A group generator $\mathcal{GG}$ is an efficient randomized algorithm that on input λ, outputs the description of a cyclic group $\mathcal{G}$ of prime order p along with a generator g for $\mathcal{G}$.

A pseudorandom family of functions $\{G(K,\cdot)\}_{K\in\{0,1\}^*}$ such that for each K of length λ, function $G(K,\cdot)$ takes λ-bit long inputs and outputs strings of length λ.

A pseudorandom family of functions $\{F(K,\cdot)\}_{K\in\{0,1\}^*}$ such that for each K of length λ, function $F(K,\cdot)$ takes λ-bit long inputs and outputs elements of $\mathbb{Z}_p$, for p prime of length $\Theta(\lambda)$.

A CPA secure private-key encryption scheme (Enc, Dec).

Algorithm 1 $\mathrm{Setup}(\lambda;\perp)\rightarrow(\mathrm{SK};\mathrm{EDB})$

Data Owner

1. Randomly select $(g,\mathcal{G})\leftarrow\mathcal{GG}(\lambda)$
2. Randomly select three master keys $K_1,K_2,K_3\xleftarrow{\$}\{0,1\}^\lambda$
3. Initialize $\mathrm{XSet},\mathrm{USet},\mathrm{ASet}\leftarrow\Phi$
4. $\mathrm{SK}\leftarrow(K_1,K_2,K_3)$
5. $\mathrm{EDB}\leftarrow(\mathrm{XSet},\mathrm{USet},\mathrm{ASet})$
6. Send EDB to the Server

Algorithm 2 $\mathrm{Add}(\mathrm{SK},\mathrm{DOC};\mathrm{EDB})\rightarrow(\perp;\mathrm{EDB}')$

Data Owner

1. Parse $\mathrm{SK}=(K_1,K_2,K_3)$
2. Initialize $\mathrm{TmpSet}\leftarrow\Phi$
3. For every document $d\in\mathrm{DOC}$
4. Set $K_d\leftarrow F(K_1,d)$, $\widetilde{K}_d\leftarrow F(K_2,d)$, $K_d^{enc}\leftarrow G(K_3,d)$
5. For every keyword $w\in KW(d)$
6. Set $X_{w,d}\leftarrow g^{F(\widetilde{K}_d,d)\cdot F(K_d,w)}$
7. Set $Y_{w,d}\leftarrow Enc(K_d^{enc},d)$

8. Update $\mathrm{TmpSet} \leftarrow \mathrm{TmpSet} \cup \{(X_{w,d}, Y_{w,d})\}$
9. Randomly permute the tuple-entries of TmpSet
10. Send TmpSet to the Server

Server

1. Parse $\mathrm{EDB} = (\mathrm{XSet}, \mathrm{USet}, \mathrm{ASet})$
2. Update $\mathrm{XSet}' \leftarrow \mathrm{XSet} \cup \mathrm{TmpSet}$
3. $\mathrm{EDB}' \leftarrow (\mathrm{XSet}', \mathrm{USet}, \mathrm{ASet})$

Algorithm 3 $\mathrm{Enroll}(\lambda; u) \rightarrow (\perp; \mathrm{UK}, \mathrm{UST})$

Data Owner

1. Randomly select two user's keys $K_u, \tilde{K}_u \xleftarrow{\$} \{0,1\}^{\lambda}$
2. Initialize $\mathrm{UsrAuth}, \mathrm{DocKey} \leftarrow \Phi$
3. $\mathrm{UK} \leftarrow \left(K_u, \tilde{K}_u\right)$
4. $\mathrm{UST} \leftarrow (\mathrm{UsrAuth}, \mathrm{DocKey})$
5. Send UK, UST to the Data User

Algorithm 4 $\mathrm{OnlineAuth}(\mathrm{SK}, \mathrm{DOC}; \mathrm{UK}, \mathrm{UST}; \mathrm{EDB}) \rightarrow (\perp; \mathrm{UST}'; \mathrm{EDB}')$

Data Owner

1. Parse $\mathrm{SK} = (K_1, K_2, K_3)$, $\mathrm{UK} = \left(K_u, \tilde{K}_u\right)$
2. Initialize $\mathrm{TmpArr} \leftarrow \Phi$, $\mathrm{TmpKey} \leftarrow \Phi$
3. For every document Data Owner authorize Data User to access $d \in \mathrm{DOC}$
4. Set $K_d \leftarrow F(K_1, d)$, $\tilde{K}_d \leftarrow F(K_2, d)$, $K_d^{enc} \leftarrow G(K_3, d)$
5. Set $uid_{u,d} \leftarrow F\left(\tilde{K}_u, d\right)$
6. Set $U_{u,d} \leftarrow F\left(\tilde{K}_d, d\right) \cdot (F(K_u, d))^{-1}$
7. Set $\mathrm{TmpArr}[uid_{u,d}] \leftarrow U_{u,d}$
8. Update $\mathrm{TmpKey} \leftarrow \mathrm{TmpKey} \cup \{(d, K_d, K_d^{enc})\}$
9. Send TmpKey to the Data User
10. Send TmpArr to the Server

Data User

1. If $\mathrm{UST} = \Phi$
2. Initialize $\mathrm{UsrAuth}, \mathrm{DocKey}, aid \leftarrow \Phi$
3. Parse $\mathrm{UST} = (\mathrm{UsrAuth}, \mathrm{DocKey}, aid)$
4. Update $\mathrm{DocKey}' \leftarrow \mathrm{DocKey} \cup \mathrm{TmpKey}$
5. $\mathrm{UST}' \leftarrow (\mathrm{UsrAuth}, \mathrm{DocKey}', aid)$

Server

1. Parse $\mathrm{EDB} = (\mathrm{XSet}, \mathrm{USet}, \mathrm{ASet})$
2. Update $\mathrm{USet}' \leftarrow \mathrm{USet} \cup \mathrm{TmpArr}$
3. $\mathrm{EDB}' \leftarrow (\mathrm{XSet}, \mathrm{USet}', \mathrm{ASet})$

Algorithm 5

$\mathrm{OfflineAuth}(\mathrm{UK_A}, \mathrm{UST_B}, \mathrm{DOC}; u_B, \mathrm{UST_B}; \mathrm{EDB}) \rightarrow (\perp; \mathrm{UST'_B}; \mathrm{EDB}')$

Data User A

1. Parse $\mathrm{UK_A} = \left(K_{u_A}, \tilde{K}_{u_A}\right)$, $\mathrm{UST_A} = (\mathrm{UsrAuth_A}, \mathrm{DocKey_A}, aid_A)$
2. Initialize $\mathrm{TmpUsrAuth} \leftarrow \Phi$, $\mathrm{TmpKey} \leftarrow \Phi$
3. Set $aid \leftarrow F\left(\tilde{K}_{u_A}, u_B\right)$
4. Set $\alpha \leftarrow F(K_{u_A}, u_B)^{-1}$
5. For every document Data User A authorize Data User B to access $d \in \mathrm{DOC}$
6. If $\mathrm{UsrAuth_A}[d] = \perp$ then
7. Set $uid \leftarrow F\left(\tilde{K}_{u_A}, d\right)$
8. Set $offtok \leftarrow g^{F(K_{u_A}, d) \cdot F(K_{u_A}, u_B)}$
9. Else if $(uid, offtok) \leftarrow \mathrm{UsrAuth_A}[d]$ then
10. Set $offtok \leftarrow offtok^{F(K_{u_A}, u_B)}$
11. Set $\mathrm{TmpUsrAuth}[d] \leftarrow (uid,\ offtok)$
12. Set $(d, K_d, K_d^{enc}) \leftarrow \mathrm{DocKey_A}$
13. Update $\mathrm{TmpKey} \leftarrow \mathrm{TmpKey} \cup \{(d, K_d, K_d^{enc})\}$
14. Send $\mathrm{TmpUsrAuth}$, TmpKey, aid to the Data User B
15. Send (aid, α), aid_A to the Server

Data User B

1. If $\mathrm{UST}_B = \Phi$
2. Initialize $\mathrm{UsrAuth}_B, \mathrm{DocKey}_B, aid_B \leftarrow \Phi$
3. Parse $\mathrm{UST}_B = (\mathrm{UsrAuth}_B, \mathrm{DocKey}_B, aid_B)$
4. Update $\mathrm{UsrAuth}'_{\mathrm{B}} \leftarrow \mathrm{UsrAuth}_{\mathrm{B}} \cup \mathrm{TmpUsrAuth}$
5. Update $\mathrm{DocKey}'_{\mathrm{B}} \leftarrow \mathrm{DocKey}_{\mathrm{B}} \cup \mathrm{TmpKey}$
6. Update $aid_B \leftarrow aid$
7. $\mathrm{UST}'_{\mathrm{B}} \leftarrow (\mathrm{UsrAuth}'_{\mathrm{B}}, \mathrm{DocKey}'_{\mathrm{B}}, aid_B)$

Server

1. Parse $\mathrm{EDB} = (\mathrm{XSet}, \mathrm{USet}, \mathrm{ASet})$
2. If $\mathrm{ASet}[aid] = \perp$
3. Initialize $dList \leftarrow \Phi$
4. Set $\mathrm{ASet}[aid] \leftarrow \alpha, dList$
5. If $aid_A \neq \perp$
6. Set $\alpha_A, dList_A \leftarrow \mathrm{ASet}[aid_A]$
7. Update $dList_A \leftarrow dList_A \cup \{aid\}$
8. Set $\mathrm{ASet}[aid_A] \leftarrow \alpha_A, dList_A$
9. Set $\alpha, dList \leftarrow \mathrm{ASet}[aid]$
10. Set $\alpha \leftarrow \alpha \cdot \alpha_A$
11. Set $\mathrm{ASet}[aid] \leftarrow \alpha, dList$
12. $\mathrm{EDB}' \leftarrow (\mathrm{XSet}, \mathrm{USet}, \mathrm{ASet})$

Algorithm 6 $\mathrm{Search}(w, \mathrm{UK}, \mathrm{UST}; \mathrm{EDB}) \rightarrow (\mathrm{Res}; \mathrm{EDB}')$

Data User

1. Parse $\mathrm{UK} = \left(K_u, \tilde{K}_u\right)$, $\mathrm{UST} = (\mathrm{UsrAuth}, \mathrm{DocKey})$
2. Initialize $\mathrm{Query} \leftarrow \Phi$, $\mathrm{Token} \leftarrow \Phi$
3. For every document key information $(d, K_d, K_d^{enc}) \in \mathrm{DocKey}$
4. If $\mathrm{UsrAuth}[d] = \perp$ then
5. Set $uid_{u,d} \leftarrow F\left(\tilde{K}_u, d\right)$

6. Set $stk_d \leftarrow g^{F(K_d,w)\cdot F(K_u,d)}$
7. Else if $(uid, offtok) \leftarrow \mathrm{UsrAuth}[d]$ then
8. Set $stk_d \leftarrow offtok^{F(K_d,w)}$
9. Update $\mathrm{Token} \leftarrow \mathrm{Token} \cup \{(uid, stk_d)\}$
10. Randomly permute the tuple-entries of Token
11. **Set** $\mathrm{Query} \leftarrow (\mathrm{Token}, aid)$
12. Send Query to the Server

Server

1. Parse $\mathrm{EDB} = (\mathrm{XSet}, \mathrm{USet}, \mathrm{ASet})$, $\mathrm{Query} = (\mathrm{Token}, aid)$
2. Initialize $\mathrm{TmpRes} \leftarrow \Phi$
3. For every query tuple $(uid, stk_d) \in \mathrm{Token}$
4. Set $x \leftarrow stk_d^{\mathrm{USet}[uid]}$
5. If $aid \neq \perp$ then
6. Set $x \leftarrow x^{\mathrm{ASet}[aid]}$
7. If $(x, Y) \in \mathrm{XSet}$ then $\mathrm{TmpRes} \leftarrow \mathrm{TmpRes} \cup \{\mathrm{Y}\}$
8. $\mathrm{EDB}' \leftarrow \mathrm{EDB}$
9. Send TmpRes to the Data User

Data User: Final Output

1. Initialize $\mathrm{Res} \leftarrow \Phi$
2. For every encrypted document identifier result $y_d \in \mathrm{TmpRes}$
3. Recover $d \leftarrow Dec(K_d^{enc}, y_d)$
4. Update $\mathrm{Res} \leftarrow \mathrm{Res} \cup \{d\}$
5. **Output** Res