$\xi^{q_0+q_1}(1-\xi)^2$. If we let $\theta = (q_0+q_1)/(q_0+q_1+2)$ be the probability for getting $s_i = 0$ in the queries of $\mathcal{H}_0$ and $\mathcal{H}_1$, then the entire probability not aborting is $\xi^{q_0+q_1}(1-\xi)^2 \leq 4/(e^2(q_0+q_1+2)^2)$. If the abortion fails, the probability of outputting the correct $\mathcal{Z}$ is $2\epsilon/q_{\mathcal{H}_2}$. Thus, the probability of solving CBDH problem is $8\epsilon/(e^2 q_{\mathcal{H}_2}(q_0+q_1+2)^2)$.

V. Concrete Construction of IB-BME

A. Identity-based broadcast matchmaking encryption

- **Setup**(λ): With the input security parameter λ, it first picks and sets a bilinear group $\mathcal{BG} = (\mathbb{G}_0, \mathbb{G}_1, \mathbb{G}_T, p, e)$, where the bilinear map $e : \mathbb{G}_0 \times \mathbb{G}_1 \to \mathbb{G}_T$ holds and p is the prime order of groups $(\mathbb{G}_0, \mathbb{G}_1)$. Next, it randomly picks a generator $g \in \mathbb{G}_0$, generators $h, u, v, w \in \mathbb{G}_1$, $\alpha, \beta, \rho \in \mathbb{Z}_p$ and calculates $g_1 = g^\rho, h_0 = h^\rho, h_1 = h^\beta$. Then, it selects the following collision-resistant hash functions $\mathcal{H}_0 : \{0,1\}^* \to \mathbb{G}_0$, $\mathcal{H}_1 : \{0,1\}^* \to \mathbb{G}_1$, $\mathcal{H}_2 : \mathbb{G}_T \to \mathbb{Z}_p$, $\mathcal{H}_3 : \mathbb{Z}_p^2 \times \mathbb{G}_0 \times \mathbb{G}_1^2 \to \{0,1\}^\ell$, $\mathcal{H}_4 : \mathbb{G}_0 \times \mathbb{G}_1^2 \times \{0,1\}^\ell \times \mathbb{Z}_p^{2t} \to \mathbb{Z}_p$. Finally, it publishes the public parameter $\mathsf{pp} = (\mathcal{BG}, g, g_1, u, v, w, h, h_0, h_1, \{\mathcal{H}_i\}_{i\in[0,4]})$ and stores the master secret key $\mathsf{msk} = (\rho, \alpha)$.
- **EKGen**(msk, id^*): Based on msk and identity id^*, it produces an encryption key $\mathsf{ek}_{\mathsf{id}^*} = \mathcal{H}_1(\mathsf{id}^*)^\alpha$.
- **DKGen**(msk, id): With the input msk and identity id, it returns a decryption key $\mathsf{dk}_{\mathsf{id}} = (\mathsf{dk}_1, \mathsf{dk}_2, \mathsf{dk}_3)$, where $\mathsf{dk}_1 = \mathcal{H}_0(\mathrm{id})^\rho, \mathsf{dk}_2 = \mathcal{H}_0(\mathsf{id})^\alpha, \mathsf{dk}_3 = \mathcal{H}_0(\mathsf{id})$.
- **Enc**(pp, $\mathcal{S}$, $\mathsf{ek}_{\mathsf{id}^*}$, m): Given pp, a target identity set $\mathcal{S}$ with its length t, an encryption key identity $\mathsf{ek}_{\mathsf{id}^*}$ and the plaintext $m \in \{0,1\}^{\ell_1}$, it first picks $s, d_1, d_2, \sigma, \tau \in \mathbb{Z}_p$ and computes $C_0 = h^s$, $C_1 = g^s$, $C_2 = h_1^\tau$. For each $\mathsf{id}_i \in \mathcal{S}$, it sets $\mathsf{U}_{\mathsf{id}_i} = \mathcal{H}_2(e(h_0, \mathcal{H}_0(\mathsf{id}_i))^s)$ and $\mathsf{V}(\mathsf{id}_i) = \mathcal{H}_2(e(\mathcal{H}_0(\mathsf{id}_i), ek_{\mathsf{id}^*} \cdot h_1^\tau))$, $f(x) = \prod_{i=1}^t (x - \mathsf{U}_{\mathsf{id}_i}) + d_1 = \sum_{i=0}^{t-1} a_j x^j + x^t \mod p$ and $g(y) = \prod_{k=1}^t (y - \mathsf{V}(\mathsf{id}_i)) + d_2 = \sum_{k=0}^{t-1} b_k y^k + y^t \mod p$, where $a_0, \ldots, a_{t-1}$ and $b_0, \ldots, b_{t-1}$ are the coefficients correspond to x^j and y^k. Next, it sets $C_3 = [\mathcal{H}_3(d_1, d_2, C_1, C_0, C_2)]_{\ell-\ell_1} || (\mathcal{H}_3(d_1, d_2, C_1, C_0, C_2)]^{\ell_1} \oplus m)$, $\varphi = \mathcal{H}_4(C_1, C_0, C_2, C_3, a_0, \ldots, a_{t-1}, b_0, \ldots, b_{t-1})$ and $C_4 = (u^\varphi v^\sigma w)^s$. Finally, it generates a ciphertext $\mathsf{ct} = (\sigma, C_1, C_0, C_2, C_3, C_4, a_0, \ldots, a_{t-1}, b_0, \ldots, b_{t-1})$.
- **Dec**(pp, $\mathsf{dk}_{\mathrm{id}_i}$, id^*, ct): Based on the public parameter pp, a decryption key $\mathsf{dk}_{\mathrm{id}_i}$, the target identity id^* and the ciphertext $\mathsf{ct} = (\sigma, C_1, C_0, C_2, C_3, C_4, a_0, \ldots, a_{t-1}, b_0, \ldots, b_{t-1})$, it first computes $\varphi = \mathcal{H}_4(C_1, C_0, C_2, C_3, a_0, \ldots, a_{t-1}, b_0, \ldots, b_{t-1})$ and then determines whether $e(C_1, u^\varphi v^\sigma w) = e(g, C_4)$ holds. If not, it returns $\perp$. Otherwise, it computes $\mathsf{U}_{\mathsf{id}_i} = \mathcal{H}_2(e(C_0, \mathsf{dk}_{i,1})) = \mathcal{H}_2(e(C_0, \mathcal{H}_0(\mathsf{id}_i)^\rho))$, $d_1 = f(\mathsf{U}_{\mathsf{id}_i}) = \sum_{j=0}^{t-1} a_j (\mathsf{U}_{\mathsf{id}_i})^j + (\mathsf{U}_{\mathsf{id}_i})^t \mod p$ and $\mathsf{V}(\mathsf{id}_i) = \mathcal{H}_2(e(\mathsf{dk}_{i,3}, C_2) e(\mathsf{dk}_{i,2}, \mathcal{H}_1(\mathsf{id}^*))) = \mathcal{H}_2(e(\mathcal{H}_0(\mathsf{id}_i), ek_{\mathsf{id}^*} \cdot h_1^\tau))$, $d_2 = g(\mathsf{V}_{\mathsf{id}_i}) = \sum_{i=0}^{t-1} b_j (\mathsf{V}_{\mathsf{id}_i})^j + (\mathsf{V}_{\mathsf{id}_i})^t \mod p$. If $[C_3]_{\ell-\ell_1} \neq [\mathcal{H}_3(d_1, d_2, C_1, C_0, C_2)]_{\ell-\ell_1}$, it returns $\perp$. Otherwise, it outputs $m = [\mathcal{H}_3(d_1, d_2, C_1, C_0, C_2)]^{\ell_1} \oplus [C_3]^{\ell_1}$.

B. Adaptive identity-based broadcast matchmaking encryption

- **Setup**(λ, ℓ): With the input security parameter λ and the maximum legitimate identity set ℓ, it first picks a bilinear group $\mathcal{BG} = (\mathbb{G}_0, \mathbb{G}_1, \mathbb{G}_T, p, e)$ with three random generators $g, v \in \mathbb{G}_0$ and $h \in \mathbb{G}_1$. Next, it chooses random $(\ell+1)$-dimensional vectors from $\mathbb{Z}_p$ with $\vec{r_1} = (r_{1,0}, \ldots, r_{1,\ell})$ and $\vec{r_2} = (r_{2,0}, \ldots, r_{2,\ell})$. It also picks $t_1, t_2, \beta_1, \beta_2, \alpha, \rho \in \mathbb{Z}_p$, $b, \tau \in \mathbb{Z}_p^*$, sets $\vec{r} = \vec{r_1} + b\vec{r_2} = (r_0, \ldots, r_\ell)$, $t = t_1 + bt_2$, $\beta = \beta_1 + b\beta_2$ and calculates $R = g^{\vec{r}} = (g^{r_0}, \ldots, g^{r_\ell})$, $T = g^t$, $e(g,h)^\beta$. Then, it selects the following hash functions $\mathcal{H}_0 : \{0,1\}^* \to \mathbb{G}_1$, $\mathcal{H}_1 : \{0,1\}^* \to \mathbb{G}_0$, $\mathcal{H}_2 : \{0,1\}^* \to \mathbb{Z}_p$, $\mathcal{H}_3 : \mathbb{G}_T \to \mathbb{Z}_p$. Finally, it publishes the public parameter $\mathsf{pp} = (\mathcal{BG}, v, v^\rho, g, g^b, R, T, e(g,h)^\beta, h, h^{\vec{r_1}}, h^{\vec{r_2}}, h^{t_1}, h^{t_2}, g^{\tau\beta}, h^{\tau\beta_1}, h^{\tau\beta_2}, h^{1/\tau}, \{\mathcal{H}_i\}_{i\in[0,3]})$ and stores the master secret key $\mathsf{msk} = (h^{\beta_1}, h^{\beta_2}, \alpha, \rho)$.
- **EKGen**(msk, id^*): Based on msk and identity id^*, it produces an encryption key $\mathsf{ek}_{\mathsf{id}^*} = \mathcal{H}_1(\mathsf{id}^*)^\alpha$.
- **DKGen**(msk, id): With the input msk and identity id, it first selects $z \in \mathbb{Z}_p$, random tags $\mathsf{rtag}_1, \ldots, \mathsf{rtag}_\ell$ and returns a decryption key $\mathsf{dk}_{\mathsf{id}} = (\mathsf{dk}_1, \mathsf{dk}_2, \mathsf{dk}_3, \mathsf{dk}_4, \mathsf{dk}_5, \mathsf{dk}_6, \{\mathsf{dk}_{7,j}, \mathsf{dk}_{8,j}, \mathsf{rtag}_j\}_{j=1}^\ell)$, where $\mathsf{dk}_1 = \mathcal{H}_0(\mathrm{id})^\rho, \mathsf{dk}_2 = \mathcal{H}_0(\mathsf{id})^\alpha$, $\mathsf{dk}_3 = \mathcal{H}_0(\mathsf{id})$, $\mathsf{dk}_4 = h^{\beta_1}(h^{t_1})^z$, $\mathsf{dk}_5 = h^{\beta_2}(h^{t_2})^z$, $\mathsf{dk}_6 = h^z$, $\mathsf{dk}_{7,j} = (h^{t_1})^{\mathsf{rtag}_j} h^{r_{1,j}} / (h^{r_{1,0}})^{(\mathcal{H}_2(\mathsf{id}))^j}$, $\mathsf{dk}_{8,j} = (h^{t_2})^{\mathsf{rtag}_j} h^{r_{2,j}} / (h^{r_{2,0}})^{(\mathcal{H}_2(\mathsf{id}))^j}$.
- **Enc**(pp, $\mathcal{S}$, $\mathsf{ek}_{\mathsf{id}^*}$, m): Given pp, a target identity set $\mathcal{S}$ with its length $n \leq \ell$, an encryption key identity $\mathsf{ek}_{\mathsf{id}^*}$ and the plaintext m, it first defines an identity vector $\vec{y} = (y_0, \ldots, y_n, \ldots, y_\ell)$, where y_i is the coefficients from $f(x) = \prod_{\mathsf{id}_j \in \mathcal{S}} (x - \mathcal{H}_2(\mathsf{id}_j)) = \sum_{i=0}^n y_i x^i$. Here please note that if $n < \ell$, $y_{n+1} = \ldots = y_\ell = 0$. It next picks $s, d_2, \mathsf{ctag} \in \mathbb{Z}_p$ and computes $C_0 = m \cdot e(g,h)^{\beta s}$, $C_1 = g^s$, $C_2 = g^{bs}$, $C_3 = (T^{\mathsf{ctag}} \prod_{i=0}^n (g^{r_i})^{y_i})^{d_2 s}$, $C_4 = v^s$. For each $\mathsf{id}_i \in \mathcal{S}$, it sets $\mathsf{V}(\mathsf{id}_i) = \mathcal{H}_3(e(\mathcal{H}_0(\mathsf{id}_i), ek_{\mathsf{id}^*} \cdot g^{bs} \cdot v^{\rho s}))$, $g(y) = \prod_{k=1}^n (y - \mathsf{V}(\mathsf{id}_i)) + d_2 = \sum_{i=0}^n b_k y^k + y^t \mod p$, where $b_0, \ldots, b_n, \ldots, b_\ell$ are the coefficients correspond to y^k. Finally, it generates a ciphertext $\mathsf{ct} = (C_0, C_1, C_2, C_3, C_4, \mathsf{ctag}, b_0, \ldots, b_n)$.
- **Dec**(pp, $\mathsf{dk}_{\mathrm{id}_i}$, id^*, ct): Based on the public parameter pp, a decryption key $\mathsf{dk}_{\mathrm{id}_i}$, the target identity id^* and the ciphertext $\mathsf{ct} = (C_1, C_2, C_3, b_0, \ldots, b_n)$, it first computes $\mathsf{V}(\mathsf{id}_i) = \mathcal{H}_3(e(\mathsf{dk}_{i,3}, C_2) e(\mathsf{dk}_{i,2}, \mathcal{H}_1(\mathsf{id}^*)) e(\mathsf{dk}_{i,1}, C_4)) = \mathcal{H}_3(e(\mathcal{H}_0(\mathsf{id}_i), ek_{\mathsf{id}^*} \cdot g^{bs} \cdot v^{\rho s}))$, $d_2 = g(\mathsf{V}_{\mathsf{id}_i}) = \sum_{j=0}^n b_j (\mathsf{V}_{\mathsf{id}_i})^j + (\mathsf{V}_{\mathsf{id}_i})^j \mod p$. It next calculates $\mathsf{rtag} = \sum_{i=1}^\ell y_i \mathsf{rtag}_i$, if $\mathsf{rtag} = \mathsf{ctag}$, it aborts and outputs $\perp$; otherwise, it computes $\mathsf{A} = (e(C_1, \prod_{j=1}^m \mathsf{dk}_{7,j}^{y_j}) e(C_2, \prod_{j=1}^m \mathsf{dk}_{8,j}^{y_j}) / (C_3^{1/d_2}, \mathsf{dk}_6))$, $\mathsf{B} = e(C_1, \mathsf{dk}_4) \cdot e(C_2, \mathsf{dk}_5)$ and recovers $m = \mathsf{A}^{1/(\mathsf{rtag}-\mathsf{ctag})} \cdot \mathsf{B}^{-1}$.

C. Security Proofs of IB-BME

***Theorem** 6:* Assume that ADDH and DDH assumptions hold, then our IB-BME realizes adaptively security.