Privacy Enhanced DigiLocker using Ciphertext-Policy Attribute-Based Encryption

Puneet Bakshi and Sukumar Nandi
Department of Computer Science and Engineering, Indian Institute of Technology, Guwahati, Assam, India

Keywords: DigiLocker, Privacy, CP-ABE.

Abstract: Recently, Government of India has taken several initiatives to make India digitally strong such as to provide each resident a unique digital identity, referred to as *Aadhaar*, and to provide several online e-Governance services based on Aadhaar such as *DigiLocker*. DigiLocker is an online service which provides a shareable private storage space on public cloud to its subscribers. Although DigiLocker ensures traditional security such as data integrity and secure data access, privacy of e-documents are yet to addressed. Ciphertext-Policy Attribute-Based Encryption (CP-ABE) can improve data privacy but the right implementation of it has always been a challenge. This paper pressents a scheme to implement privacy enhanced DigiLocker using CP-ABE.

1 INTRODUCTION

In last decade, Government of India has taken several e-Governance initiatives such as a unique digital identity (referred to as *Aadhaar* (UIDAI, 2009)) for every resident, online Aadhaar based authentication and several online citizen centric services such as *eKYC*, *eSign*, and *DigiLocker*. At present, most of these services are built using traditional Public Key Infrastructure (PKI) with limited data privacy in which specifying authorized entities beforehand which are permitted to access data may not be possible and even if possible, the solution may not scale.

In *DigiLocker* (GoI, 2015), documents of subscribers are hosted on public cloud which is assumed to be a trusted entity. However, cloud storage may not be trustyworthy and may be susceptible to insider attacks. Moreover, instead of providing a *reactive access authorization to a single requester* (using OAuth2 (IETF, 2012)), a subscriber may want to provide a *proactive access authorization to multiple requester* meeting certain criteria of attributes.

Ciphertext-Policy Attribute-Based Encryption (CP-ABE) (Bethencourt et al., 2007) is a recent cryptogrpahic mechanism which can improve data privacy, but the right implementation and efficiency of it are still some of the major concerns for its wide deployment. This paper presents a scheme to implement privacy enhanced DigiLocker based on CP-ABE.

2 RELATED WORK

Recent developments in cryptography have introduced Attribute-Based Encryption (ABE) (Goyal et al., 2006) in which encryption is done under a set of attributes. ABE is classified in Key-Policy ABE (KP-ABE) (Goyal et al., 2006) and CP-ABE. In KP-ABE, access policy is encoded in subscriber's private key and a set of attributes are encoded in ciphertext. In CP-ABE, access policy is encoded in ciphertext and a set of attributes are encoded in subscriber's private key. In CP-ABE, only if the set of required attributes encoded in receiver's private key satisfies the access policy encoded in received ciphertext, will the receiver be able to decrypt the ciphertext. Since the introduction of CP-ABE, researchers have proposed innovative mechanisms to use it to improve data privacy (Zhou and Huang, 2012), (Ji et al., 2014).

3 DIGITAL LOCKERS IN INDIA

DigiLocker is an Aadhaar based online service which facilitates *subscribers* to store e-documents, *issuer* agencies to provide e-documents and *requester* applications to get access to e-documents. An *e-document* is a digitally signed electronic document. *Repositores* are provided by issuers to host collection of e-documents. *Digital Locker* is a storage space provided to each subscriber to store e-documents. *Requester* is an application which seeks access to some

Bakshi, P. and Nandi, S.
Privacy Enhanced DigiLocker using Ciphertext-Policy Attribute-Based Encryption.
DOI: 10.5220/0009777205410546
In *Proceedings of the 17th International Joint Conference on e-Business and Telecommunications (ICETE 2020) - SECRYPT*, pages 541-546
ISBN: 978-989-758-446-6

e-document. All participating entities must adhere to *Digital Locker Technology Specification (DLTS)* (MeitY, 2019).

An e-document is uniquely identified by a *Unique Resource Identifier (URI)* which is a triplet of the form ⟨IssuerID :: DocType :: DocID⟩, where *IssuerID* is a unique identifier ot the issuer, e.g., `CBSE`, for Central Board of Secondary Education. *DocType* is a classification of e-documents as defined by the issuer. For example, CBSE may classify certificates into `MSTN` for 10th mark sheet and `KVYP` for certificates issued to KVPY scholarship fellow. *DocType* also helps issuers to use different repositories for different types of e-documents. *DocID* is an issuer defined unique identifier (an alphanumeric string) of the e-document within a document type. Some hypothetical examples of e-document URI are ⟨`CBSE` :: `MSTN` :: `22636726`⟩, ⟨`DLSSB` :: `HSMS` :: `GJSGEJXS`⟩. DigiLocker ensures data integrity of e-documents by mandating that all e-documents are digitally signed by issuers.

When an issuer is registered, it provides two APIs, namely, `PullDoc` to pull an e-document based on a given URI and `PullUri` to pull all URIs meeting a given search criteria. When a requester application is registered, it is given a unique requester identifier, a secret key which is shared between DigiLocker and requester application and a `FetchDoc` API is given to access e-documents based on URI. Based on the URI, `FetchDoc` forwards the request to appropriate issuers to retrieve the e-document. DigiLocker ensures secure data access of e-documents by API license keys, secure transport, an explicit authentication (if required by *DocType*) and all requests and responses to be digitally signed.

4 PRELIMINARIES

This section briefly describes some of the necessary background.

4.1 Bilinear Pairings (Zhang et al., 2004)

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ are elliptic groups of order p, $\mathbb{G}_T$ is a multiplicative group of order p, g_1 is a generator of $\mathbb{G}_1$, g_2 is a generator of $\mathbb{G}_2$, $P \in \mathbb{G}_1$, $Q \in \mathbb{G}_2$ and a, b $\in \mathbb{Z}_p$, then a bilinear pairing is a map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ that satisfies the following three properties.

1 Bilinearity: $e(P^a, Q^b) = e(P, Q)^{ab}$
2 Non-Degeneracy: $e(g_1, g_2) \neq 1$
3 Computability: $e(P, Q)$ can be computed efficiently.

4.2 Decision Bilinear Diffie-Hellman (DBDH) Assumption (Yacobi, 2002)

Let $\mathbb{G}$, $\mathbb{G}_T$ are cyclic groups of prime order $p > 2^\lambda$ where $\lambda \in \mathbb{N}$, g is the generator of $\mathbb{G}$, $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is an efficiently computable symmetric bilinear pairing map and $a, b, c, z \in \mathbb{Z}_p$ are random numbers. The DBDH assumption states that no probabilistic polynomial time algorithm can distinguish between $\langle g, g^a, g^b, g^c, e(g,g)^{abc} \rangle$ and $\langle g, g^a, g^b, g^c, e(g,g)^z \rangle$ with more than a negligible advantage.

4.3 Security Model

The security model of proposed scheme is based on the following IND-sAtt-CPA game (Ibraimi et al., 2009) between a challenger and an adversary $\mathcal{A}$.

Init Phase. Adversary $\mathcal{A}$ chooses a challenge access tree $\mathcal{T}^*$ and gives it to challenger.

Setup Phase. Challenger runs a *setup* procedure to generate ⟨ASK, APK⟩ and gives the public key APK to adversary $\mathcal{A}$.

Phase I. Adversary $\mathcal{A}$ makes an attribute- based private key request to the key generation oracle for any attribute set with the restriction that the attribute set should not include any attribute which is part of $\mathcal{T}^*$. Challenger generates the key as described in section 5.4 and returns the same to adversary $\mathcal{A}$.

Challenge Phase. Adversary $\mathcal{A}$ sends two equal length messages m_0 and m_1 to challenger. Challenger chooses a random number $b \in_R \{0,1\}$, encrypts m_b using $\mathcal{T}^*$ and APK as is described in section 5.3.

Phase II. Adversary $\mathcal{A}$ can send multiple requests to generate attribute-based private key with the same restriction as in Phase I.

Guess Phase. Adversary $\mathcal{A}$ outputs a guess $b\prime \in \{0,1\}$.

The advantage of adversary $\mathcal{A}$ in this game is defined to be $\varepsilon = |Pr[b\prime = b] - \frac{1}{2}|$. Only if any polynomial time adversary $\mathcal{A}$ has a negligible advantage, the scheme is considered secure against an adaptive chosen plaintext attack (CPA).

4.4 Access Tree

Access tree structure is a means to specify an access policy during encryption that must be satisfied by attribute-based private keys in order to decrypt. Let $\mathcal{T}$ be a tree representing an access structure. Each non-leaf node of the tree represents a threshold gate, described by its children and a threshold value. If num_x

is the number of children and k_x is the threshold value of a node x, then, $k_x = 1$ represents an OR gate and $k_x = num_x$ represents an AND gate. Each leaf node x of the tree is described by an attribute and a threshold value $k_x = 1$.

Let $\mathcal{T}_x$ denotes the subtree rooted at node x. If a set of attributes λ satisfies the subtree $\mathcal{T}_x$, it is represented as $\mathcal{T}_x(\lambda) = 1$. $\mathcal{T}_x(\lambda)$ is computed recursively as follows. If x is a non-leaf node, evaulate $\mathcal{T}(y)$ for all children nodes y of node x. $\mathcal{T}_x(\lambda)$ returns 1 if and only if at least k_x children return 1. If x is a leaf node, then $\mathcal{T}_x(\lambda)$ returns 1 if and only if $\texttt{attr}(x) \in \lambda$.

5 OUR CONSTRUCTION

The proposed scheme introduces two new roles, namely, *Attribute Authority Manager (AAM)* and *Attribute Authority (AA)*. AAM is an entity which manages the universe of attributes and AA is an entity which manages a set of attributes (as assigned by AAM). DigiLocker is proposed to assume the role of AAM and individual issuers are proposed to assume the role of AA. A subscriber is assigned a set of attributes from each issuer which holds at least one e-document of the subscriber. Each requester application is assigned a set of attributes from DigiLocker based on certain criteria such as purpose of access, for how long the data is going to be used, etc. To create a privacy enhanced e-document for a subscriber, issuer and susbcriber mutually creates an attribute-based token (which will be used later in encryption) for an access policy, generates a symmetric key, encrypts the document with symmetric key, encrypts the symmetric key with attribute-based token, creates an e-document enclosing both the encrypted symmetric key and the encrypted document, creates a URI for this e-document and pushes it to subscriber's digital locker using `PushURI` API. When this e-document is shared with a requester application, the requester will be able to decrypt the encrypted symmetric key only if the requester is associated with a set of attributes which satisfies the access policy used to encryt the symmetric key. Only when the requester obtais the symmetric key, will he be able to decrypt and retrieve the document.

In *Setup(κ)* procedure, AAM chose a cyclic group $\mathbb{G}_0$ of large prime order p (κ defines the size of group) on which discrete logarithm problem is assumed to be hard, generator g, a bilinear map $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$ for which bilinear diffie hellman problem is assumed to be hard, a hash function $H : \{0,1\}^* \to \mathbb{G}_0$ which maps a binary string encoded attribute to a group element, chose random numbers $\alpha, \beta \in_R \mathbb{Z}_p$ and set its private key ASK and public key APK as below.

$$\begin{aligned} \text{ASK} &= \{\beta, g^\alpha\} \\ \text{APK} &= \{g^\beta, e(g,g)^\alpha, \mathbb{G}_0, g\} \end{aligned}$$

5.1 Attribute Assignment

An attribute can be any characteristic of a subscriber or requester and is represented by a binary string $\{0,1\}^*$. Attribute assignement to both subscribers and requesters is proposed to be done lazily in the background with the aim to keep the list of associated attributes in DigiLocker up to date.

For subscriber's attribute assignment and modification, two APIs are proposed to be introduced. First is $\texttt{PullAttrs}(\texttt{ID}_i)$ which is provided by issuers and is consumed by DigiLocker to pull updated list of attributes of subscriber with Aadhaar number ID_i. Second is $\texttt{PushAttrs}(\texttt{ID}_i, \texttt{NewAttrs})$ which is provided by DigiLocker and is consumed by issuer to push any change in attributes of subscriber with Aadhaar number ID_i. For requester applications, attributes are assigned and updated by DigiLocker.

It is important to take appropriate measures to handle load of a voluminous country like India. One such measure could be to prepone part of the encryption process. This preponed encryption process generates a token with mutual cooperation between subscriber and issuer. This token can be reused every time for a given subscriber and for a given access policy.

A helper procedure $\texttt{encPartial}(\mathcal{T}, r)$ is assumed to be present which works as follows. It choses a polynomial q_x for each node x (including the leaves) in the tree $\mathcal{T}$. These polynomials are chosen in the following way in a top-down manner, starting from the root node R. For each node x in the tree, set the degree d_x of the polynomial q_x to be one less than the threshold value k_x of that node, that is, $d_x = k_x - 1$. Starting with the root node R the procedure chooses a random $r \in_R \mathbb{Z}_p$ and sets $q_r(0) = r$. Then, it chooses d_R other points of the polynomial q_R randomly to define it completely. For any other node x, it sets $q_x(0) = q_{\text{parent}(x)}(\text{index}(x))$ and chooses d_x other points randomly to completely define q_x.

5.2 Token Generation

An access tree $\mathcal{T}_{iv}$ is comprised of access subtree $\mathcal{T}_{S_{iv}}$ from subscriber S_i and access subtree $\mathcal{T}_{I_{iv}}$ from issuer I_v (refer figure 1). If issuer I_v needs to generate its part of token for subscriber S_i, for access tree $\mathcal{T}_{iv}$, it generates a random number $r_i \in_R \mathbb{Z}_p$, and generates following partial-token using APK and

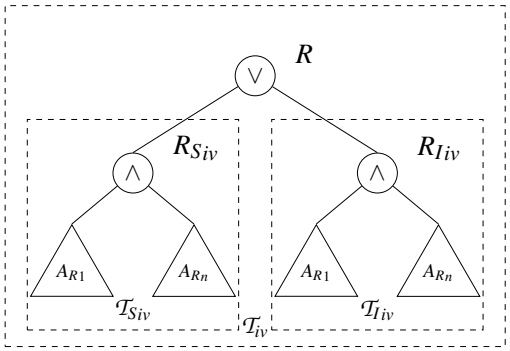

Figure 1: Example of an access policy tree.

$\texttt{encPartial}(\mathcal{T}_{\texttt{I}_{\texttt{iv}}}, \texttt{r}_\texttt{i})$. Let Y_I is the set of leaf nodes in $\mathcal{T}_{I_{iv}}$.

$$\text{CTtok}_{\text{I}_{\text{iv}}} = \left\{ \begin{array}{l} \mathcal{T}_{I_{iv}} \\ \text{C1}_\text{I} = \text{e}(\text{g},\text{g})^{\alpha \text{r}_\text{i}} \\ \text{C2}_\text{I} = \text{g}^{\beta \text{r}_\text{i}} \\ \left.\begin{array}{l} \text{C3}_{\text{I}_\text{y}} = \text{g}^{\text{q}_\text{y}(0)} \\ \text{C4}_{\text{I}_\text{y}} = \text{H}(\texttt{attr}(\text{y}))^{\text{q}_\text{y}(0)} \end{array}\right\}^{\forall\ \text{y} \in \text{Y}_\text{I}} \end{array} \right.$$

Issuer notifies subscriber to provide its part of the token. Subscriber S_i generates a random number $r_s \in_R \mathbb{Z}_p$ and generates following partial-token using APK and $\texttt{encPartial}(\mathcal{T}_{\texttt{S}_{\texttt{iv}}})$. Let Y_S is the set of leaf nodes in $\mathcal{T}_{S_{iv}}$.

$$\text{CTtok}_{\text{S}_{\text{iv}}} = \left\{ \begin{array}{l} \mathcal{T}_{S_{iv}} \\ \text{C1}_\text{S} = \text{e}(\text{g},\text{g})^{\alpha \text{r}_\text{s}} \\ \text{C2}_\text{S} = \text{g}^{\beta \text{r}_\text{s}} \\ \left.\begin{array}{l} \text{C3}_{\text{S}_\text{y}} = \text{g}^{\text{q}_\text{y}(0)} \\ \text{C4}_{\text{S}_\text{y}} = \text{H}(\texttt{attr}(\text{y}))^{\text{q}_\text{y}(0)} \end{array}\right\}^{\forall\ \text{y} \in \text{Y}_\text{S}} \end{array} \right.$$

Subscriber provides its part of partial-token to issuer. Issuer creates the final token by combining the two partial-tokens and keeps it securely with it.

$$\text{CTtok}_{\text{iv}} = \left\{ \begin{array}{l} \mathcal{T}_{iv} = \mathcal{T}_{S_{iv}} \cup \mathcal{T}_{I_{iv}} \\ \text{C1} = \text{C1}_\text{S}.\text{C1}_\text{I} \\ \quad = \text{e}(\text{g},\text{g})^{\alpha \text{r}_\text{s}} \text{e}(\text{g},\text{g})^{\alpha \text{r}_\text{i}} \\ \text{C2} = \text{C2}_\text{S}.\text{C2}_\text{I} = \text{g}^{\beta \text{r}_\text{s}} \text{g}^{\beta \text{r}_\text{i}} \\ \left.\begin{array}{l} \text{C3} = \text{C3}_{\text{S}_\text{y}} \cup \text{C3}_{\text{I}_\text{y}} \\ \quad = \text{g}^{\text{q}_\text{y}(0)} \\ \text{C4} = \text{C4}_{\text{S}_\text{y}} \cup \text{C4}_{\text{I}_\text{y}} \\ \quad = \text{H}(\texttt{attr}(\text{y}))^{\text{q}_\text{y}(0)} \end{array}\right\}^{\forall\ \text{y} \in Y_S \cup Y_I} \end{array} \right.$$

5.3 Encryption

A new *DocType* `PRIV` is proposed to be introduced for privacy enhanced e-documents. To create a privacy enahanced e-document, issuer creates a URI $\langle \text{I}_\text{v} :: \text{PRIV} :: \text{D}_\text{w} \rangle$ where I_v is the issuer identifier and D_w is the document identifier within the document type `PRIV`. Now, issuer generates a random number $r_{ie} \in_R \mathbb{Z}_p$, generates a symmetric key SK_{ivw}, encrypts e-document m with SK_{ivw}, encrypts SK_{ivw} with $CTtok_{I_{iv}}$ and produces the following ciphertext.

$$\text{CT}_{\text{ivw}} = \left\{ \begin{array}{l} \mathcal{T}_{iv} = \mathcal{T}_{S_{iv}} \cup \mathcal{T}_{I_{iv}} \\ \text{C1} = \text{e}(\text{g},\text{g})^{\alpha \text{r}_\text{s} \text{r}_\text{ie}} \text{e}(\text{g},\text{g})^{\alpha \text{r}_\text{i} \text{r}_\text{ie}} \text{SK}_{\text{ivw}} \\ \text{C2} = \text{g}^{\beta \text{r}_\text{s} \text{r}_\text{ie}} \text{g}^{\beta \text{r}_\text{i} \text{r}_\text{ie}} \\ \left.\begin{array}{l} \text{C3}_\text{y} = \text{g}^{\text{r}_\text{ie}(\text{q}_\text{y}(0))} \\ \text{C4}_\text{y} = \text{H}(\texttt{attr}(\text{y}))^{\text{q}_\text{y}(0)} \end{array}\right\}^{\forall\ \text{y} \in \text{Y}_\text{iv}} \\ \text{C5} = \{\text{m}\}_{\text{SK}_{\text{ivw}}} \end{array} \right.$$

5.4 Key Generation

A new API $\texttt{GenABPvtKey}(\texttt{ID}_\texttt{i}, \texttt{IS}_\texttt{j})$ is proposed to be provided by DigiLocker to generate an attribute-based private key for a subscriber with Aadhaar identifier ID_i and with attributes from issuers in the set IS_j. Let S_{ij} is the set of all attributes assigned to S_i by all issuers in set IS_j. DigiLocker generates random numbers $r \in_R \mathbb{Z}_p$, $r_j \in_R \mathbb{Z}_p$ for each attribute $j \in_R S_{ij}$, computes attribute-based private key $ASK_{ID_iIS_j}$ as below and keeps this key securely with it.

$$\text{ASK}_{\text{ID}_\text{i}\text{IS}_\text{j}} = \left\{ \begin{array}{l} \text{D} = \text{g}^{(\alpha+\text{r})/\beta} \\ \left.\begin{array}{l} \text{D}_\text{j} = \text{g}^{\text{r}}.\text{H}(\text{j})^{\text{r}_\text{j}} \\ \text{D}_\text{j}\prime = \text{g}^{\text{r}_\text{j}} \end{array}\right\}^{\forall\ \text{j} \in \text{S}_{\text{ij}}} \end{array} \right.$$

Note that multiple attribute based private keys can exist for a subscriber for different set of attributes. If any one issuer set IS_i is a proper subset of an other issuer set IS_j, the key corresponding to IS_i is redundant and can be removed.

5.5 Decryption

A new API `FetchPrivDocURI` is proposed to be provided by DigiLocker for decryption purpose. This API facilitates a requester with identifier ID_R to retrieve ciphertext CT_{ivw} of e-document from URI $\langle \text{I}_\text{v} :: \text{PRIV} :: \text{D}_\text{W} \rangle$ of subscriber S_i. DigiLocker extracts the set of attribute issuers IS_k from $CT_{ivw} \rightarrow \mathcal{T}_{iv}$, retrieves $ASK_{ID_RIS_k}$ and calls $\texttt{Decrypt}(\texttt{CT}_{\texttt{ivw}}, \texttt{ASK}_{\texttt{ID}_\texttt{R}\texttt{IS}_\texttt{k}})$. A helper procedure $\texttt{DecryptNode}(\texttt{CT}_{\texttt{ivw}}, \texttt{ASK}_{\texttt{ID}_\texttt{R}\texttt{IS}_\texttt{K}})$ is defined as below. Let S_k is the set of all attributes from issuers in set IS_k. If x is a leaf node and if $\texttt{attr}(\text{x}) \notin \text{S}_\text{k}$, then $\texttt{DecryptNode}(\texttt{CT}_{\texttt{ivw}}, \texttt{ASK}_{\texttt{ID}_\texttt{R}\texttt{IS}_\texttt{k}}, \text{x}) = \perp$ else if $\texttt{attr}(\text{x}) \in \text{S}_\text{k}$, then the procedure is defined as below.

$$\begin{aligned} &\texttt{DecryptNode}(\texttt{CT}_{\texttt{ivw}}, \texttt{ASK}_{\texttt{ID}_\texttt{R}\texttt{IS}_\texttt{k}}, \text{x}) \\ &= \frac{\text{e}(\text{D}_\text{x}, \text{C4}_\text{y})}{\text{e}(\text{D}_\text{x}\prime, \text{C5}_\text{y})} \\ &= \frac{\text{e}(\text{g}^{\text{r}}.\text{H}(\texttt{attr}(\text{x}))^{\text{r}_\text{j}}, \text{g}^{\text{r}_\text{ie}\text{q}_\text{y}(0)})}{\text{e}(\text{g}^{\text{r}_\text{j}}, \text{H}(\texttt{attr}(\text{x}))^{\text{q}_\text{x}(0)})} \\ &= \text{e}(\text{g},\text{g})^{\text{rr}_\text{ie}\text{q}_\text{x}(0)} \end{aligned}$$

If x is a non-leaf node, the recursive procedure is

defined as follows. For all children nodes z of x, $\texttt{DecryptNode}(\texttt{CT}_{\texttt{ivw}}, \texttt{ASK}_{\texttt{i}}, \texttt{x})$ is called and their output is stored in F_z. Let S_x be an arbitrary k_x sized set of child nodes z such that $F_z \neq \perp$. If no such set exists then the node was not satisfied and the function returns $\perp$. Otherwise, F_x is computed as below.

$$\begin{aligned} F_x &= \prod_{z \in S_x} F_z^{\Delta_{i,S_x\prime}(0)} \quad \text{where}\{i = \text{index}(z) \\ &\qquad S_x\prime = \{\text{index}(z) : z \in S_x\} \\ &= \prod_{z \in S_x} F_z^{\Delta_{i,S_x\prime}(0)} \\ &= \prod_{z \in S_x} (e(g,g)^{r.r_{ie}.q_z(0)})^{\Delta_{i,S_x\prime}(0)} \\ &= \prod_{z \in S_x} (e(g,g)^{r.r_{ie}.q_{\text{parent}(z)}(\text{index}(z))})^{\Delta_{i,S_x\prime}(0)} \\ &= \prod_{z \in S_x} e(g,g)^{r.r_{ie}.q_x(i)\Delta_{i,S_x\prime}(0)} \\ &= e(g,g)^{rr_{ie}q_x(0)} (\text{using polynomial interpolation}) \end{aligned}$$

$\texttt{Decrypt}(\texttt{CT}_{\texttt{ivw}}, \texttt{ASK}_{\texttt{ID}_\texttt{R}\texttt{IS}_\texttt{k}})$ calls $\texttt{DecryptNode}(\texttt{CT}_{\texttt{ivw}}, \texttt{ASK}_{\texttt{ID}_\texttt{R}\texttt{IS}_\texttt{k}}, \texttt{R})$ where R is root node of T_{iv}. If the access tree is satisfied by attributes in S_k, set $A = \texttt{DecryptNode}(\texttt{CT}_{\texttt{ivw}}, \texttt{ASK}_{\texttt{ID}_\texttt{R}\texttt{IS}_\texttt{k}}, \texttt{R}) = e(g,g)^{r.r_{ie}.(r_s+r_i)}$. Now the procedure obtains symmetric key SK_{ivw} by computing

$$\begin{aligned} \frac{C1}{\frac{e(C2,D)}{A}} &= \frac{e(g,g)^{\alpha r_s r_{ie}} e(g,g)^{\alpha r_i r_{ie}} SK_{ivw}}{\frac{e(g^{\beta r_s r_{ie}} g^{\beta r_i r_{ie}}, g^{(\alpha+r)/\beta})}{e(g,g)^{rr_{ie}(r_s+r_i))}}} \\ &= \frac{e(g,g)^{\alpha r_s r_{ie}} e(g,g)^{\alpha r_i r_{ie}} SK_{ivw}}{\frac{e(g^{\beta r_{ie}(r_s+r_i)}, g^{(\alpha+r)/\beta})}{e(g,g)^{rr_{ie}(r_s+r_i))}}} \\ &= \frac{e(g,g)^{\alpha r_{ie}(r_s+r_i)} SK_{ivw}}{\frac{e(g,g)^{r_{ie}(r_s+r_i)(\alpha+r)}}{e(g,g)^{rr_{ie}(r_s+r_i))}}} \\ &= SK_{ivw} \end{aligned}$$

Symmetric key SK_{ivw} is now used to decrypt the encrypted e-document.

$$m = \{CT_{ivw} \rightarrow C5\}_{SK_{ivw}}$$

DigiLocker returns the decrypted document m to requester.

6 SECURITY ANALYSIS

If the proposed scheme is not secure than an adversary $\mathcal{A}$ can win IND-sAtt-CPA game and solve the DBDH assumption with advantage $\varepsilon/2$. If the DBDG assumption is solved by adversary $\mathcal{A}$, a simulator β can be built which can solve DBDH assumption with advantage $\varepsilon/2$. Challenger chose a group $\mathbb{G}_0$, a generator g, a bilinear map e and chose random numbers $a, b, c, \theta \in_R \mathbb{Z}_p^*$. The challenger selects at random $\mu \in_R 0, 1$ and sets Z_μ as below.

$$Z_\mu = \begin{cases} (g,g)^{abc}, & \text{if } \mu = 0 \\ e(g,g)^{\theta}, & \text{if } \mu = 1 \end{cases}$$

Challenger provides DBDB challenge to the simulator: $\langle g, A, B, C, Z_\mu \rangle$ $\langle g, g^a, g^b, g^c, Z_\mu \rangle$. In IND-sAtt-CPA game, simulator β plays the role of challenger for adversary $\mathcal{A}$.

Init Phase. The adversary chose the challenge access tree $\mathcal{T}^*$ and gives it to simulator.

Setup Phase. The challenger chose a random number $x\prime \in \mathbb{Z}_p$, sets $\alpha = ab + x\prime$ and computes y as below.

$$y = e(g,g)^{\alpha} = e(g,g)^{ab} e(g,g)^{x\prime}$$

Now, challenger chose a random numbers $r \in_R \mathbb{Z}_p$ and $r_i \in_R \mathbb{Z}_p$ for $(1 \leq i \leq |\mathbb{U}|)$ and for all $a_j \in \mathbb{U}$, computes d_j and $d_j\prime$ as below.

$$\left.\begin{aligned} d_j &= \begin{cases} g^{r/b} H(j)^{r_j} & \text{...if } a_j \notin \mathcal{T}^* \\ g^{r} H(j)^{r_j} & \text{...if } a_j \in \mathcal{T}^* \end{cases} \\ d_j\prime &= g^{r_j} \end{aligned}\right\} (1 \leq j \leq |U|)$$

Now, challenger sends public parameters $APK = \{g^\beta, e(g,g)^\alpha, \mathbb{G}, g\}$ to adversary $\mathcal{A}$.

Phase 1. In this phase, adversary $\mathcal{A}$ sends requests for private key for any set of attributes w_j which does not contain any attribute in $\mathcal{T}^*$.

$$w_j = \{a_j \mid (a_j \in \mathbb{U} \wedge a_j \notin \mathcal{T}^*)\}$$

For each query from adversary $\mathcal{A}$, challenger chose a random number $r\prime \in_R \mathbb{Z}_p$, sets $r = -b(r\prime + a)$ and computes D as below.

$$\begin{aligned} D &= g^{(\alpha+r)/\beta} = (g^{(\alpha+r)})^{1/\beta} = (g^{((ab+x\prime)-b(r\prime+a))})^{1/\beta} \\ &= (g^{x\prime - br\prime})^{1/\beta} = (g^{x\prime}.(g^b)^{-r\prime})^{1/\beta} \end{aligned}$$

Because of restriction $a_j \notin \mathcal{T}^*$ in this phase, D_j can be computed as below.

$$\begin{aligned} D_j &= g^{r/b} H(j)^{r_j} = g^{r/b} H(j)^{r_j} = g^{-(r_\prime + a)} H(j)^{r_j} \\ &= (g^a)^{-1} g^{-r\prime} H(j)^{r_j} \end{aligned}$$

Now, challenger sends private key $ASK_{w_j} = D, (D_j, D_j\prime) \mid \forall a_j \in w_j$ to adversary $\mathcal{A}$

Challenge Phase. In this phase, adversary $\mathcal{A}$ submits two plaintext messages m_0 and m_1 to the challenger. Challenger selects a random plaintext message m_b from the two messages where $b \in_R \{0,1\}$, sets $r_{ie} = 1$, chose random variables r_i and r_s such that $c = r_i + r_c$. Now, set value of root node $\mathcal{T}^*$ to c and assign values to leaf nodes of $\mathcal{T}^*$ as described in section 4.4 to arrive at $C3_y$ and $C4_y$. The final ciphertext $CT_{\mathcal{T}^*}$ is computed

as below. The ciphertext is returned to adversary $\mathcal{A}$.

$$CT_{\mathcal{T}^*} = \begin{cases} \mathcal{T}_{iv} = \mathcal{T}^* \\ \mathrm{C1} = \mathrm{e(g,g)}^{\alpha r_s} \mathrm{e(g,g)}^{\alpha r_i} \mathrm{m_b} \\ \quad = \mathrm{e(g,g)}^{\alpha(r_s+r_i)} \mathrm{m_b} \\ \quad = \mathrm{e(g,g)}^{c} \mathrm{m_b} \\ \mathrm{C2} = \mathrm{g}^{\beta r_s} \mathrm{g}^{\beta r_i} = \mathrm{g}^{\beta(r_s+r_i)} \\ \quad = \mathrm{g}^{\beta} \mathrm{g}^{c} \\ \left.\begin{array}{l} \mathrm{C3_y} = \mathrm{g}^{(q_y(0))} \\ \mathrm{C4_y} = \mathrm{H}(\mathtt{attr}(y))^{q_y(0)} \end{array}\right\} \forall\, y \in Y_{iv} \end{cases}$$

Phase 2. In this phase, adversary $\mathcal{A}$ can continue to send secret key generation requests with the same restriction as in *Phase*1, i.e., $a_j \notin \mathcal{T}^*$.

Guess Phase. In this phase, adversary $\mathcal{A}$ outputs a guess $b\prime \in \{0,1\}$. If $b\prime = b$, the simulator β will guess that $\mu = 0$ and $Z_\mu = e(g,g)^{abc}$, otherwise will guess that $\mu = 1$ and $Z_\mu = e(g,g)^{\theta}$. When $Z_u = e(g,g)^{abc}$ the simulator β gives the perfect simulation and $c_{\mathcal{T}^*}$ is a valid ciphertext. Therefore the advantage of the adversary is

$$\Pr[b\prime = b \mid Z_\mu = e(g,g)^{abc}] = \frac{1}{2} + \varepsilon$$

If $\mu = 1$ then $Z_\mu = e(g,g)^{\theta}$ and $c_{\mathcal{T}^*}$ is random ciphertext for the adversary, and the adversary does not gain information about m_b. Hence we have

$$\Pr[b\prime \neq b \mid Z_\mu = e(g,g)^{\theta}] = \frac{1}{2}$$

Since the simulator β guesses $\mu\prime = 0$ when $b\prime = b$ and $\mu\prime = 1$ when $b\prime \neq b$, the overall advantage of β to solve DBDH assumption is

$$\frac{1}{2}\Pr[\mu\prime = \mu \mid \mu = 0] + \frac{1}{2}\Pr[\mu\prime = \mu \mid \mu = 1] - \frac{1}{2} = \frac{\varepsilon}{2}$$

If the adversary $\mathcal{A}$ has the above advantage ε to win the IND-sAtt-CPA game, the challenger can solve the DBDH assumption problem with $\varepsilon/2$ advantage with the help of adversary $\mathcal{A}$. However, there are no effective polynomial algorithms which can solve the DBDH assumption problem with non-negligible advantage according to the DBDH assumption. Hence, the adversary cannot win the IND-sAtt-CPA game with the above advantage ε, namely the adversary having no advantage to break through the proposed scheme.

7 CONCLUSION

This paper presented a scheme to improve data privacy in DigiLocker by using CP-ABE. The scheme also proposed to prepone part of the encryption process to increase performance. This preponed process creates a token which can be reused later. The proposed scheme is proved to be secure against IND-sAtt-CPA game. The proposed scheme can further be enhanced by using homomorphic encryption which allows processing on encrypted data and using post-quantum ABE schemes, for both of which, though schemes exist but are still non-trivial and not practical.

REFERENCES

Bethencourt, J., Sahai, A., and Waters, B. (2007). Ciphertext-policy attribute-based encryption. In *2007 IEEE symposium on security and privacy (SP'07)*, pages 321–334. IEEE.

GoI (2015). Digilocker. https://digilocker.gov.in.

Goyal, V., Pandey, O., Sahai, A., and Waters, B. (2006). Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 89–98.

Ibraimi, L., Tang, Q., Hartel, P., and Jonker, W. (2009). Efficient and provable secure ciphertext-policy attribute-based encryption schemes. In *International Conference on Information Security Practice and Experience*, pages 1–12. Springer.

IETF (2012). The oauth 2.0 authorization framework. https://tools.ietf.org/rfc/rfc6749.txt.

Ji, Y.-m., Tan, J., Liu, H., Sun, Y.-p., Kang, J.-b., Kuang, Z., and Zhao, C. (2014). A privacy protection method based on cp-abe and kp-abe for cloud computing. *JSW*, 9(6):1367–1375.

MeitY (2019). Digital locker technical specification (dlts). https://img1.digitallocker.gov.in/assets/img/technicalspecifications-dlts-ver-2.3.pdf.

UIDAI (2009). What is aadhaar. https://uidai.gov.in/myaadhaar/about-your-aadhaar.html.

Yacobi, Y. (2002). A note on the bilinear diffie-hellman assumption. *IACR Cryptology ePrint Archive*, 2002:113.

Zhang, F., Safavi-Naini, R., and Susilo, W. (2004). An efficient signature scheme from bilinear pairings and its applications. In *International Workshop on Public Key Cryptography*, pages 277–290. Springer.

Zhou, Z. and Huang, D. (2012). Efficient and secure data storage operations for mobile cloud computing. In *2012 8th international conference on network and service management (cnsm) and 2012 workshop on systems virtualiztion management (svm)*, pages 37–45. IEEE.