DET-ABE: A Java API for Data Confidentiality and Fine-Grained Access Control from Attribute Based Encryption

Miguel Morales-Sandoval(✉) and Arturo Diaz-Perez

Laboratorio de Tecnologias de Informacion, CINVESTAV-LTI, Parque Cientifico Y Tecnologico de Tamaulipas, 87130 Ciudad Victoria, TAMPS, Mexico
{mmorales,adiaz}@tamps.cinvestav.mx

Abstract. Many works in the literature have proposed information security mechanisms relying on Paring Based Cryptography (PBC), for example, Ciphertext Policy Attribute Based Encryption (CP-ABE). However, a public set of software modules that allow integrating that kind of encryption for data security of information systems in an easy and transparent way is still missing. Available APIs like PBC (C-based) or jPBC (Java-based) are focused on low level arithmetic operations and several non trivial issues must still be addressed to integrate a functional PBC/ABE scheme into end-user applications for implementing end-to-end encryption. We present a novel and portable Java library (API) to ensure confidentiality and access control of sensitive data accessed only by authorized entities having as credentials a set of attributes. Novel encryption and decryption processes are defined, using the digital envelope technique (DET) under a client-server computing model. The new DET-ABE scheme supports standard security levels (AES encryption) and provides the user with an easy interface for transparent use of next generation cryptography, hiding the complexity associated to PBC (field and group arithmetic, curve selection) and ABE (setup, key management, encryption/decryption details). Running times of main API's modules at server (ABE setup and key generation) and client (DET-ABE encryption/decryption) side are presented and discussed. From these results, it is concluded that the proposed API is easy to use and viable for providing confidentiality and access control mechanisms over data in end-user applications.

Keywords: Pairings · Cryptographic API · Attribute based encryption

1 Introduction

Since ancient times, human beings have had the necessity to protect information in a way that only authorized entities have access to it. Nowadays, cryptographic schemes are vital to provide information security services to IT applications, such as confidentiality, integrity, authentication and non-repudiation [19].

© IFIP International Federation for Information Processing 2015
R.N. Akram and S. Jajodia (Eds.): WISTP 2015, LNCS 9311, pp. 104–119, 2015.
DOI: 10.1007/978-3-319-24018-3_7

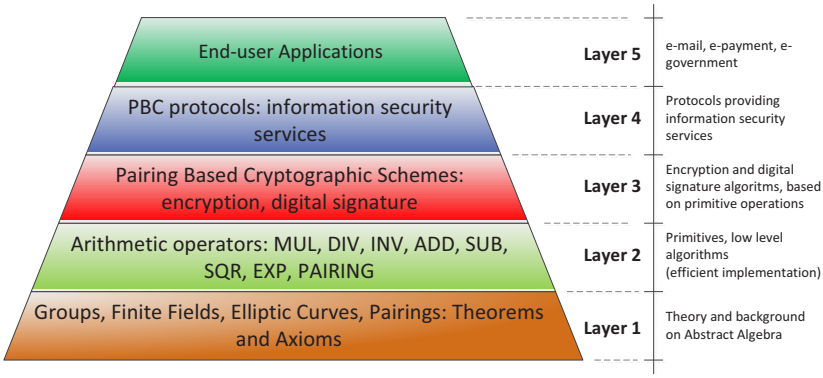

Fig. 1. Layered view of secure end-user applications based on Pairing Based Cryptography (PBC).

A relative novel field in cryptography is Pairing Based Cryptography (PBC) [5]. It is based on mathematical mappings defined on algebraic structures of abstract algebra. PBC provides elegant solutions to open problems of the past, such as Identity Based Encryption [6], where cryptographic operations are performed in function of the public identity of participants. Thus, under this approach digital certificates and their costly PKI (Public Key Infrastructure) are not required [7].

Research on PBC has increased recently. Currently, there are academic forums completely dedicated to it as the Pairings[1] and ECC[2] Conference series. A useful web resource for PBC is *The Pairing-Based Crypto Lounge* Barreto's website[3]. Attribute Based Encryption (ABE) [14,25] is a kind of public key encryption constructed over the foundation of PBC. ABE is a relatively new research topic, ideal to provide a fine-grained and non-interactive access control mechanism of encrypted data. Although several contributions on ABE exist at theoretical level, there are very few on the practical side [21].

Secure end-user applications as e-mail, e-payment, e-government, etc., can be seen as several layers, where the upper ones rely on the security and efficiency of the lower ones. Using PBC, the layered view of a secure application can be seen as shown in Figure 1.

In the past, some software modules have been published and made available for the community to serve as building blocks in the construction of applications at upper layers. Two of the most known are PBC [18] and jPBC [9]. Others for example, have been proposed to target specifically computing constrained devices [1,17,22,26]. However, these APIs are mainly focused for layer 2 of Figure 1 and provide some specific implementations for layer 3, leading to a gap for an easy incorporation of PBC and ABE in IT information systems

[1] http://www.ieccr.net/2013/pairing2013/
[2] http://www.eccworkshop.org
[3] http://www.larc.usp.br/~pbarreto/pblounge.html

at layer 5. Unlike other public key cryptosystems, as RSA [23], the mathematical background of PBC (Cyclic Groups, Finite Fields, Elliptic Curves, modular arithmetic, pairing computation,...) could be a complex subject for IT engineers without a cryptography specialization that want to use and integrate PBC-based protocols to enable security services in their applications. Going from layer 3 to layer 5 implies several important issues related to security and efficiency when using pairings in cryptography [13]. That decision taking could discourage IT engineers to use PBC.

This paper is mainly for IT engineers who are interested in using Pairing Based Cryptography and Attribute Based Encryption to provide security information services in IT systems. The aim of this work is to contribute to reduce the gap between the pairing based cryptographic foundation and the practical use of it. As main contribution, this work presents the design and implementation results of a set of classes written in Java that encapsulate the implementation details from layer 1 to layer 4 in Figure 1 (underlying algorithms, security parameters, type and size of the elliptic curve) and provide an easy interface to IT security implementers for using this type of next generation encryption at layer 5 in Figure 1.

The novel security modules proposed in this work are based on the Digital Envelope Technique (DET) [24]. In this work, an AES-key k used for bulk encryption [20] is protected by means of CP-ABE encryption [4], which is a specific type of ABE. Once encrypted, the encrypted data (with AES) together with the encrypted AES-key (with CP-ABE) can be securely distributed over insecure networks or stored in a honest but curious untrusted third party (i.e Cloud storage provider). Thus, DET allows to achieve confidentiality and CP-ABE enable fine-grained control access mechanisms. Only those authorized entities with a valid set of attributes could decrypt and recover the AES-key k to launch the AES decryption process over the encrypted data. Under this solution approach, typical applications as securing digital medical records or storing and sharing of digital documents in the Cloud could be easily implemented. As an application example, in this paper we provide a performance evaluation of the proposed security modules for encryption and decryption of digital documents (.doc and .pdf files). Due to the layered and highly modular design of the publicly available set of software modules, they can be further optimized to meet specific timing requirements.

The rest of this paper is organized as follows. Section 2 presents main aspects related to the use of pairings in cryptography as well as the general concept of Attribute Based Encryption. Section 3 describes the proposed approach for new DET-ABE encryption and decryption using pairings. Section 4 describes with details the design of the proposed set of Java classes that implement DET-ABE with PBC. Also, this section shows a practical use of the proposed API for confidentiality and access control of digital documents. Section 5 presents and discusses the running time of the main software modules for setup, encryption, decryption and key generation. Finally, section 6 gives the concluding remarks of this work and points out future work.

2 Foundations of PBC and ABE

A cyclic group $\mathbf{G}$ is a set of elements S together with a binary operation $\diamond$ where exists an element $g \in S$ such that for all $h \in S$, h can be obtained by applying the operation $\diamond$ over g $d-1$ times [11,15] (d is a positive integer number). This is denoted as $h = g^d$ using a multiplicative notation or $h = d \times g$ using an additive notation. The element g is named generator of $\mathbf{G}$ and written as $\mathbf{G} =< g >$. The most common cyclic group is $Z_r = \{0, 1, ..., r-1\}$, where r is a prime and $\diamond$ is the addition modulo r operation, being r the order of Z_r.

Let $\mathbf{G}_0 =< g_0 >$, $\mathbf{G}_1 =< g_1 >$, and $\mathbf{G}_T$ be cyclic groups of prime order r. A bilinear pairing or bilinear mapping is an efficient computable function $e : \mathbf{G}_0 \times \mathbf{G}_1 \rightarrow \mathbf{G}_T$, such that:

1. $\forall a, b \in Z_r, e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$
2. $e(g_0, g_1) \neq 1$

The tuple $(r, g_0, g_1, \mathbf{G}_0, \mathbf{G}_1, \mathbf{G}_T)$ defines an *asymmetric* bilinear pairing. If $\mathbf{G}_0 = \mathbf{G}_1 = \mathbf{G}$, and $\mathbf{G} =< g >$, the tuple $(r, g, \mathbf{G}, \mathbf{G}_T)$ defines a *symmetric* bilinear pairing [12].

In practice, many cryptographic protocols based on pairings have used a subset of points in an elliptic curve [13] as the cyclic group $\mathbf{G}$. An elliptic curve $E(F_q)$ is a set of pairs (x, y), with x, y elements of a finite field F_q satisfying an elliptic curve equation $E(x, y)$. The properties of such equation categorize the elliptic curve and determines its secure properties for practical use in cryptographic applications. So, in practice, pairings $\mathbf{G}_0$ and $\mathbf{G}_1$ are subgroups of the elliptic curve $E(F_q)$ and G_T is a subgroup of $F^*_{q^k}$ (an extension field of F_q). The number k is named the *embedded degree*. The security parameters to take into account when defining a paring over elliptic curves are: the size q of the finite field F_q, the embedding degree k, and the order l of $\mathbf{G}_0$, $\mathbf{G}_1$, and $\mathbf{G}_T$. These security parameters must be chosen carefully to ensure that the discrete logarithm problem is hard to compute in the three groups.

In our proposed set of security modules, the selection of this security parameters and the corresponding pairing is transparent to the programmers. The parameters selection is in function of the security level chosen by the programmer, which is one of those recommended by international standards, as the National Institute for Standards and Technology (NIST): 80-bit (low-term security, not recommended anymore), 128-bit (minimum security level), 192-bit (mid-term security), and 256-bit (long-term security) [2]. In our proposed DET-ABE scheme, programmers only need to specify the security level to use, the pairing security parameters will be selected internally.

2.1 Attribute Based Encryption

Firstly introduced by Sahai and Waters [25], Attribute-based Encryption (ABE) is a mean for complex access control on encrypted data. In this kind of cryptography, ciphertexts are not necessarily encrypted to one particular user as it occurs in traditional public key cryptography. Contrarily, ciphertexts and their corresponding

private keys that decrypt them are associated with a set of *attributes* or a policy over attributes. This way, if there is a match ciphertext-private key, the ciphertext will be decrypted by that private key. In [4], Betancourt et al., present Ciphertext-Policy Attribute-Based Encryption (CP-ABE), a method conceptually closer to traditional access control techniques such as Role-Based Access Control (RBAC). In CP-ABE attributes are used to describe users credentials, and an entity encrypting data determines a policy for who can decrypt. When an entity encrypts data, it specifies an associated *access structure* over attributes. This case, any other entity will only be able to decrypt a ciphertext if that entity's attributes pass through the ciphertexts access structure. What Betancourt et al proposed as access structure is a tree structure where its nodes represent threshold gates (AND, OR) and the leaves describe attributes. An AND gates is constructed as n-of-n threshold gate and an OR gate is a 1-of-n threshold gate. Generalizing, threshold gates are of the form m-of-n. Complex access controls including numeric ranges are also addressed by converting them to small access trees.

Basic modules in CP-ABE are constituted by four fundamental algorithms [4]:

1. **ABE-Setup:** Select an elliptic curve and the associated security parameters to define a pairing. As it was stated previously, it is the tuple (r,g_0,g_1,$\mathbf{G}_0$,$\mathbf{G}_1$, $\mathbf{G}_T$). With the pairing and associated cyclic groups, this module produces a public key PK and a secret master key MK.
2. **ABE-Encrypt:** The encryption algorithm uses PK to encrypt a message M under an access structure A over the universe of attributes U. As a result, the ciphertext is CT, which will be only decrypted by an entity possessing a set of attributes S that satisfies the access structure A. It is assumed that CT implicitly contains A.
3. **ABE-KeyGen:** The key generation algorithm uses MK to produce a private key SK, related to a specific set of attributes S.
4. **ABE-Decrypt:** The decryption algorithm uses PK and SK associated to a set of attributes S to decrypt a ciphertext CT, which contains an access structure A. If S satisfies A, this module will decrypt CT and return the original message M.

The main components of CP-ABE are described in Table 1, considering both symmetric [4] and asymmetric pairings [16].

The construction of the CP-ABE modules involve many computations over groups and finite field (some of them are shown in Table 1). In the proposed set of security modules, all these computations are encapsulated and hidden from user. As part of CP-ABE encryption and decryption, main operations include tree traversal, boolean function (policy) evaluation, hashing and creation of private keys by computing several pairings, polynomials generation and evaluation, and computation of Lagrange's coefficients. All those operations as well as the internal representation and the storing and recovery processes of main components in CP-ABE (PK, SK, CT, MK) are not required to be understood by CP-ABE users. In our proposal, these modules are treated as blackboxes.

Table 1. Main components in Ciphertext Policy Attribute Based Encryption (CP-ABE)

Component	Definition on a symmetric pairing	Definition on an asymmetric pairing
Public key PK	$\{g, h = g^{\beta}, e(g,g)^{\alpha}\}$	$\{g_0, g_1, h = g_0^{\beta}, e(g_0,g_1)^{\alpha}\}$
Master key MK	$\{\beta, g^{\alpha}\}$	$\{\beta, g_1^{\alpha}\}$
Private key SK	$\{D, d_j, d'_j\}$, where $D = g^{(\alpha+r)/\beta}$, $\forall j \in S$: $d_j = g^{r} \times H(j)^{r_j}$ $d'_j = g^{r^j}$ $r, r_j \in Z_r$ (random) $H : \{0,1\}^* \rightarrow G$	$\{D, d_j, d'_j\}$, where $D = g_1^{(\alpha+r)/\beta}$, $\forall j \in S$: $d_j = g_1^{r} \times H(j)^{r_j}$ $d'_j = g_0^{r^j}$ $r, r_j \in Z_r$ (random) $H : \{0,1\}^* \rightarrow G_1$
CipherText CT	$\{A, C', C\}$, where $C' = M \times e(g,g)^{\alpha s}$, $C = h^s$ $\forall$ leaf node $y \in A$, having $j \in S$, compute: $[C_y, C'_y]$, where $C_y = g^{q_y(0)}$ $C'_y = H(j)^{q_y(0)}$ $H : \{0,1\}^* \rightarrow G$	$\{A, C', C\}$, where $C' = M \times e(g_0,g_1)^{\alpha s}$ $C = h^s$ $\forall$ leaf node $y \in A$, having $j \in S$, compute: $[C_y, C'_y]$, where $C_y = g_0^{q_y(0)}$ $C'_y = H(j)^{q_y(0)}$ $H : \{0,1\}^* \rightarrow G_1$
Access Structure A	Tree structure where each internal node represents a k-of-n gate, and leaves represent attributes.	

3 Security Services from the Digital Envelope Concept with PBC and ABE

The digital envelope technique (DET) [24] is a method for key exchange, not used by all key exchange protocols [10]. DET is used generally for secure transporting of a session key, that is, a secret key to be used by a symmetric encryption algorithm for protecting all traffic exchanged by a sender and receiver in a communication session. In DET, the secret key is usually encrypted with public-key cryptography (PKC).

One advantage of DET is that end-users may switch secret keys as frequently as they would like. Switching keys often is beneficial because it is more difficult for an adversary to find a key that is only used for a short period of time. Another advantage of DET is the increase in performance which is obtained by using symmetric ciphers to encrypt the large and variably sized amount of message data, reserving PKC for encryption of short-length keys. In general, symmetric ciphers are much faster than public key cryptosystems [19].

So, in this work, data (text, image, sound, video,...) is encrypted using as symmetric cipher the Advanced Encryption Standard (AES) [20] in a way that the AES session key is protected and securely embedded in the ciphertext by using CP-ABE encryption. In this work, we re-define each main module in the

original CP-ABE scheme by Betancourt et al. in [4], leading to new building blocks for data encryption using the digital envelope technique with CP-ABE, we name it DET-ABE.

3.1 DET-ABE Setup

Setup involves the selections and settings for PBC and ABE. This includes the selection of the elliptic curve and the associated security parameters to define a pairing. The setup is performed in a *trusted third party*, that is responsible of generating and managing the public key *PK* and a secret master key *MK*.

In DET-ABE setup, the single parameter specified by the user is the security level to use in the encryption process. That security level is one recommended by the current standard for symmetric encryption, AES. According to NIST, security can be either minimum (128 bits), medium (192) or high (256). The elliptic curve and associated security parameters are internally selected to be consistent with the security level required. Table 2 shows the association of a given AES security level with a set of elliptic curves recommended for use in PBC and ABE. As it has been demonstrated and well documented in the literature [18], the best attacks over the groups with prime order r defining the pairing require $\sqrt{r}$ operations, so at least the order of $\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_T$ is twice the security level to achieve.

Table 2. Security settings

AES Security(bits)	Curve type	$\log_2 r$	Embedding degree
128 (minimum)	A (symmetric pairing)	256	2
192 (medium)	F (asymmetric pairing)	384	12
256 (high)	F (asymmetric pairing)	512	12

Pairing-based cryptographic settings given in Table 2 ensure that the discrete logarithm problem will be intractable in each group $\mathbf{G}_0, \mathbf{G}_1, \mathbf{G}_T$. In our construction, we have selected type A curves with embedding degree $k = 2$ for the security level of 128. This curve defines a symmetric pairings ($\mathbf{G}_0 = \mathbf{G}_1 = \mathbf{G}$). Also, the type F curves also known as Barreto-Naering (BN) curves [3] having embedding degree of 12 are used for the security levels 192 and 256 bits.

In this work, the generation of *PK* and *MK* are based on the definition given in [4] when using type A elliptic curves. For the case of type F curves defining an asymmetric pairing, *PK* and *MK* are derived as previously defined in [16].

As in CP-ABE, the DET-ABE setup module is executed in the trusted third party (server). As part of the DET-ABE setup process, a set of attributes U containing N distinct strings must be defined and administrated in the trusted entity.

3.2 DET-ABE Encryption

DET-ABE encryption is a client module of the trusted entity that encrypts a sequence of bytes (*data*) specified in a binary file. Main tasks in this module include:

1. Internally, an AES-key (k) is generated from a security level s given by the user.
2. With k, AES is used to encrypt *data* producing the ciphertext $\mathrm{CT}_{\mathrm{AES}}$.
3. Then, CP-ABE is used to encrypt k, given a policy P over a set of valid attributes S. Being P a boolean expression over S, it is evaluated to be logically well formed and the corresponding access structure A is generated. With A, k is encrypted using CP-ABE and the resulting ciphertext CT is stored together with $\mathrm{CT}_{\mathrm{AES}}$ and the policy A in a binary file.
4. For CP-ABE encryption, the client connects to the trusted third party (server) to retrieve the public key PK created during the DET-ABE setup module and associated to the security level s.
5. The policy P is specified by the client, and the attributes are retrieved and validated from the server.
6. The result is the tuple $T_E = \{CT_{\mathrm{AES}}, CT, A, s\}$.

The client executing the DET-ABE encryption module requires three elements: the *data* to encrypt, the security level s (see Table 2), and the policy P as a boolean expression of valid attributes S. The tuple T_E resulting from the DET-ABE encryption process can be either stored locally in the client side or uploaded to a public repository.

3.3 DET-ABE Decryption

The DET-ABE decryption module is used to decrypt previously encrypted data represented by the tuple $T_E = \{CT_{\mathrm{AES}}, CT, A, s\}$. The following tasks are performed during the execution of DET-ABE decryption:

1. The decryption client process requires T_E and the list L of user's attributes.
2. The decryption client starts a connection with the trusted third party (server), asking a private key from L. The client sends L to the server and the security level $s \in T_E$.
3. The server (trust party) validates the user's attributes L. The pairing parameters (curve type) associated to s are selected in the server side and the corresponding *private key SK* is computed by the server using those settings. SK is send back to the client together with the public key PK associated to s.
4. With SK, the client executes CP-ABE to decrypt $\mathrm{CT} \in T_E$, and recovers the session AES-key k, which is used to decrypt $\mathrm{CT}_{\mathrm{AES}} \in T_E$.

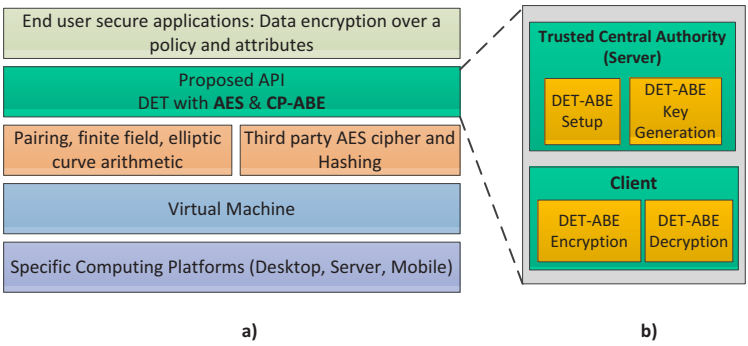

Fig. 2. a) Layered view of components for building secure applications using the digital envelope technique (DET) with AES and CP-ABE. b) Main components in the proposed API, executing the main tasks for DET-ABE data encryption/decryption under a client-server architecture.

4 Proposed API for DET-ABE

Our proposal is to integrate the concept of DET and CP-ABE in a set of software modules as a kind of middleware that allows programmers to build secure applications by mean of data encryption over a policy and a set of attributes. Figure 2 shows the layered view of modules needed to construct and execute secure applications based on the DET-ABE scheme proposed in this paper. The proposed set of security modules are written in Java and built on top of the jPBC library [9] that performs low level finite field, group and pairing computations. The use of Java allows a broader range of applications as the security scheme is able to be used over different platforms (server, desktop, mobile).

4.1 Attributes Management

Attributes management and how policies are constructed are dependent on the end-user application. Although attributes can be administrated in the trusted authority responsible for DET-ABE setup and key management, another trusted entity could be used specifically for attributes management. This entity (AA authority) should be responsible for registering and authenticating users in DET-ABE, either those that encrypt data (producers) or those that access encrypted data (consumers). When a user registers itself with the AA entity, a set of attributes are assigned to it, depending on the application specifications. For an authenticated producer U_p, it is the AA entity that authorizes the encryption operation by providing it the attributes required to construct the policy needed by DET-ABE encryption. In the case of an authenticated consumer U_c, the AA entity authorizes the decryption process by giving U_c its corresponding attributes, assigned according to the user application restrictions. Communication between the AA entity and the user must be secured for example using TLS/SSL.

4.2 The Server Side

The server modules run in a trusted central authority as specified in the CP-ABE scheme. Communication between the server and clients is secured by means of the SSL/TLS protocol. The server is able to manage the three security levels in Table 2, having a specific set of curve parameters for each of those. That selection is based on recommended elliptic curves in [9] and [18]. The server executes the DET-ABE setup module and generates CP-ABE private keys for clients executing DET-ABE decryption. The server keeps a pair $\{PK, MK\}$ for each security level supported. When clients connect to the server, they inform the security level to use and the server uses the correct curve parameters and keys. If keys are not already created for the demanded security level, the DET-ABE setup for that specific security level is launched.

4.3 The Client Side

A client properly authenticated in the AA entity can execute the DET-ABE encryption or decryption modules. In any case, a secure connection is established with the server at a specific port. During an encryption operation over a tuple T_E, the client asks the server for the public key PK associated to the security level $s \in T_E$. Also, the client constructs and validates the policy P. As PK is public, it is cached in the client side for future encryption operations using the same security level s. All the encryption operations (AES and CP-ABE) are executed in the client side. During a decryption operation, the client sends to the server the security level and its set of attributes, previously retrieved in a secure manner from the AA entity. The server constructs the private key from the client's attributes and returns that key to the client. All the main decryption operations are performed in the client side. Exceptions could be thrown during a DET-ABE encryption or decryption process due to connectivity problems. On success, during an encryption operation a file with extension .detabe is created containing the serialized version of tuple T_E. In a decryption operation, the .detabe extension is removed from the input file, which contains the decrypted data of $CT_{\text{AES}} \in T_E$.

4.4 Keys Management

The PK and MK keys are generated in the server side according to the arithmetic operations shown in Table 1. All the resulting values that characterize these keys are stored in server (trusted authority) only. The random numbers α and β used for their creation are local variables that are destroyed after the keys are created. While PK can be read by the server and sent to clients performing an encryption operation, MK is used exclusively in the server side. The private keys for clients used in CP-ABE are created only in the server side and securely sent to clients.

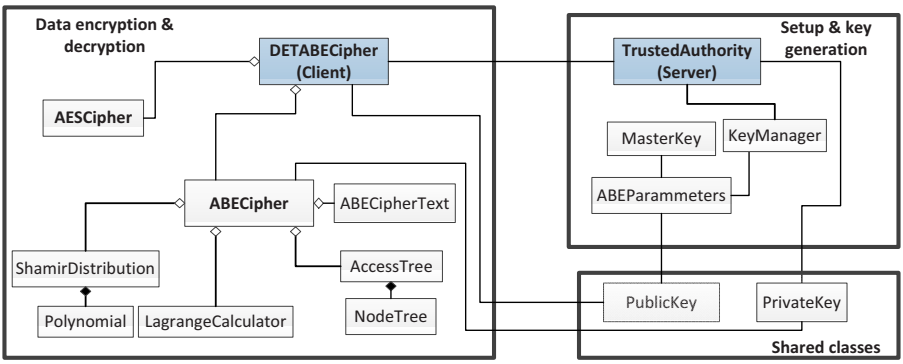

Fig. 3. Proposed set of Java classes for DET-ABE.

4.5 The Proposed Java Library for DET-ABE

The new set of classes comprising the new DET-ABE library are shown in Figure 3. The set of software modules in the client comprises classes for AES, CP-ABE encryption/decryption, and DET-ABE encryption/decryption. The set of software modules in the server side are classes for DET-ABE setup, CP-ABE setup, CP-ABE private key generation and DET-ABE key management. As explained in section 5, the AES class in Figure 3 is actually a wrapper for the AES implementation provided by Java SE. This wrapper adds the required methods to interface the symmetric cipher with CP-ABE for implementing the DET technique.

4.6 Using the Proposed API

In this section we show how the proposed API for DET-ABE can be used. We target the application of encryption and decryption of digital documents (.doc, .pdf). After encrypting these files they can be either locally saved in the client or stored at a cloud storage provider. The DET-ABE encryption of a digital document is shown in Listing 1.1.

```
1 import com.detabe.client.encryption.*;

3 public static void main(String args[]){
        String policy = "(directive AND level = 7) OR (
    accountant AND level >= 3)";
5
        DETABECipher cipher = new DETABECipher();
7       cipher.encrypt("contract.pdf", 128, policy);
}
```

Listing 1.1. Encrypting a digital document using DET-ABE

The `encrypt()` method of objetc `cipher` at line five in listing 1.1 implements all the logic specified in section 3.2. What is only specified by the programmer is the file to be encrypted (`contract.pdf`), the security level (128 bits) and the policy (`(directive AND level = 7) OR (accountant AND level >= 3)`, with four attributes). An exception can be thrown at line five in the following cases: *i*) the given security level is not supported, *ii*) the policy is a bad boolean equation, and *iii*) the attributes are not previously registered in the server. The result of `encrypt()` method is the encrypted file `contract.pdf.detabe`.

The DET-ABE decryption of the file `privateLetter.doc.detabe` is shown in Listing 1.2.

```
1  import com.detabe.client.cipher.*;

3  public static void main(String args[]){
       List<String> attributes = new LinkedList<String>();
5      attributes.add("directive");
       attributes.add("level = 5");
7      attributes.add("accountant");
       attributes.add("level = 4");
9
       DETABECipher cipher = new DETABECipher();
11     cipher.decrypt("privateLetter.doc.detabe",attributes)
   ;
}
```

Listing 1.2. Decrypting a ciphertext with DET-ABE

For decryption, the client only provides to the server its set of attributes. The server validates them and generates the corresponding private key for the client. The input encrypted file `privateLetter.doc.detabe` contains the ABE ciphertext, the the access structure, and pairing parameters. With all these elements together with the received private key from the server, the client recovers the original file `privateLetter.doc` performing all the steps described for DET-ABE decryption in section 3.3.

Before running the client programs the server must be launched, for example, by executing `java com.detabe.server.ABETrustedAuthority`. Also, in the client side the configuration file `configuration.cfgt` must have the correct IP or host name where the server (trusted authority) is running.

5 Implementation and Performance Results

In this section we present the running times of the main modules in the proposed API. The experimentation was carried out on a desktop machine 32-bit Intel Core2 Quad 2.40GHz, 4GB RAM with Windows7 as operative system. The main objective to present the execution time of DET-ABE scheme is to show the feasibility of using the proposed API in IT end-user applications. Further

optimizations could be done over specific modules in order to get the API running faster. The implementation of classes described in Figure 3 were built on top of jPBC, a Java API [9] that provides specialized modules for pairing computations and is defined over PBC[18].

All classes shown in Figure 3 except AES were implemented and validated. For AES, we use the implementation provided by Java in the packages `javax.crypto` and `javax.crypto.spec`. To support security levels above 128-bit, we use the Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files. All modules were integrated and the execution paths of DET-ABE main modules were validated performing unit and functional tests, ensuring that the result of each cryptographic function was correct.

In all tests, the client and server run on the same machine. Table 3 shows timing result of the DET-ABE setup module, that comprises the generation of pairing parameters, the master and the public keys. That module is executed once, and the previously generated keys are chosen and used by each client connection for DET-ABE encryption or decryption.

Table 3. Running times for DET-ABE setup.

	Timing (secs)		
Module	128-bit	192-bit	256-bit
Curve param. generation	30.78	8.47	24.03
PK and MK generation	2.26	2.32	4.45

Figure 4 shows the running time for DET-ABE encryption, DET-ABE decryption and CP-ABE key generation. For all experimentations, we used a PDF document of size 182Kbytes and considered the three standard security levels of 128-bit (symmetric pairing with type A elliptic curve), 192-bit, and 256-bit (asymmetric pairing with type F elliptic curve). The size of data being encrypted does not impact significantly the overall timing of DET-ABE as the number of attributes and security level do. Data size only affects the running time of the block cipher AES, which is proved to have a complexity $O(1)$ [8].

Figure 4 a) shows the running time for DET-ABE encryption. Note that more considerable time is spent in case of using a symmetric pairing (128-bit security). While a linear time with respect to the number of attributes is demanded for 128-bit encryption, that is not true for 192-bit and 256-bit. The same behaviour is exhibited in Figure 4 b), that shows the time required for CP-ABE private key generation. Figure 4 c) shows the decryption time. For 128-bit and 256-bit security levels the time is very similar whereas a security level of 192-bit achieves the better timing. Finally, Figure 4 d) contrasts the timing for each DET-ABE operation considering the three security levels. For this last experiment, a PDF file of 182Kbytes size and 6 attributes were used.

As it is observed, 128-bit encryption and private key generation using type A elliptic curves are by far the most time consuming operations. In case of 192-bit

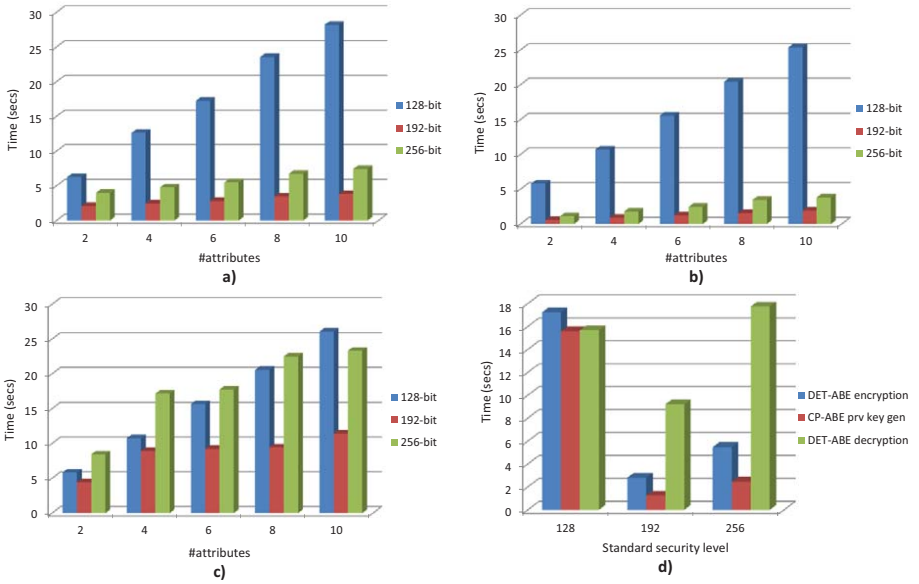

Fig. 4. Running times of DET-ABE main modules considering different number of attributes, a 182Kbyte digital document, and the three standard security levels. a) DET-ABE encryption. b) CP-ABE private key generation to open the digital envelope. c) DET-ABE decryption. d) Contrasting the three main DET-ABE modules using 6 attributes and the three standard security levels.

and 256-bit security levels, significant reduced time is obtained except for DET-ABE decryption, which remains with high timing costs. As it was previously stated, the proposed modules can be optimized to reduce the timing, for example, the optimized versions of jPBC can be used to speed up the low level operation (abstract algebra arithmetic).

6 Conclusion

We presented a set of Java classes aiming to reduce the existing gap for using Pairing Based Cryptography (PBC) and Attribute Based Encryption (ABE) in end-user IT applications. We presented the DET-ABE scheme for data encryption over a set of attributes, thus providing confidentiality and fine grained access control under an end-to-end encryption approach. DET-ABE is the result of using the Digital Envelope Technique (DET) together with CP-ABE and AES. The proposed DET-ABE software modules are built on top of libraries for low level computations in finite fields and groups. The complexity associated to the operations and settings for a secure implementation of cryptographic algorithms (adequate pairing parameters as the elliptic curve type and properties) are encapsulated in the proposed API, facilitating the use of those modules in end-user applications. The efficacy of proposed software modules was verified in a simple

application for securing digital documents. Further optimizations specially at low level modules can be done to outperform the achieved running times, which have shown to be viable in practical applications.

Future work is planned to explore implementation alternatives of the proposed API, for example with multi-threading programming using GPUs and multi-cores as underlying computing platforms to speed up the execution time of DET-ABE modules. As the running time also depended on the pairing parameters and elliptic curves used, further research will be conducted to explore other elliptic curve types to achieve faster computations, for example, to use alternative elliptic curves for the 128-bit security level.

References

1. Ambrosin, M., Conti, M., Dargahi, T.: On the feasibility of attribute-based encryption on smartphone devices. In: Proceedings of the 2015 Workshop on IoT Challenges in Mobile and Industrial Systems, IoT-Sys 2015, pp. 49–54. ACM, New York (2015). http://doi.acm.org/10.1145/2753476.2753482
2. Barker, E., Barker, W., Burr, W., Polk, W., Smid, M.: Recommendation for key management part 1: general(Revision 3). In: NIST Special Publication 800–57, pp. 1–147 (2012)
3. Barreto, P.S.L.M., Naehrig, M.: Pairing-friendly elliptic curves of prime order. In: Preneel, B., Tavares, S. (eds.) SAC 2005. LNCS, vol. 3897, pp. 319–331. Springer, Heidelberg (2006). http://dx.doi.org/10.1007/11693383_22
4. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: Proceedings of the 2007 IEEE Symposium on Security and Privacy, SP 2007, pp. 321–334. IEEE Computer Society, Washington, DC (2007). http://dx.doi.org/10.1109/SP.2007.11
5. Boneh, D.: Pairing-based cryptography: past, present, and future. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, p. 1. Springer, Heidelberg (2012)
6. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
7. Braun, J., Volk, F., Buchmann, J., Mühlhäuser, M.: Trust views for the web PKI. In: Katsikas, S., Agudo, I. (eds.) EuroMPI 2013. LNCS, vol. 8341, pp. 134–151. Springer, Heidelberg (2014)
8. Daemen, J., Rijmen, V.: The Design of Rijndael. Springer-Verlag New York Inc., Secaucus (2002)
9. De Caro, A., Iovino, V.: jPBC: Java pairing based cryptography. In: 2011 IEEE Symposium on Computers and Communications (ISCC), pp. 850–855, June 2011
10. Diffie, W., Van Oorschot, P.C., Wiener, M.J.: Authentication and authenticated key exchanges. Des. Codes Cryptography **2**(2), 107–125 (1992). http://dx.doi.org/10.1007/BF00124891
11. Escofier, J.P.: Galois Theory, Graduate Texts in Mathematics, vol. 204. Springer, New York (2001)
12. Galbraith, S.D., McKee, J.F.: Pairings on elliptic curves over finite commutative rings. In: Smart, N.P. (ed.) Cryptography and Coding 2005. LNCS, vol. 3796, pp. 392–409. Springer, Heidelberg (2005)

13. Galbraith, S.D., Paterson, K.G., Smart, N.P.: Pairings for cryptographers. Discrete Appl. Math. **156**(16), 3113–3121 (2008). http://dx.doi.org/10.1016/j.dam.2007.12.010
14. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, pp. 89–98. ACM, New York (2006). http://doi.acm.org/10.1145/1180405.1180418
15. Herstain, I.N.: Abstract Algebra, 3rd edn. Wiley (1996)
16. Jahid, S., Mittal, P., Borisov, N.: Easier: Encryption-based access control in social networks with efficient revocation. In: Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, ASIACCS 2011, pp. 411–415. ACM, New York (2011). http://doi.acm.org/10.1145/1966913.1966970
17. Liu, W., Liu, J., Wu, Q., Quin, B.: Android PBC: A pairing based cryptography toolkit for android platform. In: Communications Security Conference, CSC 2014, pp. 1–6. IEEE, May 2014
18. Lynn, B.: On the implementation of pairing-based cryptosystems. Ph.D. thesis, Stanford University, Department of Computere Science (2007)
19. Menezes, A.J., Vanstone, S.A., Oorschot, P.C.V.: Handbook of Applied Cryptography, 1st edn. CRC Press Inc., Boca Raton (1996)
20. Miller, F.P., Vandome, A.F., McBrewster, J.: Advanced Encryption Standard. Alpha Press (2009)
21. Pang, L., Yang, J., Jiang, Z.: A Survey of Research Progress and Development Tendency of Attribute-Based Encryption. The Scientific World Journal **2014**, 1–13 (2014)
22. Picazo-Sanchez, P., Tapiador, J.E., Peris-Lopez, P., Suarez-Tangil, G.: Secure publish-subscribe protocols for heterogeneous medical wireless body area networks. Sensors **14**(12), 22619 (2014). http://www.mdpi.com/1424-8220/14/12/22619
23. Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM **21**(2), 120–126 (1978). http://doi.acm.org/10.1145/359340.359342
24. Rosenberg, B.: Handbook of Financial Cryptography and Security, 1st edn. Chapman & Hall/CRC (2010)
25. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005). http://dx.doi.org/10.1007/11426639_27
26. Xiong, X., Wong, D.S., Deng, X.: TinyPairing: A fast and lightweight pairing-based cryptographic library for wireless sensor networks. In: 2010 IEEE Wireless Communication and Networking Conference, pp. 1–6. IEEE, April 2010