Temporal Access Control with User Revocation without proxy decryptor for Cloud Data

Ayan Das*, Sushmita Ruj†
* R.C. Bose Center for Cryptology and Security, Indian Statistical Institute, Kolkata, India – ayandas.infy@gmail.com
† R.C. Bose Center for Cryptology and Security, Indian Statistical Institute, Kolkata, India – sush@isical.ac.in

***Abstract*—In this scheme we propose a temporal access control scheme to protect and selectively access data in clouds without any proxy server and with revocation facility which is besically the improvement of the previous scheme. In this paper we introduced a new version of temporal access scheme where the data owner can revoke any valid user from accessing the encrypted content in any point of time.**

Keywords: Access Control, Cloud Storage, CP-ABE, Temporal Access Control, Revocation

A. Framework

A brief description of each of these algorithms is as follows:

- Setup$(1^\kappa) \rightarrow (MK, PK)$: Takes a security parameter κ as input, outputs the master secret key MK and the public key PK.
- KeyGen$(MK, u_k, \mathcal{L}) \rightarrow SK_{\mathcal{L}}$: Takes the user ID u_k, the access privilege $\mathcal{L}$, and MK as input, outputs the users private key $SK_{\mathcal{L}}$.
- Encrypt phase-1$(PK, M, \mathcal{P}) \rightarrow (\mathcal{H}_{\mathcal{P}}, C')$: Takes a comparable access policy $\mathcal{P}$, public key PK and message M as input, outputs the ciphertext header $\mathcal{H}_{\mathcal{P}}$ and ciphertext C'.
- Encrypt phase-2$(PK, MK, RL) \rightarrow (\widetilde{\mathcal{H}_{\mathcal{P}}}, RIV)$: Takes the public key PK, master key MK, and the revocation list RL as inputs, outputs modified ciphertext $\widetilde{\mathcal{H}_{\mathcal{P}}}$, which contains the information necessary to enable the data owner to partially decrypt the ciphertext.
- Delegate$(SK_{\mathcal{L}}, \mathcal{L}') \rightarrow \widetilde{SK}_{\mathcal{L}'}$: Takes a private key $SK_{\mathcal{L}}$ and a specified privilege requirement $\mathcal{L}'$ as input, outputs a derivation key $\widetilde{SK}_{\mathcal{L}'}$, if each attribute in $\mathcal{L}$ and $\mathcal{L}'$ match.
- Decrypt:-1$(\widetilde{SK}_{\mathcal{L}'}, \mathcal{H}_{\mathcal{P}}, RIV) \rightarrow \widetilde{\mathcal{H}_{\mathcal{P}}}$: Takes a users private key $\widetilde{SK}_{\mathcal{L}'}$, RIV and the ciphertext header $\mathcal{H}_{\mathcal{P}}$ as input, outputs a new ciphertext header $\widetilde{\mathcal{H}_{\mathcal{P}}}$ if $\mathcal{L}'$ satisfies the range constraints of $\mathcal{P}$.
- Decrypt:-2r$(SK_{\mathcal{L}}, \widetilde{\mathcal{H}_{\mathcal{P}}}, C') \rightarrow M$. Takes the users private key $SK_{\mathcal{L}}$, Ciphertext header $\widetilde{\mathcal{H}}_{\mathcal{P}}$, and the ciphertext C' as input, outputs the message M.

I. Construction

We provide detailed implementations of the algorithms mentioned in section -A and present our novel construction to enforce user revocation.

- *Setup*$(1^\kappa) \rightarrow (MK, PK)$
  This is run by the system manager. Let κ be the security parameter. The setup algorithm takes κ (which depends on the application) and outputs the master secret key MK and public key PK. We use bilinear map $\mathbb{S} = (\mathbb{G}, \mathbb{G}_T, e(\cdot,\cdot))$ of composite order $n = s'n'$ with two subgroups $\mathbb{G}_{s'}$ and $\mathbb{G}_{n'}$ of $\mathbb{G}$. Random generators $w \in \mathbb{G}$, $g \in \mathbb{G}_{s'}$, and $\varphi, \bar{\varphi} \in \mathbb{G}_{n'}$ are chosen. Two random numbers $\lambda, \mu \in \mathbb{Z}_n^*$ are selected, to implement forward and backward derivation functions. By Proposition 1, we have $e(g, \varphi) = e(g, \bar{\varphi}) = 1$, but $e(g, w) \neq 1$. A hash function $H : \{0,1\}^* \rightarrow \mathbb{G}$ is used to map an attribute to a binary string which represents a random group element. A randomly generated polynomial P of degree c (the maximum number of revoked users) over $\mathbb{Z}_n$ will be employed to enforce revocation. Next, this algorithm chooses two random exponents $\alpha, \beta \in \mathbb{Z}_n^*$ and outputs the public key

$$PK = (\mathbb{S}, n, g, h, \zeta, \eta, w, \varphi, \bar{\varphi}, \lambda, \mu, H(\cdot)),$$

where,

$$h = w^\beta, \zeta = e(g, w)^\alpha, \eta = g^{\frac{1}{\beta}},$$

and the master key

$$MK = (g^\alpha, \beta, n', P).$$

- *KeyGen*$(MK, u_k, \mathcal{L}) \rightarrow SK_{\mathcal{L}}$

  Given a user with ID u_k[1] and access structure $\mathcal{L}$ on a set of attributes $S = \{A_t\} \subseteq \mathcal{A}$, this algorithm chooses a unique τ_k for each user u_k. Assume that the user u_k is assigned a temporal attribute $A_t \in \mathcal{L}$ with the constraint $A_t[t_a, t_b]$ and non-temporal attribute $B_t \in \mathcal{L}$. This algorithm chooses a random $r \in \mathbb{Z}$ and sets the user's attribute key as

$$(D_t, D_{t_a}^{'}, \bar{D}_{t_b}^{'}, D_t^{''}, R_{t_a}^{'}, \bar{R}_{t_b}^{'}, R_t^{''})_{A_t[t_a,t_b] \in \mathcal{L}} \text{and}$$

$$(DB_t, RB_t^{''})_{B_t \in \mathcal{L}}$$

where,

$$\begin{aligned}
D_t &= g^{\tau_k} H(A_t)^{r\widetilde{P}(0)}, \\
D_{t_a}^{'} &= (v_{t_a})^r, \\
\bar{D}_{t_b}^{'} &= (\bar{v}_{t_b})^r, \\
D_t^{''} &= w^r, \\
R_{t_a}^{'} &= (D_{t_a}^{'})^{P(u_k)} = (v_{t_a})^{rP(u_k)}, \\
\bar{R}_{t_b}^{'} &= (\bar{D}_{t_b}^{'})^{P(u_k)} = (\bar{v}_{t_b})^{rP(u_k)}, \\
R_t^{''} &= (D_t^{''})^{P(u_k)} = w^{rP(u_k)}, \\
DB_t &= g^{\tau_k} H(B_t)^{r\widetilde{P}(0)}, \\
RB_t^{''} &= (D_t^{''})^{P(u_k)} = w^{rP(u_k)},
\end{aligned}$$

[1] We assume that each user ID $u_k \in \mathbb{Z}_l$ where $l = min(s, p', q')$, so that, for every pair $(i, j), i \neq j, u_i - u_j \in \mathbb{Z}_n^*$. As a result, its inverse $(u_i - u_j)^{-1}$, which is needed in the *ProxyRekey* phase will exist and the secret can be successfully recovered.

and,

$$v_{t_a} = \varphi^{\lambda^{t_a}}, \bar{v}_{t_b} = \bar{\varphi}^{\mu^{Z-t_b}} \in \mathbb{G}_{n'},$$
$$\widetilde{P}(0) = P(0) + 1.$$

Finally, this algorithm outputs the user's private key

$$SK_{\mathcal{L}} = (D = g^{\frac{(\alpha+\tau_k)}{\beta}}, \{(D_t, D_{t_a}^{'}, \bar{D}_{t_b}^{'}, D_t^{''}, R_{t_a}^{'}, \bar{R}_{t_b}^{'}, R_t^{''})\}_{A_t[t_a,t_b]\in\mathcal{L}}, \{(DB_t, RB_t'')\}_{B_t\in\mathcal{L}}).$$

- *Encrypt Phase 1*$(PK, \mathcal{P}, M) \rightarrow (\mathcal{H}_{\mathcal{P}}, C')$
Let $\mathcal{T}$ be the access tree for the access policy $\mathcal{P}$. Let s be a secret in $\mathbb{Z}_N$ for the access tree $\mathcal{T}$, and x, y are chosen such that $x + y = q_{A_t}(0)$, which is the secret share at leaf node corresponding to attribute A_t. Please refer to Section 4, [**?**] for a detailed meaning of $q_{A_t}(0)$. Given an access tree $\mathcal{T}$, the ciphertext is composed of a ciphertext header,

$$\mathcal{H}_{\mathcal{P}} = (\mathcal{T}, C = h^s, \{(\bar{E}_{t_i}, E_{t_i}^{'}, E_{t_j}, E_{t_j}^{'})\}_{A_t[t_i,t_j]\in\mathcal{P}}, \{(EB_t, EB_t')\}_{B_t\in\mathcal{P}}).$$

and a ciphertext $C' = Me(g, w)^{s\alpha}$, where,

$(\bar{E}_{t_i}, E_{t_i}^{'}, E_{t_j}, E_{t_j}^{'}) = ((\bar{v}_{t_i}w)^x, H(A_t)^x, (v_{t_j}w)^y, H(A_t)^y)$
and $(EB_t, EB_t') = (\omega^{q_{B_t}(0)}, H(B_t)^{q_{B_t}(0)})$.

- *Encrypt Phase 2*$(PK, MK, RL) \rightarrow RIV$
The data owner chooses a polynomial P of degree c, with coefficients in Z_n^*. Let RL be the revocation list and $u_i, i \in \{1, 2, \cdots, c\}$, be the identities of the revoked users. The owner evaluates the polynomial $P(u_i)$ at these points, using the master secret key MK. If there are less than c revoked users, then the owner generates random points x and evaluates $P(x)$ such that x does not correspond to any user's identity. This ensures that the proxy key RIV is of fixed length.

$$RIV = \forall u_i \in RL :< u_i, P(u_i) >$$

Now the data owner will calculates

$$\lambda_i = \frac{u_k}{u_k - u_i} \prod_{j\neq i} \frac{u_j}{u_j - u_i}, \forall i, j \in \{1, \cdots, c\}, k \notin \{1, \cdots, c\}$$

For every attribute $A_t[t_i, t_j] \in \mathcal{P}$, it calculates

$$R_{t_i}^{''} = (E_{t_i}^{'})^{\sum_{i=1}^{c} \lambda_i P(u_i)} = H(A_t)^{x \sum_{i=1}^{c} \lambda_i P(u_i)},$$
$$R_{t_j}^{''} = (E_{t_j}^{'})^{\sum_{i=1}^{c} \lambda_i P(u_i)} = H(A_t)^{y \sum_{i=1}^{c} \lambda_i P(u_i)},$$
$$RB_t^{'''} = (EB_t^{'})^{\sum_{i=1}^{c} \lambda_i P(u_i)} = H(B_t)^{q_{B_t}(0) \sum_{i=1}^{c} \lambda_i P(u_i)}.$$

Now the new ciphertext will be

$$\widetilde{\mathcal{H}_{\mathcal{P}}} = (\mathcal{T}, C = h^s, \{(\bar{E}_{t_i}, E_{t_i}^{'}, E_{t_j}, E_{t_j}^{'}, R_{t_i}^{''}, R_{t_j}^{''})\}_{A_t[t_i,t_j]\in\mathcal{P}}, \{(EB_t, EB_t', , RB_t^{'''})\}_{B_t\in\mathcal{P}}).$$

Since the user's private key $SK_{\mathcal{L}}$ is blinded by $\widetilde{P}(0)$, it additionally needs $R_{t_i}^{''}$, $R_{t_j}^{''}$ for decryption. The data owner also computes λ_k and gives it to the user with ID u_k.

- *Delegate*$(SK_{\mathcal{L}}, \mathcal{L}') \rightarrow \widetilde{SK}_{\mathcal{L}'}$

Given the private key $SK_{\mathcal{L}}$ and a specified $\mathcal{L}'$, this algorithm checks whether, for every $A_t[t_a, t_b] \in \mathcal{L}$ and $A_t[t_i, t_j] \in \mathcal{L}'$, $t_a \leq t_j$ and $t_b \geq t_i$ is true for each attribute $A_t \in \mathcal{L}'$.
If true, the user computes,

$$D_{t_j}^{'} \leftarrow f_{t_a\leq t_j}(D_{t_a}^{'}) \cdot D_t^{''} = (v_{t_j}w)^r$$
$$\bar{D}_{t_i}^{'} \leftarrow \bar{f}_{t_b\geq t_i}(\bar{D}_{t_b}^{'}) \cdot D_t^{''} = (\bar{v}_{t_i}w)^r$$
$$R_{t_j}^{'} \leftarrow f_{t_a\leq t_j}(R_{t_a}^{'}) = (v_{t_j})^{rP(u_k)}$$
$$\bar{R}_{t_i}^{'} \leftarrow \bar{f}_{t_b\geq t_i}(\bar{R}_{t_b}^{'}) = (\bar{v}_{t_i})^{rP(u_k)}$$

$$\begin{aligned}
R_{t_j} &= e(R_t^{''} \cdot R_{t_j}^{'}, E_{t_j}^{'})^{\lambda_k} \cdot e(D_{t_j}^{'}, R_{t_j}^{''}) \\
&= e(v_{t_j}w, H(A_t))^{yr(\lambda_k P(u_k) + \sum_{i=1}^{t} \lambda_i P(u_i))} \\
&= e(v_{t_j}w, H(A_t))^{yrP(0)} \\
\bar{R}_{t_i} &= e(R_t^{''} \cdot \bar{R}_{t_i}^{'}, E_{t_i}^{'})^{\lambda_k} \cdot e(\bar{D}_{t_i}^{'}, \bar{R}_{t_i}^{''}) \\
&= e(\bar{v}_{t_i}w, H(A_t))^{xr(\lambda_k P(u_k) + \sum_{i=1}^{t} \lambda_i P(u_i))} \\
&= e(\bar{v}_{t_i}w, H(A_t))^{xrP(0)} \\
RB_t &= e(RB_t^{''}, EB_t^{'})^{\lambda_k} \cdot e(DB_t^{'}, RB_t^{'''}) \\
&= e(w, H(B_t))^{q_{B_t}(0)rP(0)}
\end{aligned}$$

Finally, it outputs $\widetilde{SK}_{\mathcal{L}'}$ as the derivation key for $\mathcal{L}'$.

- *Decrypt Phase 1*$(\widetilde{SK}_{\mathcal{L}'}, \mathcal{H}_{\mathcal{P}}) \rightarrow \widetilde{\mathcal{H}}_{\mathcal{P}}$

On receiving the private key $\widetilde{SK}_{\mathcal{L}'}$, and the ciphertext header $\mathcal{H}_{\mathcal{P}}$, this algorithm checks whether each range attribute $A_t[t_i, t_j] \in \mathcal{L}'$ is consistent with $A_t[t_i, t_j] \in \mathcal{P}$. If true, the secret share $q_{A_t}(0)$ over $\mathbb{G}_T$ is reconstructed

$$\begin{aligned}
F_1 &\longleftarrow \frac{e(D_t, E_{t_j})}{R_{t_j} \cdot e(D_{t_j}^{'}, E_{t_j}^{'})} \\
&= \frac{e(g^{\tau_k} H(A_t)^{r\widetilde{P}(0)}, (v_{t_j}w)^y)}{R_{t_j} \cdot e((v_{t_j}w)^r, H(A_t)^y)} \\
&= \frac{e(g^{\tau_k}, (v_{t_j}w)^y) \cdot e(H(A_t)^{r\widetilde{P}(0)}, (v_{t_j}w)^y)}{R_{t_j} \cdot e((v_{t_j}w)^r, H(A_t)^y)} \\
&= \frac{e(g^{\tau_k}, (v_{t_j}w)^y) \cdot e(v_{t_j}w, H(A_t))^{yrP(0)}}{R_{t_j}} \\
&= e(g^{\tau_k}, v_{t_j}^y) \cdot e(g^{\tau_k}, w^y) \\
&= e(g^{\tau_k}, w)^y
\end{aligned}$$

Similarly,

$$
\begin{aligned}
F_2 \quad \longleftarrow \quad & \frac{e(D_t, \bar{E}_{t_i})}{\bar{R}_{t_i} \cdot e(\bar{D'}_{t_i}, E'_{t_i})} \\
= \quad & \frac{e(g^{\tau_k} H(A_t)^{rP'(0)}, (\bar{v}_{t_i} w)^x)}{\bar{R}_{t_i} \cdot e((\bar{v}_{t_i} w)^r, H(A_t)^x)} \\
= \quad & \frac{e(g^{\tau_k}, (\bar{v}_{t_i} w)^x) \cdot e(H(A_t)^{rP'(0)}, (\bar{v}_{t_i} w)^x)}{\bar{R}_{t_i} \cdot e((\bar{v}_{t_i} w)^r, H(A_t)^x)} \\
= \quad & \frac{e(g^{\tau_k}, (\bar{v}_{t_i} w)^x) \cdot e(\bar{v}_{t_i} w, H(A_t))^{xrP(0)}}{\bar{R}_{t_i}} \\
= \quad & e(g^{\tau_k}, \bar{v}_{t_i}^x) \cdot e(g^{\tau_k}, w^x) \\
= \quad & e(g^{\tau_k}, w)^x
\end{aligned}
$$

For $B_t \in \mathcal{L}'$ is consistent with $B_t \in \mathcal{P}$, then the secret share $q_{B_t}(0)$ of s over $\mathbb{G}_T$ is reconstructed as,

$$
\begin{aligned}
F_3 \quad \longleftarrow \quad & \frac{e(D_t, EB_t)}{RB_t \cdot e(D''_t, EB'_t)} \\
= \quad & \frac{e(g^{\tau_k} H(B_t)^{rP'(0)}, w^{q_{B_t}(0)})}{RB_t \cdot e(w^r, H(B_t)^{q_{B_t}(0)})} \\
= \quad & \frac{e(g^{\tau_k}, w^{q_{B_t}(0)}) \cdot e(H(B_t)^{r\widetilde{P}(0)}, w^{q_{B_t}(0)})}{RB_t \cdot e(w^r, H(B_t)^{q_{B_t}(0)})} \\
= \quad & \frac{e(g^{\tau_k}, w^{q_{B_t}(0)}) \cdot e(w, H(B_t))^{q_{B_t}(0) rP(0)}}{RB_t} \\
= \quad & e(g^{\tau_k}, w^{q_{B_t}(0)}) \\
= \quad & e(g^{\tau_k}, w)^{q_{B_t}(0)}
\end{aligned}
$$

$F_t = F_1 \cdot F_2 = e(g^{\tau_k}, w)^{q_{A_t}(0)}$

where, $e(g^{\tau_k}, v_{t_j}^y) = e(g^{\tau_k}, \bar{v}_{t_i}^x) = 1$ because $g^{\tau_k} \in \mathbb{G}_{s'}$ and $v_{t_j}^y, \bar{v}_{t_i}^x \in \mathbb{G}_{n'}$. Next, the value of $T = e(g^{\tau_k}, w)^s$ is computed from $\{e(g^{\tau_k}, w)^{q_{A_i}(0)}\}_{A_i \in \mathcal{P}}$ and $\{e(g^{\tau_k}, w)^{q_{B_i}(0)}\}_{B_i \in \mathcal{P}}$ by using the recursive DecryptNode algorithm described in Bethencourt et al. [**?**]. Finally, the new ciphertext header $\widetilde{\mathcal{H}}_{\mathcal{P}} = (C, T)$ is returned.

- *Decrypt Phase 2*$(SK_{\mathcal{L}}, \widetilde{\mathcal{H}}_{\mathcal{P}}, C') \rightarrow M$

  On receiving the new ciphertext header $\widetilde{\mathcal{H}}_{\mathcal{P}} = (C, T) = (w^{\beta s}, e(g^{\tau_k}, w)^s)$, Now it will compute $D' = D \cdot \eta = g^{\frac{(\alpha+\tau_k)}{\beta}} g^{\frac{1}{\beta}} = g^{\frac{(\alpha+\tau_k)}{\beta}}$.
  Let $ek = \dfrac{e(C, D')}{T} = \dfrac{e(g^{\frac{(\alpha+\tau_k)}{\beta}}, w^{\beta s})}{e(g^{\tau_k}, w)^s} = e(g^{\alpha}, w)^s$.
  Finally, the plaintext message is computed by $M = C'/ek$.