We give a complete security model and security analysis.

I. Security Model

In the CLHS scheme, there are two types of adversaries $\mathcal{A}_I$ and $\mathcal{A}_{II}$. A Type-I adversary $\mathcal{A}_I$ can replace the public key PK_{ID} of the user ID to launch attacks, but it is not given the master key msk or the partial private key k_{ID} of the user ID. A Type-II adversary $\mathcal{A}_{II}$ is a malicious-but-passive KGC proposed by Au et al. [1], as the KGC may not generate a master public/private key pair honestly to launch attacks. However, $\mathcal{A}_{II}$ cannot replace the user's public key or obtain the user's secret value.

We use two games between the adversaries and challengers to specify the formal security model of the CLHS scheme. There are five oracles which can be accessed by the adversaries in two games.

- **Create-User:** On input a query $ID \in \{0,1\}^*$, nothing is to be carried if ID has already been created. Otherwise, this oracle runs the algorithms *Partial-Private-Key-Query*, *Secret-Value-Query*, and *Set-Public-Key* to obtain the partial private key d_{ID}, the secret value x_{ID}, and the public key PK_{ID}. Then it adds $(ID, d_{ID}, x_{ID}, PK_{ID})$ to the list L_{user}. In this case, ID is said to be created. In both cases, PK_{ID} is returned.
- **Replace-Public-Key:** On input a query (ID, PK'_{ID}), where ID and PK'_{ID} denote the user identity and a public key value, this oracles replaces the user ID's public key with PK'_{ID} and updates the corresponding information in the list L_{user} if ID has been created. Otherwise, a symbol $\perp$ is returned.
- **Partial-Private-Key-Query:** On input a query $ID \in \{0,1\}^*$, this oracles returns d_{ID} if ID has been created. Otherwise, a symbol $\perp$ is returned. Note that this oracle does not return the partial private key associated with the replaced public key PK'_{ID}.
- **Secret-Value-Query:** On input a query $ID \in \{0,1\}^*$, this oracles returns x_{ID} if ID has been created. Otherwise, a symbol $\perp$ is returned. Note that this oracle does not return the secret value associated with the replaced public key PK'_{ID}.
- **Sign-Query:** On input the j-th query (ID, V_j), where V_j is described by properly augmented basis vectors $\mathbf{v}_{j1}, \cdots, \mathbf{v}_{jm}$, if ID has been created, this oracles reutrns an identifier id_j and the signatures $\sigma_{j1}, \cdots, \sigma_{jm}$ such that *Verify*$(params, PK_{ID}, id, \mathbf{v}_{ji}, \sigma_{ji}) = 1$ for all $i \in [1, m]$. Otherwise, a symbol $\perp$ is returned. Note that PK_{ID} is the public key returned from the oracle **Create-User**.

The unforgeability of the CLHS scheme against adaptive chosen identity and adaptive chosen subspace adversary $\mathcal{A}_I$ is defined by the following game.

Game-I: Let $\mathcal{B}_1$ be the game simulator, it interacts with $\mathcal{A}_I$.

Setup: $\mathcal{B}_1$ runs the algorithm *Setup* to obtain msk and returns the public parameters $params$ to $\mathcal{A}_I$.

Queries: $\mathcal{A}_I$ can adaptively make queries onto **Create-User**, **Replace-Public-Key**, **Partial-Private-Key-Query**, **Secret-Value-Query**, and **Sign-Query**.

Forgery: After all queries, $\mathcal{A}_I$ outputs an identity ID^*, the public key PK_{ID^*}, an identifier id^*, a non-zero vector $\mathbf{v}^*$, and a signature σ^*.

$\mathcal{A}_I$ wins the above **Game-I** if $\mathcal{A}_I$ has never submitted ID^* to the oracle **Partial-Private-Key-Query**, *Verify*$(params, PK_{ID^*}, id^*, \mathbf{v}^*, \sigma^*) = 1$, and one of the following conditions is satisfied.

- Type 1 Forgery: (ID^*, id^*) never appears in any **Sign-Query** query.
- Type 2 Forgery: $ID^* = ID_j$ and $id^* = id_j$ for some j, i.e., (ID^*, id^*) appears in some **Sign-Query** query, but $\mathbf{v}^* \notin V_j$.

Define the advantage $Adv^{csa,cida}_{\mathcal{A}_I,CLHS}(\lambda)$ as the probability of $\mathcal{A}_I$ winning the above game.

The unforgeability of the CLHS scheme against adaptive chosen identity and adaptive chosen subspace adversary $\mathcal{A}_{II}$ is defined by the following game.

Game-II: Let $\mathcal{B}_2$ be the game simulator, it interacts with $\mathcal{A}_{II}$.

Setup: $\mathcal{A}_{II}$ returns the public parameters $params$, but it is not allowed to query any oracle in this phase[1].

Queries: $\mathcal{A}_{II}$ can adaptively make queries onto **Create-User**, **Replace-Public-Key**, **Secret-Value-Query**, and **Sign-Query**. However, the **Create-User** oracle is changed as follows.

Create-User: On input a user identity ID and the corresponding partial private key d_{ID}, nothing is to be carried if ID has already been created. Otherwise, this oracle runs the algorithms *Secret-Value-Query* and *Set-Public-Key* to obtain the secret value x_{ID} and the public key PK_{ID}. Then it adds $(ID, d_{ID}, x_{ID}, PK_{ID})$ to the list L_{user}. In this case, ID is said to be created. In both cases, PK_{ID} is returned.

Forgery: After all queries, $\mathcal{A}_{II}$ outputs an identity ID^*, the public key PK_{ID^*}, an identifier id^*, a non-zero vector $\mathbf{v}^*$, and a signature σ^*.

$\mathcal{A}_{II}$ wins the above **Game-II** if $\mathcal{A}_{II}$ neither submitted ID^* to the oracle **Secret-Value-Query** nor queried **Replace-Public-Key** to replace the public key of ID^*, *Verify*$(params, PK_{ID^*}, id^*, \mathbf{v}^*, \sigma^*) = 1$, and one of the following conditions is satisfied.

- Type 1 Forgery: (ID^*, id^*) never appears in any **Sign-Query** query.
- Type 2 Forgery: $ID^* = ID_j$ and $id^* = id_j$ for some j, i.e., (ID^*, id^*) appears in some **Sign-Query** query, but $\mathbf{v}^* \notin V_j$.

Define the advantage $Adv^{csa,cida}_{\mathcal{A}_{II},CLHS}(\lambda)$ as the probability of $\mathcal{A}_{II}$ winning the above game.

Definition 1. *The CLHS scheme is unforgeable under against adaptive chosen identity and adaptive chosen subspace attack if both* $Adv^{csa,cida}_{\mathcal{A}_{II},CLHS}(\lambda)$ *and* $Adv^{csa,cida}_{\mathcal{A}_I,CLHS}(\lambda)$ *are negligible.*

II. Security Analysis

Theorem 1. *Our CHNCS scheme is unforgeable against adaptive chosen identity-and-subspace attacks in the random*

[1]The exception is that if the security analysis is done under some model such as random oracle model [2], the adversary is allowed to access the specific random oracles of the underlying scheme at any stage of the game.

oracle model, assuming that solving the CDH problem in $\mathbb{G}_1$ is infeasible.

Proof. To prove Theorem 1, we prove the following two lemmas. □

Lemma 1. *In Game-I, our CLHS scheme is unforgeable against adaptive chosen identity-and-subspace attacks in the random oracle model, assuming that solving the CDH problem in $\mathbb{G}_1$ is infeasible.*

Proof. Suppose $\mathcal{A}_I$ is an adversary in Game-I who can break the unforgeability of our CLHS scheme, then we construct a $\mathcal{B}_1$ to solve the CDH problem. In addition, hash functions H_1, H_2, H_3, and H_4 are modeled as random oracles simulated by $\mathcal{B}_1$. Given a random instance $(g, g^a, g^b) \in \mathbb{G}_1^3$ of the CDH problem over the bilinear groups $(\mathbb{G}_1, \mathbb{G}_2, e, p, g)$, $\mathcal{B}_1$ interacts with $\mathcal{A}_I$ as follows.

H_4-Query: $\mathcal{B}_1$ maintains a list L_{H_4}, which is initially empty. For the query (P_{pub}, g, i), $\mathcal{B}_1$ returns the corresponding value to $\mathcal{A}_I$ if $i \in [1, n]$. Otherwise, a symbol $\perp$ is returned.

Setup: For all $i \in [1, n]$, $\mathcal{B}_1$ randomly chooses $s_i, t_i \in \mathbb{Z}_p^*$, computes $g_i = (g^a)^{s_i} g^{t_i}$, and adds (g^b, g, i, g_i) to the list L_{H_4}. Then $\mathcal{B}_1$ sets $P_{pub} = g^b$, outputs $params = (\mathbb{G}_1, \mathbb{G}_2, e, p, g, P_{pub}, g_1, \cdots, g_n, H_1, H_2, H_3, H_4)$, and initializes $\mathcal{A}_I$ with $params$.

H_1-Query: $\mathcal{B}_1$ maintains a list L_{H_1}, which is initially empty. For the query (ID_i, Y_{ID_i}), if (ID_i, Y_{ID_i}) exists in L_{H_1}, $\mathcal{B}_1$ returns the corresponding value to $\mathcal{A}_I$. Otherwise, $\mathcal{B}_1$ chooses a random $h_{1i} \in \mathbb{Z}_p^*$, adds (ID_i, Y_{ID_i}, h_{1i}) to L_{H1}, and returns it to $\mathcal{A}_I$.

Create-User: $\mathcal{B}_1$ maintains a list L_{user}, which is initially empty. For the query ID_i, $\mathcal{B}_1$ executes as follows.

- If ID_i exists in L_{user}, return the corresponding PK_{ID_i} to $\mathcal{A}_I$.
- Otherwise, randomly choose $x_{ID_i}, k_{ID_i}, h_{1i} \in \mathbb{Z}_p^*$, compute $Y_{ID_i} = g^{k_{ID_i}} (P_{pub})^{-h_{1i}}$ and $X_{ID_i} = g^{x_{ID_i}}$, set $PK_{ID_i} = (Y_{ID_i}, X_{ID_i})$, add $(ID_i, k_{ID_i}, x_{ID_i}, PK_{ID_i})$ to the list L_{user}. Then add (ID_i, Y_i, h_{1i}) to L_{H_1} and return PK_{ID_i} to $\mathcal{A}_I$.

Partial-Private-Key-Query: For the query ID_i, $\mathcal{B}_1$ returns (Y_{ID_i}, k_{ID_i}) to $\mathcal{A}_I$ if ID_i has been created. Otherwise, a symbol $\perp$ is returned.

Secret-Value-Query: For the query ID_i, $\mathcal{B}_1$ returns x_{ID_i} to $\mathcal{A}_I$ if ID_i has been created. Otherwise, a symbol $\perp$ is returned.

Public-Key-Replace: For the query $(ID_i, Y'_{ID_i}, X'_{ID_i})$, if ID_i has been created, $\mathcal{B}_1$ replaces the public key of the identity ID_i with $PK'_{ID_i} = (Y'_{ID_i}, X'_{ID_i})$ and updates the list L_{user}. Otherwise, a symbol $\perp$ is returned.

H_2-Query: $\mathcal{B}_1$ maintains a list L_{H_2}, which is initially empty. For the query $(Y_{ID_i}, X_{ID_i}, P_{pub})$, if $(Y_{ID_i}, X_{ID_i}, P_{pub})$ exists in L_{H_2}, $\mathcal{B}_1$ returns the corresponding value to $\mathcal{A}_I$. Otherwise, $\mathcal{B}_1$ chooses a random $h_{2i} \in \mathbb{Z}_p^*$, adds $(Y_{ID_i}, X_{ID_i}, P_{pub}, h_{2i})$ to L_{H_2}, and returns it to $\mathcal{A}_I$.

H_3-Query: $\mathcal{B}_1$ maintains a list L_{H_3}, which is initially empty. For the query (id_j, l), where j indicates the j-th **Sign-Query** query and $l \in [1, m]$, if (id_j, l) exists in L_{H_3}, $\mathcal{B}_1$ returns the corresponding value to $\mathcal{A}_I$. Otherwise, $\mathcal{B}_1$ chooses a random $\alpha_{jl}, \beta_{jl} \in \mathbb{Z}_p^*$, adds $(id_j, l, (g^a)^{\alpha_{jl}} g^{\beta_{jl}})$ to L_{H_3}, and returns $(g^a)^{\alpha_{jl}} g^{\beta_{jl}}$ to $\mathcal{A}_I$.

Sign-Query: For the j-th query ID_i and the properly augmented basis vectors $\mathbf{v}_{j1}, \cdots, \mathbf{v}_{jm} \in \mathbb{Z}_p^N$ of a subspace V_j, if ID_i has not been created, a symbol $\perp$ is returned. Otherwise, $\mathcal{B}_1$ first chooses a random $id_j \in \{0,1\}^\lambda$, then executes the following steps for all $l \in [1, m]$.

1) Make a **H_2-Query** and a **H_3-Query** to obtain the value h_{2i} and $(g^a)^{\alpha_{jl}} g^{\beta_{jl}}$ respectively, where $H_2(Y_{ID_i}, X_{ID_i}, P_{pub}) = h_{2i}$ and $H_3(id_j, l) = (g^a)^{\alpha_{jl}} g^{\beta_{jl}}$.
2) Compute a signature
$$\sigma_{jl} = (\prod_{u=1}^{n} g_u^{v_{jl,u}} \cdot (g^a)^{\alpha_{jl}} g^{\beta_{jl}})^{x_{ID_i} h_{2i} + k_{ID_i}}$$
of the vector $\mathbf{v}_{jl}$.

Finally, $\mathcal{B}_1$ returns $(id_j, \sigma_{j1}, \cdots, \sigma_{jm})$ to $\mathcal{A}_I$.

Forgery: After all queries, $\mathcal{A}_I$ outputs a forgery tuple $(ID^*, PK_{ID^*}, id^*, \mathbf{v}^*, \sigma^*)$, where
$$\sigma^* = (\prod_{i=1}^{n} g_i^{v_i^*} \prod_{u=1}^{m} H_3(id^*, u)^{v_{n+u}^*})^{x_{ID^*} h_2^* + y_{ID^*} + h_1^* b},$$
$h_1^* = H_1(ID^*, Y_{ID^*})$, $h_2^* = H_2(Y_{ID^*}, X_{ID^*}, P_{pub})$, and PK_{ID^*} is the user ID^*'s current public key and $\mathcal{B}_1$ might be unknown to the corresponding (x_{ID^*}, y_{ID^*}). Without loss of generality, we assume that $\mathcal{A}_I$ first submits (ID^*, Y_{ID^*}) to **H_1-Query** and then $(Y_{ID^*}, X_{ID^*}, P_{pub})$ to **H_2-Query**. No matter $\mathcal{A}_I$ outputs Type 1 Forgery or Type 2 Forgery, $\mathcal{B}_1$ can successfully solve the CDH problem as follows.

1) Due to the forking lemma [3], rewind $\mathcal{A}_I$ with the same random tape, choose $h_2^{(1)} \neq h_2^*$, and set $h_2^{(1)} = H_2(Y_{ID^*}, X_{ID^*}, P_{pub})$, $\mathcal{A}_I$ can output another forgery tuple $(ID^*, PK_{ID^*}, id^{(1)}, \mathbf{v}^{(1)}, \sigma^{(1)})$ in polynomial time, which depends on the number of **H_2-Queries**. Suppose $H_3(id^{(1)}, u) = (g^a)^{\alpha_u^{(1)}} g^{\beta_u^{(1)}}$ and $H_3(id^*, u) = (g^a)^{\alpha_u^*} g^{\beta_u^*}$ for all $u \in [1, m]$, then compute
$$T_1 = \sigma^{(1)} \cdot (X_{ID^*}^{h_2^{(1)}} Y_{ID^*} P_{pub}^{h_1^*})^{-\sum_{i=1}^{n} t_i v_i^{(1)} - \sum_{u=1}^{m} \beta_u^{(1)} v_{n+u}^{(1)}},$$
$$T_2 = \sigma^* \cdot (X_{ID^*}^{h_2^*} Y_{ID^*} P_{pub}^{h_1^*})^{-\sum_{i=1}^{n} t_i v_i^* - \sum_{u=1}^{m} \beta_u^* v_{n+u}^*},$$
$$g^{a x_{ID^*}} = (\frac{T_2^{(\sum_{i=1}^{n} s_i v_i^* + \sum_{u=1}^{m} \alpha_u^* v_{n+u}^*)^{-1}}}{T_1^{(\sum_{i=1}^{n} s_i v_i^{(1)} + \sum_{u=1}^{m} \alpha_u^{(1)} v_{n+u}^{(1)})^{-1}}})^{(h_2^* - h_2^{(1)})^{-1}}.$$
2) Due to the forking lemma [3], rewind $\mathcal{A}_I$ with the same random tape, choose $h_1^{(1)} \neq h_1^*$, and set $h_1^{(1)} = H_1(ID^*, Y_{ID^*})$, $\mathcal{A}_I$ can output another forgery tuple $(ID^*, PK'_{ID^*}, id^{(2)}, \mathbf{v}^{(2)}, \sigma^{(2)})$ in polynomial time,

which depends on the number of H_1-**Queries**. Since $PK'_{ID^*} = (Y_{ID^*}, X'_{ID^*})$ and X'_{ID^*} may not be equal to X_{ID^*}, rewind $\mathcal{A}_I$ in the same method as that in step 1) to obtain $(X'_{ID^*})^a = g^{ax'_{ID^*}}$. Suppose $H_2(Y_{ID^*}, X'_{ID^*}, P_{pub}) = h_2^{(2)}$ and $H_3(id^{(2)}, u) = (g^a)^{\alpha_u^{(2)}} g^{\beta_u^{(2)}}$ for all $u \in [1, m]$, then compute

$$T_3 = (g^{ax_{ID^*}})^{-h_2^*(\sum_{i=1}^{n} s_i v_i^* + \sum_{u=1}^{m} \alpha_u^* v_{n+u}^*)} \cdot T_2,$$

$$T_4 = \sigma^{(2)} \cdot (g^{ax'_{ID^*}})^{-h_2^{(2)}(\sum_{i=1}^{n} s_i v_i^{(2)} + \sum_{u=1}^{m} \alpha_u^{(2)} v_{n+u}^{(2)})} \cdot ((X'_{ID^*})^{h_2^{(2)}} Y_{ID^*} P_{pub}^{h_1^{(1)}})^{-\sum_{i=1}^{n} t_i v_i^{(2)} - \sum_{u=1}^{m} \beta_u^{(2)} v_{n+u}^{(2)}},$$

$$g^{ab} = \left(\frac{T_3^{(\sum_{i=1}^{n} s_i v_i^* + \sum_{u=1}^{m} \alpha_u^* v_{n+u}^*)^{-1}}}{T_4^{(\sum_{i=1}^{n} s_i v_i^{(2)} + \sum_{u=1}^{m} \alpha_u^{(2)} v_{n+u}^{(2)})^{-1}}}\right)^{(h_1^* - h_1^{(1)})^{-1}}.$$

If $\mathcal{A}_I$ first submits $(Y_{ID^*}, X_{ID^*}, P_{pub})$ to H_2-**Query** then (ID^*, Y_{ID^*}) to H_1-**Query**, $\mathcal{B}_1$ can also solve the CDH problem in a similar method as above.

Now we analyze $\mathcal{B}_1$'s probability of failure. There is no abort in the simulation as $\mathcal{B}_1$ manages to respond to every query made by $\mathcal{A}_I$. During the **Forgery** phase, since $s_1, \cdots, s_n, \alpha_1^*, \cdots, \alpha_m^*$ are each independently uniform in $\mathbb{Z}_p^*$ even conditioned on the view of $\mathcal{A}_I$, the event that $\sum_{i=1}^{n} s_i v_i^* + \sum_{u=1}^{m} \alpha_u^* v_{n+u}^* = 0$ occurs with a probability $\frac{1}{p}$. Therefore, the probability that $\mathcal{B}_1$ can solve the CDH problem is $Adv_{\mathcal{B}_1,CDH}(\lambda) \geq Adv^{csa,cida}_{\mathcal{A}_I,CLHS}(\lambda) - \frac{1}{p}$. This completes the proof of Lemma 1. □

Lemma 2. *In Game-II, our CLHS scheme is unforgeable against adaptive chosen identity-and-subspace attacks in the random oracle model, assuming that solving the CDH problem in $\mathbb{G}_1$ is infeasible.*

Proof. Suppose $\mathcal{A}_{II}$ is an adversary in Game-II who can break the unforgeability of our CLHS scheme, then we construct a $\mathcal{B}_2$ to solve the CDH problem. In addition, hash functions H_3 and H_4 are modeled as random oracles simulated by $\mathcal{B}_2$. Given a random instance $(g, g^a, g^b) \in \mathbb{G}_1^3$ of the CDH problem over the bilinear groups $(\mathbb{G}_1, \mathbb{G}_2, e, p, g)$, $\mathcal{B}_2$ selects a random $\eta \in \{1, \cdots, q_c\}$, where q_c is the number of **Create-User** queries made by $\mathcal{A}_{II}$, then interacts with $\mathcal{A}_{II}$ as follows.

H_4-**Query:** $\mathcal{B}_1$ maintains a list L_{H_4}, which is initially empty. For the query (P_{pub}, g', i), $\mathcal{B}_1$ returns the corresponding value to $\mathcal{A}_I$ if (P_{pub}, g', i) exists in L_{H_4}. Otherwise, $\mathcal{B}_1$ randomly chooses $s_i, t_i \in \mathbb{Z}_p^*$, computes $g_i = (g^a)^{s_i} g^{t_i}$, returns it to $\mathcal{A}_I$, and adds (P_{pub}, g', i, g_i) to L_{H_4}.

Setup: $\mathcal{A}_{II}$ first generates g' and P_{pub}, then obtains g_i by submitting (P_{pub}, g', i) to the oracle H_4-**Query** for all $i \in [1, n]$. Finally, $\mathcal{A}_{II}$ outputs $params = (\mathbb{G}_1, \mathbb{G}_2, e, p, g', P_{pub}, g_1, \cdots, g_n, H_1, H_2, H_3, H_4)$ to $\mathcal{B}_2$. Suppose $g' = g^r$, $P_{pub} = g^{sr}$, and s is the master key. Note that $\mathcal{B}_2$ does not know the knowledge of both s and r.

Create-User: $\mathcal{B}_2$ maintains a list L_{user}, which is initially empty. For the query $(ID_i, Y_{ID_i}, k_{ID_i})$, where (Y_{ID_i}, k_{ID_i}) is the identity ID_i's partial private key, $\mathcal{B}_2$ returns the corresponding PK_{ID_i} to $\mathcal{A}_{II}$ if ID_i exists in L_{user}. Otherwise, $\mathcal{B}_2$ executes as follows.

- If $ID_i = ID_\eta$, set $X_{ID_i} = g^b$, $x_{ID_i} = \perp$, and $PK_{ID_i} = (Y_{ID_i}, X_{ID_i})$. Then add $(ID_i, k_{ID_i}, x_{ID_i}, PK_{ID_i})$ to L_{user} and return PK_{ID_i} to $\mathcal{A}_{II}$.
- Otherwise, randomly choose $x_{ID_i} \in \mathbb{Z}_p^*$, compute $X_{ID_i} = g^{x_{ID_i}}$, set $PK_{ID_i} = (Y_{ID_i}, X_{ID_i})$, add $(ID_i, k_{ID_i}, x_{ID_i}, PK_{ID_i})$ to L_{user}, and return PK_{ID_i} to $\mathcal{A}_{II}$.

Secret-Value-Query: For the query ID_i, if ID_i does not exist in L_{user}, a symbol $\perp$ is returned. Otherwise, $\mathcal{B}_2$ executes as follows.

- If $ID_i \neq ID_\eta$, return x_{ID_i} to $\mathcal{A}_{II}$.
- Otherwise, abort.

Public-Key-Replace: For the query $(ID_i, Y'_{ID_i}, X'_{ID_i})$, if ID_i has been created, $\mathcal{B}_2$ replaces the public key of the identity ID_i with $PK'_{ID_i} = (Y'_{ID_i}, X'_{ID_i})$ and updates the list L_{user}. Otherwise, a symbol $\perp$ is returned.

H_3-**Query:** $\mathcal{B}_2$ maintains a list L_{H_3}, which is initially empty. For the query (id_j, l), where j indicates the j-th **Sign-Query** query and $l \in [1, m]$, if (id_j, l) exists in L_{H_3}, $\mathcal{B}_2$ returns the corresponding value to $\mathcal{A}_{II}$. Otherwise, $\mathcal{B}_2$ randomly chooses $\alpha_{jl}, \beta_{jl} \in \mathbb{Z}_p^*$, adds $(id_j, l, (g^a)^{\alpha_{jl}} g^{\beta_{jl}})$ to L_{H_3}, and returns $(g^a)^{\alpha_{jl}} g^{\beta_{jl}}$ to $\mathcal{A}_{II}$.

Sign-Query: For the j-th query ID_i and the properly augmented basis vectors $\mathbf{v}_{j1}, \cdots, \mathbf{v}_{jm} \in \mathbb{Z}_p^N$ of a subspace V_j, if ID_i has not been created, a symbol $\perp$ is returned. Otherwise, $\mathcal{B}_2$ first chooses a random $id_j \in \{0,1\}^\lambda$, then executes the following steps for all $l \in [1, m]$.

1) If $ID_i = ID_\eta$, select a random $\beta_{jl} \in \mathbb{Z}_p^*$, compute $\alpha_{jl} = -\sum_{u=1}^{n} s_u v_{jl,u}$, and check the list L_{H_3}:
   - If (id_j, l) has already existed L_{H_3}, abort.
   - Otherwise, add $(id_j, l, (g^a)^{\alpha_{jl}} g^{\beta_{jl}})$ to L_{H_3}.

   Then compute $h_{2i} = H_2(Y_{ID_i}, X_{ID_i}, P_{pub})$ and a signature

   $$\sigma_{jl} = (\prod_{u=1}^{n} g^{t_u v_{jl,u}} \cdot g^{\beta_{jl}})^{k_{ID_i}} \cdot (X_{ID_i}^{h_{2i}})^{\sum_{u=1}^{n} t_u v_{jl,u} + \beta_{jl}}$$

   of the vector $\mathbf{v}_{jl}$.
2) Otherwise, $ID_i \neq ID_\eta$. In this case, compute $h_{2i} = H_2(Y_{ID_i}, X_{ID_i}, P_{pub})$, make a H_3-**Query** to obtain the value of $H_3(id_j, l)$, a signature

   $$\sigma_{jl} = (\prod_{u=1}^{n} g_u^{v_{jl,u}} \cdot H_3(id_j, l))^{x_{ID_i} h_{2i} + k_{ID_i}}$$

   of the vector $\mathbf{v}_{jl}$.

Finally, $\mathcal{B}_2$ returns $(id_j, \sigma_{j1}, \cdots, \sigma_{jm})$ to $\mathcal{A}_{II}$.

Forgery: After all queries, $\mathcal{A}_{II}$ outputs a forgery tuple $(ID^*, PK_{ID^*}, id^*, \mathbf{v}^*, \sigma^*)$, where

$$\sigma^* = (\prod_{i=1}^{n} g_i^{v_i^*} \prod_{u=1}^{m} H_3(id^*, u)^{v_{n+u}^*})^{x_{ID^*} h_2^* + k_{ID^*}},$$

$h_1^* = H_1(ID^*, Y_{ID^*})$, $h_2^* = H_2(Y_{ID^*}, X_{ID^*}, P_{pub})$. If $ID^* \neq ID_\eta$, $\mathcal{B}_2$ aborts. Otherwise, we consider the following two types of forgery.

If $\mathcal{A}_{II}$ outputs a Type 1 Forgery, $\mathcal{B}_2$ can successfully solve the CDH problem as follows,

1) For all $u \in [1, m]$, suppose $H_3(id^*, u) = (g^a)^{\alpha_u^*} g^{\beta_u^*}$.
2) Compute

$$T_1 = \sigma^* \cdot (\prod_{i=1}^{n} g_i^{v_i^*} \prod_{u=1}^{m} H_3(id^*, u)^{v_{n+u}^*})^{-k_{ID^*}},$$

$$g^{ab} = (T_1^{\frac{1}{h_2^*}} \cdot g^{-\sum_{i=1}^{n} t_i v_i^* - \sum_{u=1}^{m} \beta_u^* v_{n+u}^*})^{\frac{1}{\sum_{i=1}^{n} s_i v_i^* + \sum_{u=1}^{m} \alpha_u^* v_{n+u}^*}}.$$

If $\mathcal{A}_{II}$ outputs a Type 2 Forgery, $\mathcal{B}_2$ can successfully solve the CDH problem as follows,

1) Suppose $H_3(id^*, u) = (g^a)^{\alpha_u^*} g^{\beta_u^*}$ for all $u \in [1, m]$ and $H_3(id^*, u) = H_3(id_{j'}, u)$ for some $j' \in [1, q_s]$, where q_s is the number of **Sign-Queries** made by $\mathcal{A}_{II}$.
2) Compute $\tilde{\mathbf{v}}^* = \mathbf{v}^* - \sum_{i=1}^{m} v_{n+i}^* \mathbf{v}_{j'i}$ $(\tilde{\mathbf{v}}^* \neq 0)$,

$$\tilde{\sigma}^* = \frac{\sigma^*}{\prod_{i=1}^{m} \sigma_{j'i}^{v_{n+i}^*}} = (\prod_{i=1}^{n} g_i^{\tilde{v}_i^*})^{x_{ID^*} h_2^* + k_{ID^*}},$$

$$g^{ab} = ((\tilde{\sigma}^* \cdot (\prod_{i=1}^{n} g_i^{\tilde{v}_i^*})^{-k_{ID^*}})^{\frac{1}{h_2^*}} \cdot g^{-\sum_{i=1}^{n} t_i \tilde{v}_i^*})^{\frac{1}{\sum_{i=1}^{n} s_i \tilde{v}_i^*}}.$$

Now we analyze $\mathcal{B}_2$'s failure probability. First, $\mathcal{B}_2$ chooses a random η such that $ID^* \neq ID_\eta$ before simulation, and the probability of this event is $1 - \frac{1}{q_c}$. Second, if $\mathcal{B}_2$ responds to a **Sign-Query** query by choosing an identifier id such that $\mathcal{A}_2$ has already asked H_3**-Query** to obtain the value of $H_3(id, u)$ for some u, the simulation aborts. And the probability of this event is at most $\frac{q_s(q_s+q_{h_3})}{p}$, where q_s and q_{h_3} are the number of **Sign-Query** queries and H_3**-Queries** made by $\mathcal{A}_{II}$. Third, since $s_1, \cdots, s_n, \alpha_1^*, \cdots, \alpha_m^*$ are each independently uniform in $\mathbb{Z}_p^*$ even conditioned on the view of $\mathcal{A}_{II}$, the probability that either $\sum_{i=1}^{n} s_i v_i^* + \sum_{u=1}^{m} \alpha_u^* v_{n+u}^* = 0$ or $\sum_{i=1}^{n} s_i \tilde{v}_i^* = 0$ is $\frac{1}{p}$. Therefore, the probability that $\mathcal{B}_2$ can solve the CDH problem is $Adv_{\mathcal{B}_2, CDH}(\lambda) \geq (Adv_{\mathcal{A}_{II}, CLHS}^{csa, cida}(\lambda) - \frac{q_s(q_s+q_{h_3})}{p} - \frac{1}{p}) \cdot \frac{1}{q_c}$. This completes the proof of Lemma 2. □

REFERENCES

[1] M. H. Au, Y. Mu, J. Chen, D. S. Wong, J. K. Liu, and G. Yang, "Malicious kgc attacks in certificateless cryptography," in *Proceedings of the 2nd ACM symposium on Information, computer and communications security*, 2007, pp. 302–311.

[2] M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," in *Proceedings of the 1st ACM Conference on Computer and Communications Security*, 1993, pp. 62–73.

[3] D. Pointcheval and J. Stern, "Security arguments for digital signatures and blind signatures," *Journal of cryptology*, vol. 13, pp. 361–396, 2000.