple

$$(Y^{x_1} = e(g,g)^{x_1 \cdot y}, T_i^{x_1} = g^{x_1 \cdot t_i}, Y^{x_2} = e(g,g)^{x_2 \cdot y}, T_j^{x_2} = g^{x_2 \cdot t_j})$$

to obtain ($Y^z = e(g, g)^{z \cdot y}$, $T_i^z = g^{z \cdot t_i}$, $T_j^z = g^{z \cdot t_j}$) where x_1, x_2, t_i, t_j are extracted uniformly at random from Z_p, $i \neq j$ and $z \in Z_p$ then we can construct a simulator S that uses A to play the decisional BDH game with non-negligible advantage. The steps are the following:

- The challenger C generates the groups G_1 and G_2 with an efficient bilinear map e: $G_1 \times G_1 \rightarrow G_2$ and a generator g for G_1. It then extracts uniformly at random from Z_p the values a, b, c and k. A bit $u \in \{0, 1\}$

is also randomly chosen. If $u = 0$, C sets the tuple $(A = g^a, B = g^b, C = g^c, Z = e(g, g)^{abc})$, otherwise it sets $(A = g^a, B = g^b, C = g^c, Z = e(g, g)^k)$. The tuple (A, B, C, Z) is then given to the simulator S.

- *S* sets the values $(Z, A^{t_i}, e(B^{t_j}, C), B^{t_j}) = (Z, g^{a \cdot t_i}, e(g,g)^{t_j \cdot bc}, g^{t_j \cdot b})$ where t_i and t_j are randomly chosen from Z_p. The tuple is then sent to the adversary A.

- The adversary tries to use the tuple $(Z, g^{a \cdot t_i}, e(g, g)^{t_j \cdot bc}, g^{t_j \cdot b})$ to compute the values $(e(g, g)^{z \cdot bc}, g^{z \cdot t_i}, g^{z \cdot b})$. Note that the probability to do so is only if $Z = e(g, g)^{abc}$, as per the definition of A. It then sends the attempt (A', B', C') to S.

- The simulator performs two checks to confirm whether the adversary's attempt was successful or not:
  - Compute $e(C', C)$ and compare the result to A'.
  - Compute $e(B', B)$ and compare the result to $e(g, C'^{t_i})$.

If both hold then the simulator sets $u' = 0$ and gives it to the challenger as a guess of u. Otherwise, the challenger receives $u' = 1$. This is because the above comparisons return true only if the challenger successfully computes the values $(A' = e(g, g)^{z \cdot bc}, B' = g^{z \cdot t_i}, C' = g^{z \cdot b})$ which means that $Z = e(g,g)^{abc}$:

$$e(C', C) = e(g^{z \cdot b}, g^c) = e(g, g)^{z \cdot bc} = A'$$

and

$$e(B', B) = e\left(g^{z \cdot t_i}, g^b\right) = e(g, g)^{z \cdot t_i \cdot b}$$

$$e(g, C'^{t_i}) = e(g, (g^{z \cdot b})^{t_i}) = e(g, g^{z \cdot b \cdot t_i}) = e(g, g)^{z \cdot t_i \cdot b}$$

When the challenger sets $u = 0$ the adversary has, by definition, the probability ϵ of successfully computing the required values. This means that the probability for *S* to guess u is either ϵ if *A*'s attempt is successful or $\frac{1}{2}$ otherwise. That means that the probability to guess u when it is 0 is:

$$Pr[u' = u | u = 0] = \epsilon + \frac{1}{2}.$$

When $u = 1$ the adversary does not gain any information to compute the tuple since $Z = e(g, g)^k$. This means that the probability for the simulator to guess u is: $Pr[u' = u | u = 1] = \frac{1}{2}$.

The overall advantage of the simulator *S* in the decisional BDH game is thus:

$$\begin{aligned} Pr[u' = u] - \frac{1}{2} = \frac{1}{2} Pr[u' = u | u = 0] + \frac{1}{2} Pr[u' = u | u = 1] - \frac{1}{2} = \\ = \frac{1}{2}(\epsilon + \frac{1}{2}) + \frac{1}{2}\frac{1}{2} - \frac{1}{2} \\ = \frac{\epsilon}{2} \end{aligned}$$

We conclude that if the probability of the adversary is non-negligible then the advantage of the simulator *S* in the decisional BDH game is also non-negligible which contradicts the BDH assumption.

3.2. Cryptographic enforcement of the Bell-LaPadula and Biba policies

The Biba and Bell-LaPadula policies combine security levels with security categories in order to enforce data integrity and confidentiality, respectively. The underlying lattice structures $L = (SC, \geq)$ are the result of

the Cartesian product between a total order over the set of security levels (L, $\leq$) and a partial order over the power set of the set of categories ($\mathrm{P}(C)$, $\subseteq$) where $SC = L \times \mathrm{P}(C)$ and the dominance relation $\geq$ is defined as follows:

$$sc_1 = (l_1, C_1) \geq sc_2 = (l_2, C_2) \Longleftrightarrow l_2 \leq l_1 \wedge C_2 \subseteq C_1$$
$$\forall sc_1, sc_2 \in SC, \forall l_1, l_2 \in L, \forall C_1, C_2 \in \mathrm{P}(C)$$

Whilst both models can be cryptographically enforced using the general technique described in the previous section that maps security classes to attributes, we would lose the granularity offered by the fact that a security class is composed from a security level and a subset of categories. Moreover, if we consider $|L| = n$ and $|C| = m$ then $|SC| = n \cdot 2^m$. This implies a huge number of attributes and also large sizes for the ABE user keys. We therefore want to map the security levels and the security categories directly to attributes. This not only considerably decreases the size of the universe of attributes ($n+m$), but it also allows us to construct richer and more expressive access trees while at the same time exploiting the possibility of encrypting data with several attributes: a security level and multiple categories.

Let $L = \{l_1, ..., l_n\}$ be a set of security levels, $C = \{c_1, ..., c_m\}$ a set of security categories and the universe of attributes $\mathrm{U} = \{1, ..., n, n + 1, ..., n + m\}$. The one to one mapping from security levels and categories to attributes is done by the bijective function $a: L \cup C \rightarrow \mathrm{U}$. We can now update the **Access Tree Generation** and **Encryption Parameters Generation** functions to support security levels and security categories in order to enforce read and write operations according to the security rules of either Biba or Bell-LaPadula policies.

3.2.1. Modeling the Biba policy

In the Biba model, security classes are replaced by integrity classes: $\mathrm{L} = (IC, \geq)$ where $IC = L \times \mathrm{P}(C)$. If we consider $\beta : S \cup O \rightarrow IC$ a function that returns the integrity class of a subject or an object and the sets of subjects and objects S and respectively O, the information flow regulated by the two properties:

- **Simple-integrity property**: A subject $s \in S$ can read data from an object $o \in O$ only if $\beta(o) \geq \beta(s)$.
- **Integrity *-property**: A subject $s \in S$ can write data to an object $o \in O$ only if $\beta(s) \geq \beta(o)$.

Using the same ABE scheme for monotonic access structures, the two functions can be defined as follows:

- **Access Tree Generation** (ic, L, U): Given a bounded combined lattice L, a universe of attributes U and an integrity class $ic = (l, C_{ic}) \in IC$, the function constructs an access tree T that will be satisfied by any integrity class for which its security level dominates l and its subset of categories includes C_{ic}. T will be used by the **Key Generation** from the attribute-based encryption scheme to produce a key that allows a user to decrypt an object only if information can flow from the integrity class of the object to the integrity class of the user ($\beta(o) \geq \beta(s)$).

  The following steps are required to build T:

  – Let T_L be the subtree of T that covers the security levels. T_L has a leaf node for each attribute $a(l_i)$ such that $l \leq l_i, \forall l_i \in L$. The leaves are then connected to an *OR* gate, which serves as the root node of T_L.
  – Let T_C be the subtree of T that covers the security categories. T_C has a leaf node for each attribute $a(c_j)$ associated with a category $c_j \in C_{ic}$. The leaves are then connected to an *AND* gate, which serves as the root node of T_C.
  – The subtrees T_L and T_C are both connected to an *AND* gate, which serves as the root node of T.

  Because the **simple-integrity property** requires the integrity class of the object to dominate that of the subject in order for read access to be permitted, the private key obtained from T will allow for decryption of messages encrypted with a security level above or equal to l and at least all the categories found in C_{ic} (to model the inclusion relation).

- **Encryption Parameters Generation** (*ic*, L, *PK*): Given a bounded lattice L, an integrity class $ic = (l, C_{ic}) \in IC$ and a set of public parameters generated by the **Setup** algorithm of the ABE scheme $PK = \{T_1, ..., T_n, T_{n+1}, ..., T_{n+m}, Y\}$, the function filters out parameters such that encryption is only possible with attributes corresponding to security levels below l and categories found in the set C_{ic}. This is because **integrity *-property** requires the integrity class of the subject to dominate that of the object in order for write operations to be allowed. The remaining parameters are randomized in order to prevent collusion and then given to a user associated with *ic*.

  If x is a value extracted uniformly at random from Z_p, the returned parameters will be:

$$EP_{ic} = \{Y^x,\ \{T^x_{a(l_i)} \mid l_i \le l,\ l_i \in L\},\ \{T^x_{a(c_j)} \mid c_j \in C_{ic}\}\}$$

  We have already seen that raising the public parameters to the power of a random value x does not affect the attribute-based encryption scheme in any way.

  Since any user belonging to the class *ic* will not possess encryption parameters associated with the attributes for which the security levels are above l and the categories are not included in C_{ic}, he cannot encrypt with any attributes that would violate the **integrity *-property**.

Figure 3 illustrates an example of how an access tree is generated and how the public parameters are filtered out in the case of the Biba model.

As far as security and resistance to collusion are concerned, the discussions and proofs from the enforcement technique for general mandatory policies also apply here.

3.2.2. Modeling the Bell-LaPadula policy

The policy sets out to ensure data confidentiality and is defined over a set of subjects S and a set of objects O. If we consider $\alpha : S \cup O \rightarrow SC$ a function that returns the security class assigned to a subject or an object, the flow of information in the Bell-LaPadula mandatory policy is governed by the two following rules:

- **Simple-security property**: A subject $s \in S$ is allowed to read the information contained in an object $o \in O$ if $\alpha(s) \ge \alpha(o)$.

- ***-property**: A subject $s \in S$ is allowed to write data into an object $o \in O$ if $\alpha(o) \ge \alpha(s)$.

Before describing the **Access Tree Generation** and **Encryption Parameters Generation** functions, we note that the access trees for a security class $sc = (l, C_{sc})$ should not be satisfied if the ciphertext is encrypted with an attribute corresponding to a category that is not found in C_{sc}. This means that in order to avoid duplication of attributes (to include the negations of each attribute) and ciphertexts of large sizes (messages have to be encrypted with attributes corresponding to the negation of the missing attributes), the non-monotonic ABE scheme [3] is more appropriate. However, it needs to be slightly modified to support filtering of encryption parameters. The alterations to the algorithms are the following:

- **Setup**(d): The encryption parameters corresponding to attributes are given by the functions $T, V : Z_p \rightarrow G_2$. This means that encryption is possible with any attribute $x \in \mathbb{Z}_p^*$. This approach does not lend itself well to preventing a user to create ciphertexts with particular attributes. For this reason we will consider a predefined universe of attributes $U = \{1, ..., n\}$ with $n \ge d$ and replace the function T (and the underlying polynomial $h(x)$) with the set of elements $\{T_i = g^{t_i}\}_{i \in U}$, where t_i are random elements from Z_p. The other elements are chosen in the same manner as in the initial solution. The public key then becomes:

$$PK = \{g, g_1 = g^{\alpha}, g_2 = g^{\beta} = g^{q(0)}, g^{q(1)}, ..., g^{q(d)}, T_1, T_2, ..., T_n\}$$

  The master key is $MK = \{\alpha, t_1, t_2, ..., t_n\}$.

- **Encryption** (m, γ, *PK*): The encryption of a message $m \in G_2$ with a set of d attributes $\gamma \subseteq U$ is:

$$E = \left(\gamma,\ E^{(1)} = m \cdot e(g_1, g_2)^s,\ E^{(2)} = g^s,\ \{E^{(3)}_x = T^s_x\}_{x \in \gamma},\ \{E^{(4)}_x = V(x)^s\}_{x \in \gamma}\right)$$

where $s \in Z_p$ is chosen uniformly at random.

• **Key Generation** ($\tilde{A}$, *MK*, *PK*): Only the components of the private key corresponding to positive attributes are changed. As such, for each unprimed attribute $x_i \in P$ we have:

$$D_i = g_2^{\frac{\lambda_i}{t_i}}$$

The rest of the private key is computed as in the scheme proposed by Ostrovsky et al.

• **Decryption** (*E*, *D*): The decryption process also changes only in relation to unprimed attributes. For each positive attribute $\tilde{x_i} \in \gamma'$ ($x_i \in \gamma$) the following element is evaluated:

$$Z_i = e(D_i, E_i^{(3)}) = e(g_2^{\frac{\lambda_i}{t_i}}, g^{s \cdot t_i}) = e(g_2, g)^{s\lambda_i}$$

The Z_i elements corresponding to negative attributes are computed as in the scheme for non-monotonic access structures [3], as well as the reconstruction of the encrypted message.

We can now define the two functions used to cryptographically enforce Bell-LaPadula policies:

• **Access Tree Generation** (*sc*, L, U): Given a bounded combined lattice L, a universe of attributes U and a security class $sc = (l, C_{sc}) \in SC$, the function constructs an access tree T that will be satisfied by any security class for which its security level is dominated by l and its subset of categories is included in C_{sc}. T will be used by the **Key Generation** from the attribute-based encryption scheme to produce a key that allows a user to decrypt an object only if information can flow from the security class of the object to the security class of the user ($\alpha(s) \geq \alpha(o)$).

The following steps are required to build T:

- Let T_L be the subtree of T that covers the security levels. T_L has a leaf node for each attribute $a(l_i)$ such that $l_i \leq l, \forall l_i \in L$. The leaves are then connected to an *OR* gate, which serves as the root node of T_L.
- Let T_C be the subtree of T that covers the security categories. T_C has a negated leaf node for each attribute $a(c_j)$ associated with a category that is not found in C_{sc}: $c_j \in C \setminus C_{sc}$. The leaves are then connected to an *AND* gate, which serves as the root node of T_C.
- The subtrees T_L and T_C are both connected to an *AND* gate, which serves as the root node of T .

Because the **simple-security property** requires the security class of the subject to dominate that of the object in order for read access to be permitted, the private key obtained from T will allow for decryption of messages encrypted with a security level lower or equal to l. As far as categories are concerned, decryption of messages encrypted with at least one category that is not found in C_{sc} will be denied. This is because the subset of categories belonging to the subject needs to include the subset of the object.

• **Encryption Parameters Generation** (*sc*, L, *PK*): Given a bounded lattice L, a security class $sc = (l, C_{sc}) \in SC$ and a set of public parameters generated by the **Setup** algorithm of the ABE scheme for non-monotonic access structures $PK = \{g, g_1 = g^{\alpha}, g_2 = g^{\beta} = g^{q(0)}, g^{q(1)}, \ldots, g^{q(d)}, T_1, T_2, \ldots, T_n, T_{n+1}, \ldots, T_{n+m}\}$, the function filters out parameters such that encryption is only possible with attributes corresponding to security levels above l. As far as categories are concerned, no parameters are filtered out because the requirement is not to prevent encryption with a certain category but to ensure that each ciphertext is associated with at least all the categories from C_{sc}. This is because the ***-property** requires the security class of the object to dominate that of the subject in order for write operations to be allowed. The remaining parameters are randomized in order to prevent collusion and then given to a user associated with *sc*.

If x is a value extracted uniformly at random from Z_p, the returned parameters will be:

$$EP_{sc} = \{g^x, g_1^x, g_2, g^{x \cdot q(0)}, g^{x \cdot q(1)}, \ldots, g^{x \cdot q(d)}, \{T^x_{a(l_i)} \mid l \leq l_i,\ l_i \in L\}, T^x_{n+1}, \ldots, T^x_{n+m}\}$$

The randomization process follows the principles of the encryption algorithm, the random element s is simply replaced by $x \cdot s$, everything else stays the same. This change is transparent for the encryption and decryption agents.

Since any user belonging to the class *sc* will not possess encryption parameters associated with the attributes for which the security levels are below l, he cannot encrypt with any attributes that would violate the **-property* relative to the security levels. However, the user should be allowed to encrypt with any attribute corresponding to a category as long as the ciphertexts include the categories from C_{sc}. This requires forcing the user to encrypt with a set of attributes which is not supported by neither ABE scheme described in [2] nor [3] without significant alterations. An idea would be to share a secret value between the encryption parameters corresponding to the attributes of the categories in C_{sc} such that share recovery and decryption is only possible if the encryption parameters in question are used. We leave further discussions regarding this matter as an open problem.

Figure 4 illustrates an example of how an access tree is generated and how the public parameters are filtered out and randomized in the case of the Bell-LaPadula model.

The proofs regarding security and collusion resistance follow a similar reasoning as the ones from the enforcement techniques of general mandatory policies. The randomization process is the same and the used ABE scheme is a mix between the two presented in [2] and [3] and both have been proven secure in the selective-set model.

Let L = ($IC, \geq$) be a lattice underlying a Biba policy where $IC = L \times P(C)$, $L = \{L_1, L_2, L_3, L_4\}$ such that $L_1 \leq L_2 \leq L_3 \leq L_4$ and $C = \{x, y, z\}$. The two lattice structures are illustrated below (Fig. 2).

Let U = $\{a(L_1), a(L_2), a(L_3), a(L_4), a(x), a(y), a(z)\}$ be a universe of attributes associated where $a : L \cup C \rightarrow U$ is a function that maps an attribute to each security level and category. Then the public key generated by the **Setup** algorithm of ABE is $PK = \{T_{a(L_1)}, T_{a(L_2)}, T_{a(L_3)}, T_{a(L_4)}, T_{a(x)}, T_{a(y)}, T_{a(z)}, Y\}$.

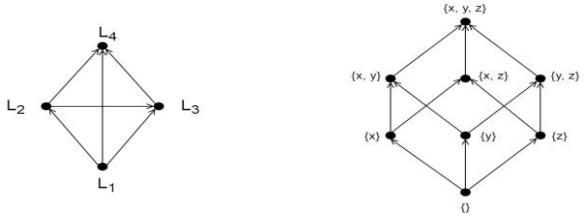

Fig. 2 – Lattice structure used in the ABE scheme.

Encryption Parameters Generation ($ic = (L_3, \{x, y\})$, L, *PK*) returns the encryption parameters

$$EP_{ic} = \{T^k_{a(L_1)},\ T^k_{a(L_2)},\ T^k_{a(L_3)},\ T^k_{a(x)},\ T^k_{a(y)},\ Y^k\}$$

where k is a random value from Z_p.

The tree generated by the **Access Tree Generation** ($ic = (L_3, \{x, y\})$, L, U) function is illustrated below.

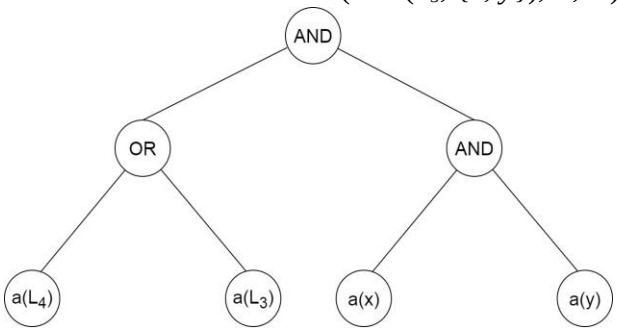

Fig. 3 – Example of access tree and encryption parameters generation for the Biba policy.

Let L = (SC, ≥) be a lattice underlying a Bell-LaPadula policy where SC = L × P(C), L = {L_1, L_2, L_3, L_4} such that $L_1 \leq L_2 \leq L_3 \leq L_4$ and C = {x, y, z}.

The two lattice structures are illustrated in Fig. 2.

Let U = {$a(L_1)$, $a(L_2)$, $a(L_3)$, $a(L_4)$, $a(x)$, $a(y)$, $a(z)$} be a universe of attributes associated where $a : L \cup C \rightarrow U$ is a function that maps an attribute to each security level and category. Then the public key generated by the **Setup** algorithm of ABE is

$$PK = \{g, g_1, g_2, g^{q(0)}, g^{q(1)}, ..., g^{q(d)}, T_{a(L_1)}, T_{a(L_2)}, T_{a(L_3)}, T_{a(L_4)}, T_{a(x)}, T_{a(y)}, T_{a(z)}\}$$

Encryption Parameters Generation (*sc* = (L_3, {*x*}), L, *PK*) returns the encryption parameters $EP_{sc} = \{g^k, g_1^k, g_2, g^{k \cdot q(0)}, g^{k \cdot q(1)}, \ldots, g^{k \cdot q(d)}, T^k_{a(L_3)}, T^k_{a(L_4)}, T^k_{a(x)}, T^k_{a(y)}, T^k_{a(z)}\}$ where *k* is a random value from Z_p.

The tree generated by the **Access Tree Generation** (*sc* = (L_3, {*x*}), L, U) function is illustrated below.

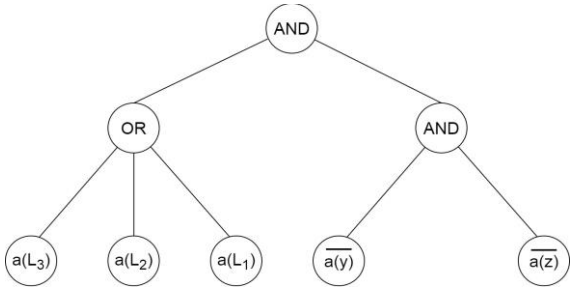

Fig. 4 – Example of access tree and encryption parameters generation for the Bell-LaPadula policy.

4. CONCLUSIONS

This paper proposes a solution for enforcing mandatory access control policies through cryptography. The motivation is that the central authority used to mediate each access request in the traditional approach inevitably represents a single point of failure. Any functional or performance issues at its level can badly hamper the normal use of the system. Added to that is the fact that it is desirable for data not to be stored in clear but to have it secured by a layer of protection. This is in order to mitigate any risks of theft in case the storage is compromised.

We have started by introducing the reader to some of the formal concepts regarding mandatory policies and attribute-based encryption.

We have then shown how we can model the lattice structures that underline mandatory policies in order to cryptographically enforce them through attribute-based encryption. Read operations are enforced by generating access trees that capture the lattice structure for a given security class, while write operations rely on randomization and careful distribution of the public parameters. We have used the monotonic ABE scheme [2] to enforce general mandatory policies based on the information flow model defined by Denning and the Biba policy concerned with data integrity.

We also tackle the secrecy-based Bell-LaPadula policy which requires the use of a combination between the monotonic and non-monotonic [3] ABE schemes. This is in order to be able to express negations as well as to filter out public parameters.

Security of both read and write operations is proven, as well as collusion resistance. We made use of the results obtained by the authors of the ABE schemes and also of the decisional Diffie-Hellman and Bilinear Diffie-Hellman assumptions.

Finally, we have highlighted the limitations concerning the enforcement of write operations at the category level in the case of the Bell-LaPadula policy. Forcing encryption with a set of attributes requires significant changes to either of the ABE schemes and we leave it as an open problem.

REFERENCES

1. B. W. LAMPSON, *Protection*, ACM SIGOPS Operating Systems Review, **8**, *1*, pp. 18-24, 1974.
2. V. GOYAL, O. PANDEY, A. SAHAI, B. WATERS, *Attribute-based encryption for fine-grained access control of encrypted data*, Proceedings of the 13th ACM Conference on Computer and Communications security (CCS 06), pp. 89-98, 2006.
3. R. OSTROVSKY, A. SAHAI, B. WATERS, *Attribute-based encryption with non-monotonic access structures*, Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS 07), pp. 195-203, 2007.
4. D. E. DENNING, *A lattice model of secure information flow*, Communications of the ACM, **19**, *5*, pp. 236-243, 1976.
5. K. J. BIBA, *Integrity Considerations for Secure Computer Systems*, MITRE Corp., 1977.
6. D. E. BELL, L. J. LA PADULA, *Secure Computer Systems: Mathematical Foundations*, MITRE Corporation, **1**, 1973.