Privacy Enhanced Attribute based eSign

Puneet Bakshi and Sukumar Nandi

Indian Institute of Technology, Guwahati, Assam, India

Abstract. In recent years, Government of India has introduced many *Aadhaar* based online services. Although these initiatives helped India compete in digital revolution across world and were acclaimed by many, they have also raised some concerns about security especially the privacy aspects. One of the initiative in this direction is *eSign* which provides an online electronic signature service to its subscribers. Although most of the security aspects are addressed by eSign, some of the privacy aspects are yet to be addressed. This paper presents a scheme to implement privacy enhanced eSign using Attribute based Signatures (ABS). For the practical and efficient realization of the scheme, a token based approach is proposed.

Keywords: eSign · Aadhaar · eSign · Privacy.

1 Introduction

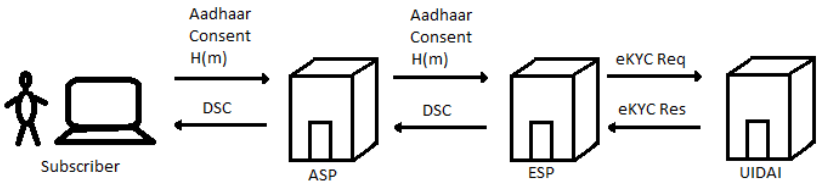
In recent years, Government of India has taken several initiatives [1] to make India digitally strong. One of the earliest initiative in this direction was to provide each resident a unique 12 digit identification number called *Aadhaar* [2]. In this initiative, a resident registers himself with a central nationwide agency, *Unique Identification Authority of India (UIDAI)* [3] and provides information such as proof of address, proof of identity, biometric data (fingerprints/iris-scan), mobile number, email address, etc. After verification of details, resident is given a unique 12 digit identification number called Aadhaar. UIDAI hosts several online services such as eKYC, DigiLocker and eSign based on Aadhaar based online authentication. With the widespread success of Aadhaar, many service providers such as financial and educational institutions, public and private offices, telecommunication service providers have started integrating with Aadhaar.

eSign [4] is an online electronic signature service in India which has received legal sanctity after the passage of Information Technology Act (ITA), 2000. Any *eSign* document is now considered at par with the handwritten signature of the same document. eSign is based on national Public Key Infrastructure (PKI) framework of the country [5] and is regulated by *Controller of Certifying Agency (CCA)* [6] which is the *Root Certifying Authority (RCA)* of India. CCA can authorize other agencies as next level *Certifying Authority (CA)* who can provide DSCs to their subscribers. CCA also authorizes other agencies as *eSign Service Providers (ESP)*. Institutions and organizations which need eSign service can integrate with ESP to provide electronic signature service to their subscribers. These other institutions and organizations are referred to as *Application Service Providers (ASP)*.

Dhinaharan Nagamalai et al. (Eds): NATP, ACSTY, ADCOM, SOFE, CCCIOT, MLSC, SVC - 2020
pp. 65-77, 2020. © CS & IT-CSCP 2020 DOI: 10.5121/csit.2020.100406

The present model of eSign, version v2.1, works as follows (refer figure 1). To eSign a document m, subscriber provides his Aadhaar number, consent and H(m) of the document to ESP through ASP. H() is the one way secure hash function. ESP authenticates subscriber based on his Aadhaar number and call eKYC API of UIDAI to retrieve information about the subscriber. ESP creates a private public key pair for subscriber, signs H(m) using private key, creates Digital Signature Certificate (DSC) including public key and returns the same to subscriber. eSign also provides *bulk signature* feature which allows few tens of documents to be signed in single eSign request. Because

Fig. 1: Present model of eSign



of inherent limitations of PKI, eSign has certain limitations such as it attests an identity and not the possession of attributes, to a claim. Another limitation is that it does not permit the signer to remain indistinguishable among the set of people in possession of same attributes. The second limitation prevents maintaining privacy of the subscriber. One more limitation is the assurance level of subscriber's consent which in most of the cases is taken in an HTML form.

Recent developments in cryptography have introduced Attribute Based Signature (ABS) [7], in which the signature attests the possession of attributes (and not identity) to a claim. Although ABS can address the limitations cited above, it is not deployed widely and the right and efficient implementation is still a major concern. Some other concerns are performance of bulk signatures, assurance level of subscriber's consent and the overall workflow describing roles of each participant.

This paper presents a mechanism for participating entities to collaborate in generating an attribute based token which can be reused in eSign requests to prevent initial time spent in authenticating the subscriber and generating the access tree. Other than that, this paper presents the overall scheme to implement privacy enhanced eSign using attribute based token.

Rest of the paper is organized in following sections. Section 2 presents the related work. Section 2 presents some of the preliminaries which are required to understand the scheme. Section 4 presents the proposed scheme. Section 5 presents the security analysis of the proposed scheme. Section 6 presents the performance analysis of the proposed scheme. Section 7 presents the conclusion of the present work along with the possible future scope of improvement.

2 Related Work

A major percentage of secure systems (such as eSign) till date are built using PKI introduced by Diffie and Hellman [8]. In PKI, each subscriber has a *descriptive information* and is associated with a private key and the corresponding public key. A trusted entity referred to as *Certifying Authority (CA)* issues a DSC which attests the public key with subscriber's *descriptive information*. One major limitation of PKI is the overhead of management and distribution of DSCs.

Further developments in cryptography introduced *Identity based Encryption (IBE)*, which was introduced as a concept by Shamir in 1984 [9] but was realized later by Boneh and Franklin in 2001 [10] using pairing based cryptography and by Cock using quadratic residues in 2001 [11]. In IBE, each subscriber has an identity and is associated with a private key and the corresponding public key. A trusted entity referred to as *Public Key Generator (PKG)* generates private key for the subscriber and gives it to him. The corresponding public key can be derived using subscriber's identity. Some major benefits of IBE over PKI are that public key of the subscriber can be derived from subscriber's identity without any overhead of certificate management.

Later developments in cryptography introduced Attribute based Encryption (ABE) which facilitates encryption based on a set of attributes. ABE is classified in two types, viz. a viz., Key Policy ABE (KP-ABE) and Ciphertext Policy ABE (CP-ABE). KP-ABE was introduced by Sahai and Waters [12] as an extension to the Identity based encryption and CP-ABE was introduced by Bethencourt et al [13]. These two schemes differ mainly on what is encoded (access policy or set of attributes) and where (in ciphertext or private key). In KP-ABE, access policy is encoded in subscriber's private key and a set of attributes are encoded in ciphertext. Only if the access policy encoded in receiver's private key satisfies the set of attributes encoded in received ciphertext will the receiver be able to decrypt the ciphertext. In CP-ABE, access policy is encoded in each ciphertext and a set of attributes are encoded in subscriber's private key. Only if the set of attributes encoded in receiver's private key satisfies the access policy encoded in received ciphertext, will the receiver be able to decrypt the ciphertext.

Attribute based Signature (ABS) is another related development in which a signer can sign a document with the proof of possession of certain attributes without revealing those attributes and his identity. Several researchers have worked on realizing the ABS scheme. Guo et al. [7] presented the ABS scheme which was proven using strong extended diffie hellman assumption. Later Tan [14] et al. presented that Guo's scheme is vulnerable to partial key replacement attack. Maji et al. [15] presented an ABS scheme which supported strong predicates containing AND, OR and threshold gates.

3 Preliminaries

This section briefly describes some of the necessary background.

3.1 Bilinear pairings [16]

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.

3.2 Decision bilinear diffie-hellman (DBDH) assumption [17]

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.

3.3 Strong extended diffie hellman (S-EDH) assumption [18]

Let $\mathbb{G}_1$, $\mathbb{G}_2$ are cyclic groups of prime order $p > 2^\lambda$ where $\lambda \in \mathbb{N}$, g_1 is the generator of $\mathbb{G}_1$, g_2 is the generator of $\mathbb{G}_2$, $\mathbb{O}_{x,y}(.)$ is an oracle that takes as input $m \in \mathbb{Z}_p^*$ and outputs $\langle g_1^r, g_2^{1/(x+r)}, g_2^{1/(m+r)}, g_2^{yr} \rangle$ for a random $r \in \mathbb{Z}_p^*$. For all probabilistic polynomial-time adversaries $\mathbb{A}$, all $v, c \in \mathbb{Z}_p^*$ and all $a \in \mathbb{G}_1$ such that $a \neq 1$, $\Pr[x \xleftarrow{R} \mathbb{Z}_p : \mathbb{A}^{O_{xy}}(g, g^x, g_2, g_2^y) = (m, a, a^x, a^r, g_2^{1/(x+r)}, g_2^{1/(m+r)}, g^{yr}) | m \notin Q] \prec 1/\text{poly}(k)$ where Q is the set of queries adversaries A make to oracle $\mathbb{O}_{x,y}(.)$.

3.4 Access structure

Access structure [19] is defined as follows. Let $P_1, P_2, ..., P_n$ be the set of parties. A collection $A \subseteq 2^{P_1, P_2, ..., P_n}$ is monotone if $B \in A$ and $B \subseteq C$ implies $C \in A$. An access structure is monotone collection A of non empty subsets of $\{P_1, P_2, .., P_n\}$ (that is, $A \subseteq 2^{P_1, P_2, ..., P_n} \setminus \{\phi\}$). The sets in A are called the authorized sets and the sets not in A are called the unauthorized sets. Access policy is generally represented by a monotone access structure implemented as an access tree.

Let $\mathcal{T}$ be an access 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 of a node x and $k_x = 1$, the threshold gate is an OR gate and when $k_x = num_x$, it is an AND gate. Each leaf node x of the tree is described by an attribute and a threshold value $k_x = 1$, it is an AND gate. Each leaf node x of the tree is described by an attribute and a threshold value $k_x = 1$. To facilitate working with the tree access structure, three functions are defined. Parent of the node x in the tree is denoted by parent(x). The function attr(x) in defined only if x is a leaf node and denotes the attribute associated with the leaf node x in the tree. The tree access structure $\mathcal{T}$ also defines an ordering between the children of every node, that, the children oaf a node

are numbered from 1 to num. The function index(x) returns such a number associated with the node x. Where the index values are uniquely assigned to nodes in the access structure for a viven key in an arbitrary manner.

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, evaluate $\mathcal{T}(y)$ 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 attr(x) $\in \lambda$.

4 Our Construction

This section presents the proposed scheme of privacy enhanced token based eSign using attribute based signature.

4.1 Attribute Authority

An attribute can be any characteristic of a subscriber and is represented by a private key (an integer) and a corresponding public key (a point on the group). Two new entities named *Attribute Authority Manager (AAM)* and *Attribute Authority (AA)* are proposed to be introduced. AAM manages the universe of attributes and AA manages a set of attributes assigned to him by AAM. The scheme consists of a single AAM and multiple AAs. UIDAI can assume the role of the AAM and individual agencies such as ESP, RTO, etc. can assume the role of AAs. Each subscriber also assumes the role of an AA since it also manages a small set of attributes representing his consent, purpose for which signature is taken, consumer of signature, etc. A new API `RegisterAsAA()` and a procedure `setup(k)` are proposed to be introduced by UIDAI for this purpose.

When UIDAI starts as AAM, it executes `setup(k)` procedure, where k is a security parameter. In this procedure, UIDAI chose two cyclic groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of large prime order p on which discrete logarithm problem is assumed to be hard, a generator g_1 of $\mathbb{G}_1$, a generator g_2 of $\mathbb{G}_2$, a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ for which bilinear diffie hellman problem is assumed to be hard. Security parameter k defines the size of chosen groups. Now, AAM defines universe of attributes $\mathbb{U} = \{1, 2, ...n\}$ and designate specific subset for specific AAs such as attributes $\{001 - 100\}$ for itself, $\{101 - 200\}$ for ESPs, $\{201 - 225\}$ for subscribers, etc. These attributes are represented by $\mathbb{A}_A$, $\mathbb{A}_E$, and $\mathbb{A}_U$ respectively. Subscriber is given an ownership of only few of the attributes which facilitates him provide his consent, designate consumer of the signature, designate purpose for which the signature is taken, etc.

Now, to define a private key for itself (SK), the corresponding public key (PK) and a master public key (MPK), AAM chose a random number $\gamma \in_R \mathbb{Z}_p$, random numbers $t_i \in_r \mathbb{Z}_p$ for each attribute $i \in \mathbb{A}_A$ and defines them as below.

$$\begin{aligned} SK &= \{\gamma, \{t_i\}_{\forall i \in \mathbb{A}_A}\} \\ PK &= \{g^\gamma, \{T_i, \{T_i\}_{PVTA}\}_{\forall i \in \mathbb{A}_A}\} \\ MPK &= \{\mathbb{A}_U, \mathbb{A}_E, \mathbb{A}_A, \mathbb{G}_1, \mathbb{G}_2, g_1, g_2, p\} \end{aligned} \quad (1)$$

where $T_i = g^{t_i}$ and $\{T_i\}_{PVTA}$ is T_i signed by another private key PVTA of AAM.

Before subscribing as an AA, each subscriber is assumed to have a secure device such as a smart card or a mobile having Trusted Execution Environment (TEE) which already has MPK in it and is capable of establishing a secure communication channel between itself and UIDAI. When a subscriber (or an agency) wants to register itself as an AA, it calls `RegisterAsAA()` API of UIDAI. UIDAI authenticates the requester using his Aadhaar number (or requester id) and requests him to provide public keys for subscriber (or AA_i) specific attributes. Requester generates a random number $\alpha \in_R \mathbb{Z}_p$, random numbers $r_i \in_R \mathbb{Z}_p$ for each attribute $i \in \mathbb{A}_U$ (or $i \in \mathbb{A}_i$), generates public key $T_i = g^{t_i}$ for each of them and sends them to UIDAI. UIDAI digitally signs α and each of T_i and sends $\{g^{\alpha}\}_{PVTA}, \{T_i\}_{PVTA}\ \forall i \in \mathbb{A}_U$ back to the subscriber (or AA). Subscriber (or AA_i)now has the private key $AASK_i$ and public key $AAPK_i$ as below.

$$\begin{aligned} AASK_i &= \{\alpha, \{t_i\}_{\forall i \in \mathbb{A}_U}\} \\ AAPK_i &= \{g^{\alpha}, \{g^{\alpha}\}_{PVTAA}, \{T_i, \{T_i\}_{PVTAA}\}_{\forall i \in \mathbb{A}_U}\} \end{aligned} \tag{2}$$

Requester (or AA_i) securely stores secret key USK in his secure device (or server).

4.2 Key Generation

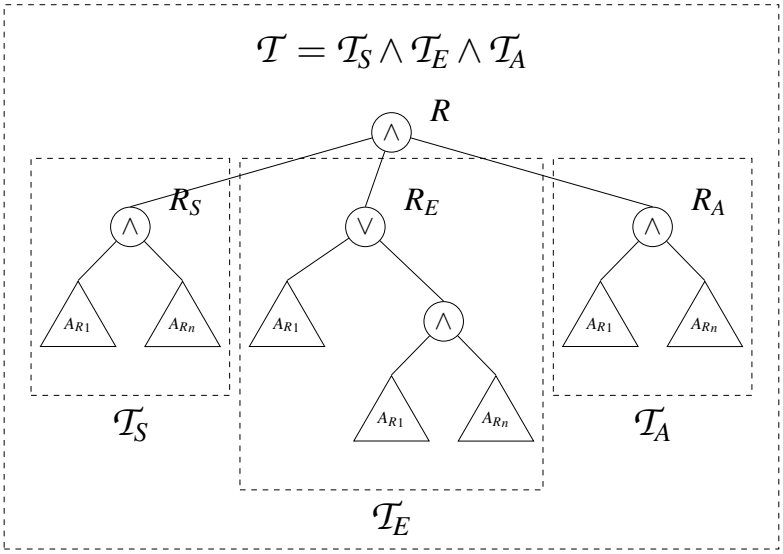
An *Attribute based Private Key (ABPvK)* can be generated for a subscriber based on his associated attributes and an access tree. Since a subscriber can be associated with attributes from different AAs such as RTO, University, etc, an access tree can have access subtrees from different AAs. An example of access tree $\mathcal{T}$ comprising of three access subtrees $\mathcal{T}_S$, $\mathcal{T}_E$ and $\mathcal{T}_A$ is illustrated in figure 2. All attributes from single AA are assumed to be in one access subtree and only ESP is assumed to have permission to request ABPvK on behalf of subscriber.

Each *Attribute based Token (ABT)* is identified by a tuple $IDT_{ij} = \langle ID_i, \mathcal{T}_j \rangle$ where ID_i is subscriber's identifier and $\mathcal{T}_j$ is the access tree against which ABT is to be generated. Let $K \in \mathbb{G}_1$ and $r \in_R \mathbb{Z}_p$. Presence of some helper functions is assumed such as $\text{GetAA}(\mathcal{T})$ returns set of AAs whose attributes are present in $\mathcal{T}$, $L((T)) = \text{leaves}(T) \cap \mathbb{A}$, where $\mathbb{A}$ is the set of attributes manged by AA calling the function, $\text{AASK}(\mathcal{T})$ returns first component from all $AASK_i$ of all $AA_i \in \text{GetAA}(\mathcal{T})$.

A helper procedure $\texttt{genParitalKey}(\texttt{IDT}_{ij}, K, r)$ is proposed to be introduced which works as follows. AA prepares a set λ of attributes associated with ID_i. A polynomial q_x is chosen for each node x (including the leaves) in access tree $\mathcal{T}_j$. The nodes in the tree are chosen in a top-down manner, starting from the root node R. For each node x, degree d_x of the polynomial q_x is set to one less than the threshold value k_x of that node, that is, $d_x = k_x - 1$. Now, for the root node R, set $q_R(0) = r$ and chose d other points randomly to define the polynomial $q(x)$ completely. For any other node x, set $q_x(0) = q_{\text{parent}(x)}(\text{index}(x))$ and chose d_x other points randomly to completely define q_x. Once the polynomials are decided, for each leaf node x, set the secret value $D_x = K^{q_x(0)/t_i}$ where $i = \text{att}(x)$ and $x \in \lambda$.

One API $\texttt{PullKey}(\texttt{IDT}_{ij}, K)$ is proposed to be introduced by AA which is consumed by UIDAI and returns ABPvK. Second API $\texttt{PullKeyAll}(\texttt{IDT}_{ij}, K)$ is proposed to be introduced by UIDAI which is consumed by ESP and returns ABPvK from all participating AAs. Refer algorithms [1 - 2].

Fig. 2: Example of an access policy tree



4.3 Token Generation

Since for bulk signatures, generation of ABPvK every time may be inefficient, an ABT is proposed to be introduced which contains ABPvK and associated token usage claims (such as expiry date) from all AA $\in$ GetAA($\mathcal{T}$). The ABT can be reused (till it expires) for every eSign request from the subscriber if those requests are against the same access tree.

To generate an ABT, all AA $\in$ GetAA($\mathcal{T}$) collaborate to arrive at a common group element K $\in \mathbb{G}_1$. For this, two APIs are proposed to be introduced by all AAs (including ESP), $\texttt{PullK}(\texttt{IDT}_{\texttt{ij}},\texttt{K})$ to pull an updated value of K and $\texttt{PushK}(\texttt{IDT}_{\texttt{ij}},\texttt{K})$ to let AA store updated value of K against IDT_{ij}. These APIs are consumed by UIDAI. One API $\texttt{GenTokPullAllK}(\texttt{IDT}_{\texttt{ij}},\texttt{K})$ is proposed to be introduced by UIDAI in which it facilitates arriving at a common group element K. This API is consumed by ESP. One API $\texttt{GenTok}(\texttt{IDT}_{\texttt{ij}},\texttt{K})$ is proposed to be introduced by ESP which is consumed by subscriber. Subscriber initiates this process by invoking its own procedure $\text{genTok}(\text{IDT}_{\text{ij}})$. Refer algorithms [3 - 7]. As seen in algorithm 6, ESP has created token ABT_{ij} which it keeps securely.

$$ABT_{ij} = \begin{cases} IDT_{ij} & = ID_i, \mathcal{T}_j \\ D_1 & = \bigcup_{\forall k} D_k \quad \mid AA_k \in GetAA(\mathcal{T}_j) \\ D_2 & = K^{\alpha}, K^{\beta}, K^{\gamma}, .. \mid \{\alpha,\beta,\gamma,..\} \in AASK(\mathcal{T}_j) \end{cases} \tag{3}$$

Algorithm 1 AA : PullKey

Require: $\langle IDT_{ij}, K\rangle$
  $r \in_R \mathbb{Z}_p$
  $\langle D_1, D_2\rangle \leftarrow$ genParitalKey(IDT_{ij}, K, ff)
    $D_1 = K^{q_x(0)/t_i} \quad \forall i \in L(IDT_{ij} \rightarrow T)$
    $D_2 = K^{ff}$
  return $\langle D_1, D_2\rangle$

Algorithm 2 UIDAI : PullKeyAll

Require: $\langle IDT_{ij}, K\rangle$
  $D_1 = D_2 = \phi$
  $AA \leftarrow$ GetAA$(IDT_{ij} \rightarrow T)$
  while AA $\neq$ empty **do**
    $AA_i \leftarrow$ DEQUEUE(AA)
    Call API of AA_i API :
      $\langle D_1\prime, D_2\prime\rangle \leftarrow$ PullKey(IDT_{ij}, K)
    $D_1 = D \cup D_1\prime$
    $D_2 = D, D_2\prime$
  end while
  return $\langle D_1, D_2\rangle$

Algorithm 3 AA : PullK

Require: $\langle IDT_{ij}, K\rangle$
  $r \in_R \mathbb{Z}_p$
  Store mapping : $IDT_{ij} \leftrightarrow r$
  return K^r

Algorithm 4 AA : PushK

Require: $\langle IDT_{ij}, K\rangle$
  $r \in_R \mathbb{Z}_p$
  Update mapping : $IDT_{ij} \leftrightarrow K$
  return

Algorithm 5 UIDAI : GenTokPullAllK

Require: $\langle IDT_{ij}, K\rangle$
  $AA \leftarrow$ GetAA$(IDT_{ij} \rightarrow T)$
  while AA $\neq$ empty **do**
    $AA_i \leftarrow$ DEQUEUE(AA)
    Call AA_i API : $K \leftarrow$ PullK(IDT_{ij}, K)
  end while
  $AA \leftarrow$ GetAA$(IDT_{ij} \rightarrow T)$
  while AA $\neq$ empty **do**
    $AA_i \leftarrow$ DEQUEUE(AA)
    Call AA_i API : PushK(IDT_{ij}, K)
  end while
  return K

Algorithm 6 ESP : GenTok

Require: $\langle IDT_{ij}, K\rangle$
  $r \in_R \mathbb{Z}_p$
  $K \leftarrow K^r$
  $K \leftarrow$ UIDAI : GenTokPullAllK(IDT_{ij}, K)
  Store mapping : $IDT_{ij} \leftrightarrow K$
  $\langle D_1, D_2\rangle \leftarrow$ UIDAI : PullKeyAll(IDT_{ij}, K)
  $IDT_{ij} =$
    $ID_{ij} : ID_i, \mathcal{T}_j$
    $D_1 : D_U \cup D_{AA_1} \cup D_{AA_2} \cup ...$
    $D_2 : K^{\alpha}, K^{\beta}, K^{\gamma}, ...$
  return

Algorithm 7 Subscriber : genTok

Require: $\langle IDT_{ij}\rangle$
  $r \in_R \mathbb{Z}_p$
  $K \leftarrow g^r$
  $K \leftarrow$ ESP : GenTok$(IDT_{ij}, K, \mathcal{T})$
  Store mapping : $IDT_{ij} \leftrightarrow K$
  return

4.4 eSign using token

A new version of eSign API $\texttt{eSign}(\texttt{ID}_\texttt{i}, \mathcal{T}_\texttt{j}, \texttt{H}(\texttt{m}))$ is proposed to be introduced by ESP where $\mathrm{H}(\mathrm{m})$ is one way secure hash of message to be signed. When ESP receives this request, it generates a random number $\mathrm{r}_4 \in_\mathrm{R} \mathbb{Z}_\mathrm{p}$ and computes the signature σ as below. The signature is then given back to subscriber.

$$\sigma_{ij} = \left\{ \begin{array}{l} A = g^{r_4} \\ C = g^{\frac{1}{r_4 + H(m)}} \\ D = \{K^{\alpha}\}^{r_4} \cdot \{K^{\beta}\}^{r_4} \cdot \{K^{\gamma}\}^{r_4} \ldots \\ \quad = g^{r_4(\prod_{\forall k} r_k)(\sum_{\forall k} AASK_k)} \\ E_i = D_k{}^{r_4} = g^{r_4(\prod_{\forall k} r_k)(\frac{q_x(0)}{t_i})} \end{array} \right\}_{|AA_k \in GetAA(\mathcal{T})} \tag{4}$$

4.5 eSign Verification

To verify an eSigned document, an offline procedure $\mathrm{Verify}(\mathrm{M}, \sigma, \mathrm{MPK})$ is proposed to be introduced. A recursive helper procedure $\mathrm{VerN}(\mathrm{T_i}, \mathrm{E_i}, \mathrm{i})$ is defined. For each leaf node x in $\mathcal{T}$, the helper procedure takes, three parameters, public key of the attribute, corresponding private key component from signature and $\mathrm{i} = \mathrm{attr}(\mathrm{x})$. This is defined as below.

$$VerN(T_i, E_i, x) = \begin{cases} e(T_x, E_x) & \text{if } i \in \gamma \\ \perp & \text{otherwise} \end{cases} \tag{5}$$

For each non-leaf node x in access tree $\mathcal{T}$, the procedure is defined as follows. For all nodes z that are children of x, it calls $\mathrm{VerN}(\mathrm{T_z}, \mathrm{E_z}, \mathrm{z})$ and stores the output as $\mathrm{F_z}$. Let $\mathrm{S_x}$ be an arbitrary $\mathrm{k_x} -$ sized set of child nodes z such that $\mathrm{F_z} \neq \perp$. If no such set exists then the node was not satisfied and the function returns $\perp$. Otherwise, $\mathrm{F_x}$ is computed as below.

$$\begin{aligned} F_x &= \prod_{z \in S_x} F_z^{\Delta_{i,S_x\prime}(0)} \qquad where\{i = index(z) \\ &\qquad\qquad S_x\prime = \{index(z) : z \in S_x\} \\ &= \prod_{z \in S_x} F_z^{\Delta_{i,S_x\prime}(0)} \\ &= \prod_{z \in S_x} (e(g,g)^{rr_4 q_z(0)})^{\Delta_{i,S_x\prime}(0)} \\ &= \prod_{z \in S_x} (e(g,g)^{rr_4 q_{parent(z)}(index(z))})^{\Delta_{i,S_x\prime}(0)} \\ &= \prod_{z \in S_x} e(g,g)^{rr_4 q_x(i) \Delta_{i,S_x\prime}(0)} \\ &= e(g,g)^{rr_4 q_x(0)} \; using\ polynomial\ interpolation \end{aligned} \tag{6}$$

It can be deduced that for the root node R, $\mathrm{VerN}(\mathrm{E_R}, \mathrm{T_R}, \mathrm{R})$ returns $\mathrm{e}(\mathrm{g}, \mathrm{g})^{r_4(\prod_{\forall k} r_k)(\Sigma_{\forall k} AASK_k)}$ if and only if the signature satisfies the access tree $\mathcal{T}_\mathrm{R}$.

Now, the signature verifier verifies following equalities.

$$e(g,D) \stackrel{?}{=} F_R$$
$$e(g^m.A,C) \stackrel{?}{=} e(g,g) \quad (7)$$

Only if these equalities hold true, the verifier accepts the signature.

5 Security Analysis

This section presents the security analysis of the proposed scheme based on intuitive reasoning. To keep the focus on objective of this paper, it is assumed that communication channel between different entities is secure and before communicating any message, both entities authenticate each other. This will ensure confidentiality, data integrity and mutual authentication.

5.1 Privacy

The signed document and the signature on it does not reveal any information about identity of the user. The signature is done using attributes of the user and does not include identity of the user. The signature verifier also does not need to know the identity of the user to verify the signature. Hence, privacy of the user is maintained.

5.2 Unforgeability

The proposed scheme is existentially unforgeable under chosen-message attack under the strong extended diffie hellman assumption. This is true since it it is not the case, and the scheme is forgeable with a non-negligible probability ε, then the strong extended diffie hellman assumption can also be broken with the same non-negligible probability ε. Moreover, since user's key is never given to the user himself and ESP is a trusted entity, partial key replacement attack will not be possible.

6 Performance Analysis

This section presents performance analysis of the proposed scheme. However, since present model of eSign is based on PKI, results of this analysis cannot be compared directly with the present model of eSign. This section assumes presence of two procedures, viz. a viz., L($\mathcal{T}$) and NL($\mathcal{T}$) which returns the set of leaves and non-leaves in access tree $\mathcal{T}$. Table 1 depicts various costs of each phase (in terms of number of operations) for each participating entity. Three columns of this table indicate the number of signing operations, exponent operations and the pairing operations.

The major procedures in this scheme are setup(), userRegistrationAA($\mathrm{ID_{U_i}}$), esp RegistrationAA(), tokenGeneration(), eSign(H(m), $\mathcal{T}$) and eVerification(σ, H(m)). For analysis, we will consider two helper procedures, genPartialKey() and mutually ArriveAtK().

Table 1: Cryptographic cost

			Signing	Exponent	Pairing
h	Setup	User			
		ESP			
		AA	$\|\mathcal{A}_{\mathrm{A}}\|+1$	$\|\mathcal{A}_{\mathrm{A}}\|+1$	
	User/ESP Registration	User	$\|\mathcal{A}_{\mathrm{U_i}}\|+1$	$\|\mathcal{A}_{\mathrm{U_i}}\|+1$	
		ESP	$\|\mathcal{A}_{\mathrm{E_i}}\|+1$	$\|\mathcal{A}_{\mathrm{E_i}}\|+1$	
		AA			
	Token Generation	User		$\|\mathrm{L}(\mathcal{T}_{\mathrm{U_i}})\|+2$	
		ESP	1	$\|\mathrm{L}(\mathcal{T}_{\mathrm{E_i}})\|+2$	
		AA	1	$\|\mathrm{L}(\mathcal{T}_{\mathrm{A}})\|+2$	
	eSign	User			
		ESP		$\|\mathrm{L}(\mathcal{T})\|+5$	
		AA			
	eSign Verification	User		$\|\mathrm{NL}(\mathcal{T})\|$	$\|\mathrm{L}(\mathcal{T})\|+2$
		ESP			
		AA			

The setup procedure is executed by attribute authority at the very begining to set up its private key ASK, the corresponding public key APK and the master public key MPK. ASK consists of a set of integers chosen for attributes in $\mathbb{A}_{\mathrm{A}}$. In APK, corresponding public key for each attribute is arrived by raising g to the power of the respective private chosen integer. In addition to this, the APK component is also arrived by raising g to the power of secret random number γ. Hence, this procedure involves $|\mathbb{A}_{\mathrm{A}}|+1$ exponentiation and $|\mathbb{A}_{\mathrm{A}}|$ signatures by attribute authority.

In user registration procedure, g^{α} involves one exponentiation and user attribute based public key UPK is arrived by raising g the corresponding components in USK. This involves $(\mathbb{A}_{\mathrm{U_i}}+1)$ exponentiations, $(\mathbb{A}_{\mathrm{U_i}}+1)$ signatures by attribute authority and 1 Aadhaar based authentication. ESP registration procedure is also similar to user registration procedure and will involve $(\mathbb{A}_{\mathrm{E_i}}+1)$ exponentiation, $(\mathbb{A}_{\mathrm{E_i}}+1)$ signatures by attribute authority and 1 authentication.

In helper function $\mathrm{genKey}(\mathcal{T}_{\mathrm{i}}, \mathrm{ID}, \mathrm{K})$, a polynomial is created for every node (including the leaf nodes) and for each leaf node representing an attribute, K is raised to the power of $\mathrm{q_x}(0)/\mathrm{t_i}$. Ignoring the polynomial creation cost, this will involve $|\mathrm{L}(\mathcal{T}_{\mathrm{i}}\mathrm{i})|$ exponentiation.

In helper procedure mutuallyArriveAtK(), K, K_{α}, K_{β} and K_{γ} are arrived by using 6 exponentiation.

Procedure genToken() involves three invocations of $\mathrm{genKey}(\mathcal{T}_{\mathrm{i}}, \mathrm{ID}, \mathrm{K})$ for $\mathcal{T}_{\mathrm{U_i}}$, $\mathcal{T}_{\mathrm{E_i}}$ and $\mathcal{T}_{\mathrm{A}}$ which will involve $|\mathrm{L}(\mathcal{T})|$ exponentiation, one invocation of mutuallyArriveAtK() which will involve 6 exponentiation. Other than that, σ_{E_i} and σ_A contributes two signatures, one by $\mathrm{E_i}$ and one by AA.

In procedure, $\mathrm{eSign}(\mathcal{T}_{\mathrm{q}}, \mathrm{H(m)})$, D is computed using 3 exponentiation, A is computed using 1 exponentiation, C is computed using 1 exponentiation and 1 hash and $\mathrm{E_i}$ is computed using $|\mathrm{L}(\mathcal{T})|$ exponentiation. This involves a total of $|\mathrm{L}(\mathcal{T})|+5$ exponentiation and 1 hash.

In eSignVerification() procedure, for each leaf node of $\mathcal{T}$, a paring is computed, for each internal node of $\mathcal{T}$ the values obtained from child are raised to lagrange's coeffi-

cient and then multiplied. Further to this, 2 pairings are computed for final verification. This involves $|\mathrm{L}(\mathcal{T})|+2$ pairings and $|\mathrm{NL}(\mathcal{T})|$ exponentiation.

Table 1 depicts the cryptographic cost of various entities in various phases with respect to signing, exponent and pairing operations.

Setup, UserRegistration and ESPRegistraion are one time operations. Based on regulatory guidelines, ESP can keep the token and if possible, reuse the same later for multiple eSign requests. For bulk eSign operations TokenGeneration is also a one time operation. The recurring cost is only for eSign which is $|(\mathrm{L}(\mathcal{T}))|+5$ exponents and which is beared by ESP. Thus, the amortized cost of eSign grows linear to the number of attributes in terms of exponents, ($\mathrm{AmortizedCost}_{\mathrm{eSign}} = O(\mathrm{L(T)})\mathrm{Cost}_{\mathrm{exponent}}$) .

7 Conclusion and Future Work

Recently, Government of India has taken several initiatives to make India digitally strong which are acclaimed by many but have also raised privacy concerns. eSign is one of these initiatives. Although recent developments in cryptography have introduced ABS which can address privacy related concerns, efficiency and right implementation of it are still some of the major challenges. This paper proposed a scheme to implement privacy enhanced eSign using ABS. The paper also presented the security and performance analysis of the scheme.

The privacy aspects of the proposed scheme can further be improved by mechanisms such as oblivious transfer [20] and secure multi party computation [21] which can help enhance privacy from server perspective in which even though the server provides information to the requester but remains oblivious as to what piece of information has been provided

References

1. Government of India, Digital india. https://www.digitalindia.gov.in, 2015.
2. UIDAI, Aadhaar. https://uidai.gov.in/what-is-aadhaar.html, 2009.
3. UIDAI, . Unique identification authority of india. https://uidai.gov.in, 2009.
4. CCA, esign service. http://cca.gov.in/cca/?q=eSign.htmll, 2009.
5. CCA, PKI framework in india. http://www.cca.gov.in/cca/?q=pki_framework.html, 2020.
6. CCA, Empanelled esign service providers. http://www.cca.gov.in/cca/?q=service-providers.html, 2017.
7. Shanqing, G. and Yingpei, Z., Attribute-based signature scheme. In 2008 International Conference on Information Security and Assurance (ISA 2008), Pp. 509-511, IEEE, 2008.
8. Diffie, W. and Hellman, M., New directions in cryptography. IEEE transactions on Information Theory, 22(6):644-654, 1976.
9. Shamir, A., Identity-based cryptosystems and signature schemes. In Workshop on the theory and application of cryptographic techniques, Pp. 47-53. Springer, 1984.
10. Boneh, D. and Franklin, M., Identity-based encryption from the weil pairing. In Annual international cryptology conference, Pp. 213–229, 2001 Springer.
11. Cocks, C., An identity based encryption scheme based on quadratic residues. In IMA international conference on cryptography and coding, Pp 360-363, Springer, 2001.

12. Goyal, V., et al., Attribute-based encryption for fine-grained access control of encrypted data. In Proceedings of the 13th ACM conference on Computer and communications security, Pp. 89-98, 2006.
13. Bethencourt, J., et al., Ciphertext-policy attribute-based encryption. In 2007 IEEE symposium on security and privacy (SP'07), Pp. 321–334, IEEE, 2007.
14. Tan, S. et al., On the security of an attribute-based signature scheme. In International Conference on U-and E-Service, Science and Technology, Pp. 161-168. Springer, 2009.
15. Maji, H., et al., Attribute-based signatures. In Cryptographers' track at the RSA conference, Pp. 376-392, Springer, 2011.
16. Zhang, F., et al., An efficient signature scheme from bilinear pairings and its applications. In International Workshop on Public Key Cryptography, Pp. 277-290. Springer, 2004
17. Yacobi, Y., A note on the bilinear diffie-hellman assumption. IACR Cryptology ePrint Archive, 2002:113.
18. Ateniese, G., et al., Practical group signatures without random oracles. IACR Cryptology ePrint Archive, 2005.
19. Beimel, A., et al., Secure schemes for secret sharing and key distribution. Technion-Israel Institute of technology, Faculty of computer science, 1996.
20. Rabin Michael O., How to exchaneg secrets with Oblivious Transfer. IACR Cryptology ePrint Archive, 2005.
21. Goldreich, Oded, Secure multi-party computation. Manuscript, Preliminary Version 78, 1998.