Anonymous Credentials with efficient revocation based on accumulators and Bilinear Mappings

* First, Issuing Authority (IA) runs the initialization steps to generate the Bilinear Mapping (BM), the accumulator, and its secret and public keys.

* IssuerKeygen(1^k, n):
  - 1^k :- The security parameter used to generate the parameters of the BM. (private)
  - n :- Maximum number of users the accumulator can accumulate. (public)

i) RunBMGen(1^k) to generate the parameters of the BM:
  - $params_{BM} = (q, G, G_T, e, g)$ of a symmetric bilinear map $e: G \times G \rightarrow G_T$ of order q and generator g. (public)

ii) Pick additional bases $h, h_0, \ldots, h_{\ell+1}, \tilde{h}, u \in G$ and $x, sk, \gamma \in \mathbb{Z}_q$ and compute $y = h^x$ and $pk = g^{sk}$
  - h → Element of G used in the signature scheme (public)
  - $h_0, h_1, \ldots, h_{\ell+1}$ → Elements of G used to encode the messages being signed (attributes of the certificate). ℓ messages can be encoded. (public)
  - u → Element of G used to create a part of the witness (public)
  - $\tilde{h}$ → Element of G used in ZKPOK of user authentication (public)
  - x → $x \in \mathbb{Z}_q$ is the secret key used by the I.A. to sign the certificate (private)
  - sk → $sk \in \mathbb{Z}_q$ → Secret key used to sign the witness of the user. (private)
  - y → $y = h^x$ is the public key used to ~~verg~~ verify signature on the certificate (public)
  - pk → $pk = g^{sk}$ → Public key used to verify witness signature (public)

iii) Compute $g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}$, where $g_i = g^{\gamma^i}$, and $z = e(g, g)^{\gamma^{n+1}}$
  - i → $i \in [1, n]$. Id of user i. (private)
  - g_i → $g_i = g^{\gamma^i}$. Value accumulated in the accumulator (public)
  - γ → γ Random element $\in \mathbb{Z}_q$ used to generate g_i's (private)
  - z → $z = e(g, g)^{\gamma^{n+1}}$. Public value used in witness verification (public)

P.T.O.

* Output :-

- $(sk_I, pk_I) = ((x, sk, \gamma), (params_{BM}, y, h, h_0, \ldots, h_{l+1}, \tilde{h}, u, pk, z))$
- $epoch_\phi = (acc_\phi = 1, \phi)$
  - → We update our accumulator and witnesses after every 'epoch' time
  - → $\underline{acc_V}$ = value of accumulator when {V} indices are valid
  - → $\underline{V}$ = Set of all currently valid users. (Initially ϕ).
- $state_\phi = (\phi, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n})$
  - → $State_V = (V, \{g_{1 \ldots 2n}\} / \{g_{n+1}\})$
  - → $\underline{State_U} = (U, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n})$
  - → $\underline{U}$ = Set ~~off~~ of all indices ever validated by IA. $V \subseteq U$.

→ Obtain Cert $(sk_I, pk_I, \{m_j\}_{j \in \{1..l\}}, epoch_V, state_U)$

- sk_I, pk_I → Secret and ~~private~~ public key of the issuing authority
- $\{m_j\}_{j \in \{1..l\}}$ → The messages/attributes the user wants signed.
- $epoch_V = (acc_V, V)$, where V is current set of authenticated users.
- $State_U = (U, g_1, \ldots g_n, g_{n+2}, \ldots, g_{2n})$, ~~U = all users~~

* i) Update the accumulator

→ $epoch_{V \cup \{i\}} = (acc_{V \cup \{i\}} = acc_V \cdot g_{n+1-i}, V \cup \{i\})$ (public)

→ $State_{U \cup \{i\}} = (U \cup \{i\}, g_1, \ldots, g_n, g_{n+1}, \ldots, g_{2n})$ (public)

ii) Generate certificate

- The issuer chooses random $c, s \in \mathbb{Z}_q^*$ and computes the signature as

$$\sigma = ((\Pi_{j \in \{1..l\}} h_j^{m_j}) g_i h_{l+1}^s)^{\frac{1}{x+c}}$$

- The user recieves

$\underline{cert_i} = (\sigma, c, m_1, \ldots, m_l, g_i, s, i)$ (private)

iii) Generate witness

- The issuer computes

$$\underline{w} = \Pi_{j \in V}^{j \neq i} g_{n+1-j+i}$$

- The issuer also computes σ_i and u_i as follows

$\underline{\sigma_i} = g^{1/(sk + \gamma^i)}$ $\underline{u_i} = u^{\gamma^i}$

- The issuer outputs ~~#~~ to the user

$\underline{wit_i} = (\sigma_i, u_i, g_i, w, V \cup \{i\})$ (private)

* Output: $epoch_{V \cup \{i\}}$, $state_{U \cup \{i\}}$, wit_i

→ Update Epoch $(V, state_U)$

* Checks whether $V \subseteq U$ and outputs
  $epoch_V = \{acc_V = \prod_{j \in V} g_{n+1-j}, V\}$

→ Update Witness $(wit_i, epoch_{V_w}, epoch_V, state_U)$

- $wit_i = (\sigma_i, u_i, g_i, w, V_w)$
- $V_w \to$ old valid users.
- $V \to$ new valid users.

* (i) Checks whether $V \subseteq U$ (aborts otherwise)

(ii) Updates w as follows

#
$$w = w \frac{\prod_{(j \in V \setminus V_w)} g_{n+1-j+i}}{\prod_{j \in V_w \setminus V} g_{n+1-j+i}}$$

* Output:
  $wit_i = (\sigma_i, u_i, g_i, w, V)$

→ Verifying User $(epoch_V, state_U)$

* ~~If $i \in V$, then the~~ The below protocol allows a user to prove possession of an unrevoked (and updated) credential $cred_i = (\sigma, c, m_i, g_i, s_i)$ using $wit_i = (\sigma_i, g_i, u_i, w, V_w)$

(Without preserving anonimity, the ~~se~~ user can be verified by computing
$$\frac{e(g_i, acc_V)}{e(g, w)} = \frac{e(g_i, \prod_{j \in V} g_{n+1-j})}{e(g, \prod_{j \in V}^{j \neq i} g_{n+1-j+i})} = \frac{e(g,g)^{\sum_{j \in V} \gamma^{n+1-j+i}}}{e(g,g)^{\sum_{j \neq i} \gamma^{n+1-j+i}}} = e(g,g)^{\gamma^{n+1}} = z$$
).

* (i) The user (as prover) picks $\rho, \rho', r, r', r'', r''' \in \mathbb{Z}_q$

(ii) User picks open, open' $\in \mathbb{Z}_q$ to commit to ρ and r respectively. Then the user computes and sends the following commitments to the verifier

$$C = h^\rho \tilde{h}^{open} \qquad D = g^r \tilde{h}^{open'}$$

iii) The user computes the following randomizations and sends the blinded values to the verifier

$$A = \sigma \tilde{h}^\rho \quad G = g_i \tilde{h}^r \quad W = w \tilde{h}^{r'} \quad S = \sigma_i \tilde{h}^{r''} \quad U = u_i \tilde{h}^{r'''}$$

Note:- While engaging in the ZKPOK, ~~the~~ for the terms requiring exponentiation by the random values, the verifier requests the prover for the result along with the proof of knowledge of its discrete logarithm.

P.T.O.

iv) The user now engages in the following ZKPOK with the verifier

$\nearrow$ Secret values

$$PK\{(c, \rho, open, mult, tmp, m_1, \dots, m_\ell, s, r, open', mult', tmp', r', r'', r'''):$$

$$C = h^{\rho} \tilde{h}^{open} \quad \wedge \qquad \text{— ①}$$

$$1 = C^{c} h^{-mult} \tilde{h}^{-tmp} \quad \wedge \qquad \text{— ②}$$

$$\frac{e(h_0 G, h)}{e(A, y)} = e(A, h)^{c} \cdot e(\tilde{h}, h)^{r} \cdot e(\tilde{h}, y)^{-\rho} \cdot e(\tilde{h}, h)^{-mult} \cdot \prod_{j=1}^{\ell} e(h_j, h)^{-m_j} \cdot e(h_{\ell+1}, h)^{-s} \quad \wedge \qquad \text{— ③}$$

$$\frac{e(G, acc_V)}{e(g, W) z} = e(\tilde{h}, acc_V)^{r} e(1/g, \tilde{h})^{r'} \quad \wedge \qquad \text{— ④}$$

$$D = g^{r} \tilde{h}^{open'} \quad \wedge \qquad \text{— ⑤}$$

$$1 = D^{c} g^{-mult'} \tilde{h}^{-tmp'} \quad \wedge \qquad \text{— ⑥}$$

$$\frac{e(pk \cdot G, s)}{e(g, g)} = e(pk\, G, \tilde{h})^{r'} e(\tilde{h}, \tilde{h})^{-mult'} e(\tilde{h}, s)^{r} \quad \wedge \qquad \text{— ⑦}$$

$$\frac{e(G, u)}{e(g, U)} = e(\tilde{h}, u)^{r} e(g, \tilde{h})^{-r''} \qquad \text{— ⑧}$$

* Proof

- ① and ⑤ correspond to the committed values.
- From ② and ⑥, we get that
  $mult = \rho c$, $tmp = open\, c$, $mult' = rc$, $tmp' = open'\, c$
  ~~not finding~~ (Since one can compute $\log_g h$ otherwise, breaking our hardness assumptions)
- ④ Simplifies to the following, verifying the user's ~~credentials~~ witness

$$\frac{e(G \tilde{h}^{-r''}, acc_V)}{e(g, W \tilde{h}^{-r'})\, z} = \frac{e(g_i, acc_V)}{e(g, w)\, e(g, g)^{\gamma^{n+1}}} = \frac{e(g_i, \prod_{j \in V} g_{n+1-j})}{e(g, \prod_{j \in V}^{j \neq i} g_{n+1-j+i})\, z} = 1 \quad ✓$$

- In ③, we assert the user's knowledge of values $m_1, \dots, m_\ell$. Simplifying it gives

$$\frac{e(h_0 g_i \tilde{K}, h)}{e(\sigma \tilde{K}, y)} = e(\sigma^{c} \tilde{K}^{\rho c}, h) \cdot \cancel{e(\tilde{K}^{r}, h)} \cdot \cancel{e(\tilde{K}^{-\rho}, y)} \cdot \cancel{e(\tilde{K}^{-\rho c}, h)} \cdot e(\prod h_j^{-m_j}, h) \cdot e(h_{\ell+1}^{-s}, h)$$

$$\Rightarrow e(h_0 \prod_{i=1}^{\ell} h_j^{m_j} g_i h_{\ell+1}^{s}, h) = e(\sigma^{c}, h) \cdot e(\sigma, h)^{x}$$

$$\Rightarrow e(\sigma^{x+c}, h) = e(\sigma^{c+x}, h)$$

H.P.

• ⑦ simplifies as shown below

$$\frac{e(pk \cdot g_i \tilde{K}^{r}, \sigma_i \tilde{K}^{r'})}{e(g,g)} = \cancel{e(pk\, g_i \tilde{h}^{r}, \tilde{h}^{r'})} \cdot e(\tilde{h},\tilde{h})^{-rc} \cdot \cancel{e(\tilde{h}^{r}, \sigma_i \tilde{h}^{r'})}$$

$\Rightarrow \cancel{e(pk\, g_i,}$

$$\Rightarrow \frac{e(g^{sk} \cdot g_i, g^{\frac{1}{sk+y_i}})}{e(g,g)} = e(\tilde{h},\tilde{h})^{-rc}$$

$$\Rightarrow \frac{e(g,g)}{e(g,g)} = e(\tilde{h},\tilde{h})^{-rc}$$

• Finally, ⑧ simplifies as

$$\frac{e(\tilde{G},u)}{e(g,u)} = e(\tilde{h},u)^{r} e(1/g,\tilde{h})^{r'''}$$

$$\Rightarrow \frac{e(g_i \tilde{K}^{r}, u)}{e(g, u_i \tilde{K}^{r'''})} = \cancel{e(\tilde{h}^{r}, u)}\ \cancel{e(g, \tilde{h}^{-r''})}$$

$$\Rightarrow \frac{e(g_i, u)}{e(g, \underset{u^{y_i}}{u_i})_i} = 1$$

H.P.

* Hence the user can authenticate himself without revealing any information about themselves.
* The above scheme also allows for efficient ~~revocation and~~ credentials revocation.