1 Security Model

The security of a multi-TA WIBE is parametrized by a bit $b \in \{0,1\}$ and a positive integer k through the following game. An attacker, $\mathcal{A}$, is tasked with guessing the value of b chosen by the challenger. Initially an attacker is given the public parameters for the system. These parameters include $(g_1, g_2, u_{1,0}, u_{1,1}, u_{2,0}, u_{2,1}, \ldots, u_{L,0}, u_{L,1}) \in \mathbb{G}$ defining a scheme of maximum depth L, an identity space $\mathcal{ID} = \{0,1\}^k$, and a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}^T$. In the first phase the attacker is allowed to query the challenger. For chosen-plaintext security (IND-ID-CPA), the attacker is given polynomially many queries to the following oracles:

- `CreateTA`(TA): The oracle computes $(pk_i, sk_i) \xleftarrow{\$} \texttt{Setup}(1^k, TA)$ for the TA identity TA_i and returns pk_i. This oracle can only be queried once for each identity TA_i.
- `SetupCoalitionBroadcast`($TA_i, \mathcal{C}$): This oracle runs the `SetupCoalitionBroadcast` algorithm for coalition $\mathcal{C}$ containing TA_i and returns messages $w_{i,j} \forall TA_j \in \mathcal{C} \setminus TA_i$.
- `SetupCoalitionKeys`($TA_i, w_{j,i} \forall TA_j \in \mathcal{C} \setminus TA_i$): This oracle can only be queried if each $TA_i, TA_j \in \mathcal{C}$ has been queried to the `SetupCoalitionBroadcast` oracle with k TA's in the coalition. The oracles runs the `SetupCoalitionKeys` algorithm assuming that message $w_{j,i}$ was sent by TA_j. Note that this does not imply that all the TAs believe that they're in the same coalition.
- `Corrupt`($\boldsymbol{ID}$): The oracle returns $d_{\boldsymbol{ID}}$ for the identity $\boldsymbol{ID}$. Note that if $\boldsymbol{ID} = TA_i$ then this method returns TA_i's secret key sk_i.

For chosen-ciphertext security, the attacker is additionally given access to decryption oracles:

- `UserDecrypt`($\boldsymbol{ID}, C^*$): This oracle decrypts the ciphertext with the decryption key $d_{\boldsymbol{ID}}$.
- `CoalitionDecrypt`($\mathcal{C}, \boldsymbol{ID}, C^*$): This oracle decrypts the ciphertext with the coalition decryption key $c_{\boldsymbol{ID}}$ corresponding to coalition $\mathcal{C}$.

To end the first phase the attacker outputs two equal-length messages m_1, m_2 and a challenge pattern $\boldsymbol{P}^{=}(P_1, P_2, \ldots, P_\ell)*$ for $P_i \in \mathcal{ID} \cup \{*\}$. This is modeled as a single query to the oracle:

- `Test`($\mathcal{C}^*, P, m_0, m_1$): This oracle takes as input two messages (m_0, m_1) of equal length. It encrypts the message m_b for the coalition using $pk_i \forall TA_i \in \mathcal{C}^*$ under the pattern P. This oracle may only be access once and outputs a ciphertext C^*. We will let $\mathcal{C}^*$ denote the challenge coalition $(TA_a, \ldots, TA_k)$.

The relationship between the attacker's queries and choice of $\boldsymbol{P}^*$ are restricted. Specifically, the attacker is not allowed access to the decryption keys of nodes matching the challenge pattern nor nodes capable of generating those keys. The disallowed oracle queries are:

1. A `Corrupt` query for any $\boldsymbol{ID}$ matching pattern P
2. A `Corrupt` query for any $\boldsymbol{ID}$ that is an ancestor of pattern P, i.e. there exists $\boldsymbol{ID}^*$ such that $\boldsymbol{ID} \| \boldsymbol{ID}^*$ matches the pattern P
3. A `UserDecrypt` decrypt query for C^* and any user ID in the test coalition matching the pattern P or an ancestor thereof.

For the second phase the attacker is once again given access to the queries from the first phase (still restricted as above). The attacker ends the game by outputting a bit b' as its guess for the value of bit b. We define the attacker's advantage as:

$$Adv_{\mathcal{A}}^{\texttt{IND}}(k) = |Pr[b' = b] - \tfrac{1}{2}|$$

2 Non-Selective Identity Security in the Random Oracle Model

Like the Boneh-Boyen HIBE it is based on [**?**], the multi-*TA* scheme is only proven secure in the (selective-identity) multi-TA IND-sWID-CPA model. For non-wildcard single *TA* schemes, selective-identity IND-s(H)ID security in the standard model can be transformed into non-selective-identity secure IND-(H)ID schemes in the random oracle model [**?**]. This technique was extended to securely enable wildcards in IND-sHID-CPA WIBE schemes [**?**] and we further extend it to prove non-selective-identity security of our proposed multi-*TA* scheme in the random oracle model.

The security game in the random oracle model is the same as in the standard model with the addition of one new oracle [**?**]. This oracle responds to each query with a uniformly random element chosen from its output domain. Additionally, for each specific query the response is always the same whenever that query is made.

To prove security in a non-selective-identity setting we alter the routines of the multi-TA WIBE scheme to use the hash of an identity ($H_i(ID_i)$) rather than the identity (ID_i) for each level i. The key derivation and encryption routines pass the identity to a hash function (modeled as a random oracle) and use the result in place of the identity in the relevant algorithm. This hash function maps from the finite space of identifiers to an equal or smaller sized target space. This transform changes the pattern $\boldsymbol{P} = (P_1, P_2, ..., P_\ell)$ into pattern $H(P) = \boldsymbol{P'} = (P'_1, P'_2, ..., P'_\ell)$ as follows:

$$P'_i = H_i(P_i) \text{ for } i \notin W(P), \text{ otherwise } H(P_i) = * = P'_i$$

We note that a finite family of independent random oracles of predetermined maximum size may be simulated using a single random oracle. Given a single random oracle $H(\text{query})$, a WIBE of maximum depth L. and a fixed-length format to enumerate $i \in \{1 \dots L\}$, we may create i independent random oracles. Each oracle $H_i(\text{query})$ corresponding to level i of the WIBE is simulated by prepending the query with the number of the oracle it is addressed to, $H_i(\text{query}) = H(i||\text{query})$.

Theorem 1. *Suppose that there exists a polynomial-time attacker $\mathcal{A}$ against the non-selective-identity multiple TA IND-WID-CPA Boneh-Boyen WIBE with advantage ϵ with access to q_H queries from each of the L random oracles associated with hierarchy levels, then there exists a polynomial-time attacker $\mathcal{B}$ against the selective-identity multiple TA IND-sWID-CPA Boneh-Boyen WIBE with advantage ϵ' when allowed q_K key derivation queries with ϵ' bounded:*

$$\epsilon' \geq \left(\frac{\epsilon}{2^n L(q_H+1)^L}\right)\left(1 - \frac{(q_H+1)}{|ID|}\right)^L$$

Proof

Suppose there exists adversary $\mathcal{A}$ against the multi-TA IND-WID-CPA scheme (with hashed identities), we will construct an adversary $\mathcal{B}$ which gains an advantage against the multi-TA IND-sWID-CPA scheme using the algorithm $\mathcal{A}$ as a black box.

For a multi-TA WIBE of maximum depth L consisting of a maximum of k $TA's$, the algorithm $\mathcal{B}$ runs as follows:

1. $\mathcal{B}$ randomly chooses a length $\ell^* \in \{1 \dots L\}$ for its challenge identity vector. Also, $\mathcal{B}$ randomly chooses a coalition $\boldsymbol{TA}^* = (TA_1^*, \dots, TA_n^*)$ from all possible non-empty combinations of the k *TA*'s. The numbering assignment of *TA*'s in the coalition is assigned randomly.
2. $\mathcal{B}$ chooses a challenge pattern $\boldsymbol{P}^*$ as follows. First, $\mathcal{B}$ randomly assigns a family $t_i \xleftarrow{\$} \{0, 1, ..., q_H\}$ for all $i \in \{1, ..., \ell\}$. If $t_i = 0$ then $\mathcal{B}$ assigns $P_i = *$. For $t_1 \neq 0$ $\mathcal{B}$ randomly chooses $P_1 \xleftarrow{\$} \{TA_1, *\}$. For $i > 1$ with $t_i \neq 0$ $\mathcal{B}$ randomly chooses $P_i \xleftarrow{\$} \mathcal{ID}$
3. The challenger responds with WIBE parameters $param = (\hat{g}_1, \hat{g}_2, \hat{u}_{0,0}, \dots, \hat{u}_{L,1})$. $\mathcal{B}$ initiates $\mathcal{A}$ with the same parameters and a family of hashes H_i for each level i allowed in the HIBE.

4. Since both $\mathcal{B}$ and $\mathcal{A}$ are in WIBE environments playing the chosen plaintext game, they have access to the same set of oracles with $\mathcal{A}$ having access to an additional random oracle. To prepare for random oracle queries from $\mathcal{A}$, $\mathcal{B}$ initializes associative lists J_i for each level i allowed in the HIBE to answer queries for each oracle H_i. Initially empty, $\mathcal{B}$ must either track the number of entries made on each list with a counter or equivalently through measure of its size, we let $|J_i| \leftarrow 1$ denote the size of empty lists. When $\mathcal{A}$ queries random oracle H_i on identity ID, $\mathcal{B}$ answers as follows:
   - If $J_i(ID)$ has been previously defined, then $\mathcal{B}$ returns $J_i(ID)$.
   - Otherwise:
     - For $t_i = J_i$, if $i = 1$ then $\mathcal{B}$ assigns the value $J_1(ID) \leftarrow TA_1$. If $i \neq 1$ then $\mathcal{B}$ assigns the value $J_i \leftarrow P_i^*$.
     - For $t_i \neq J_i$, $\mathcal{B}$ assigns $J_i(ID) \leftarrow H_i(ID)$ and increments the counter $|J_i|$.
     - $\mathcal{B}$ then returns the value $J_i(ID)$
5. For other queries, $\mathcal{B}$ hashes the relevant identity pattern $\boldsymbol{P}$ as a random oracle query with result $\boldsymbol{P'}$. Note that for TA-level queries with $\boldsymbol{ID} = (TA_i)$ the pattern is still transformed as $\boldsymbol{ID'} = (J_1(TA_i))$. $\mathcal{A}$ may query:
   - $\texttt{CreateTA}(TA_i)$ $\mathcal{B}$ computes $TA_i' = J_1(TA_i)$ and forwards the query using its own oracle as $\texttt{CreateTA}(TA_i')$, returning the result.
   - $\texttt{Corrupt}(\boldsymbol{P})$ If $J(\boldsymbol{P}) \in_* \boldsymbol{P^*}$ then $\mathcal{B}$ aborts because it would have to make an illegal query. Otherwise, $\mathcal{B}$ computes $\boldsymbol{P'} = J(\boldsymbol{P})$ and forwards the query using its own oracle as $\texttt{Corrupt}(\boldsymbol{P'})$, returning the result.
   - $\texttt{SetupCoalitionBroadcast}(TA_i, \mathcal{C})$ For each TA_j in the coalition $\mathcal{C}$, $\mathcal{B}$ computes the hashed TA identity $TA_j' = J_1(TA_j)$ and adds it to coalition $\mathcal{C}'$. $\mathcal{B}$ also computes the hashed identity $TA_i' = J_1(TA_i)$ then forwards the query as $\texttt{SetupCoalitionBroadcast}(TA_i', \mathcal{C}')$, returning the result.
   - $\texttt{SetupCoalitionKeys}(TA_j, sk_j, \mathcal{C}, \hat{W}_j)$ For each TA_j in the coalition $\mathcal{C}$, $\mathcal{B}$ computes the hashed TA identity $TA_j' = J_1(TA_j)$ and adds it to coalition $\mathcal{C}'$. $\mathcal{B}$ then computes $TA_j' = J_1(TA_j)$ and forwards the query as $\texttt{SetupCoalitionKeys}(TA_j', sk_j, \mathcal{C})$, returning the result.
   - $\texttt{ExtractCoalitionKey}(\mathcal{C}, v_j, d_{\boldsymbol{ID}})$ For each TA_j in the coalition $\mathcal{C}$, $\mathcal{B}$ computes the hashed TA identity $TA_j' = J_1(TA_j)$ and adds it to coalition $\mathcal{C}'$. $\mathcal{B}$ then computes $\boldsymbol{ID'} = J(\boldsymbol{ID})$ and queries $\texttt{ExtractCoalitionKey}(\mathcal{C}', v_j, d_{\boldsymbol{ID'}})$, returning the result.
6. $\mathcal{A}$ ends this stage of the game by outputting a challenge pattern $\widehat{\boldsymbol{P}} = (\widehat{P}_1, ..., \widehat{P}_\ell)$ and challenge coalition $\widehat{\boldsymbol{TA}} = \{\widehat{TA_1}, ..., \widehat{TA_k}\}$ and two equal-length messages (m_0, m_1). If any $J_i(\widehat{P}_i)$ has not yet been defined then $\mathcal{B}$ sets $J_i(\widehat{P}_i) \leftarrow H_i(\widehat{P}_i)$.
7. $\mathcal{B}$ must abort if:
   - if $\ell \neq \ell^*$
   - if there exists $i \in W(\widehat{\boldsymbol{P}})$ such that $i \notin W(\boldsymbol{P}^*)$
   - if $J_i(\widehat{P}_i) \neq P_i^*$
8. If $\mathcal{B}$ has not aborted it outputs the messages (m_0, m_1) to the challenger.
9. The challenger computes the challenge ciphertext C^* by encrypting message m_β for $\beta \xleftarrow{\$} \{0, 1\}$ and returns it to $\mathcal{B}$.
10. $\mathcal{B}$ in turn returns the challenge ciphertext to $\mathcal{A}$, which outputs a bit $\widehat{\beta}$ as a guess for the value of β.
11. In turn, $\mathcal{B}$ outputs $\widehat{\beta}$ as its guess for the value of β.

The algorithm $\mathcal{B}$ wins the IND-sWID-CPA game if $\mathcal{A}$ wins the IND-WID-CPA game and $\mathcal{B}$ does not abort. $\mathcal{B}$ may abort if it has guess the challenge coalition and pattern incorrectly, or if it was forced to make an illegal query. $\mathcal{B}$ uses the t_i's to guess which of $\mathcal{A}$'s queries will be used to define the i^{th} level of its challenge pattern with $t_i = 0$ corresponding to a wildcard.

To avoid $\mathcal{B}$'s making of an illegal query we require that there are no collisions in the hash. For distinct identifiers $ID \neq ID'$ we require that the hash output $H_i(ID) \neq H_i(ID')$. With a random oracle such a collision could only occur by chance. Given $(q_H + 1)$ random oracle queries for a level the probability of a collision has an upper bound of $(q_H + 1)/|ID|$. Then the probability of successfully avoiding a collision has a lower bound of $(1 - (q_H + 1)/|ID|)$. With L independent random oracles, the probability of avoiding a

collision on all levels is then $(1-(q_H+1)/|ID|)^L$. If a collision occurs then $\mathcal{B}$ will gain no benefit from $\mathcal{A}$'s advantage as it cannot properly simulate the WIBE environment. Of $\mathcal{A}$'s original advantage ϵ, $\mathcal{B}$ may only leverage an advantage of $\epsilon(1-(q_H+1)/|ID|)^L$.

We require that $\mathcal{B}$ correctly choose the challenge pattern and coalition in order to win the game. The probability that $\ell = \ell^*$ is $1/L$, the probability that t_i correctly identifies P_i is $1/(q_H+1)$ for each level $i \leq L$, and the probability of the challenge coalition $\mathcal{C}$ being correct as $1/2^n$. Each of these factors reduce $\mathcal{B}$'s advantage multiplicatively in comparison to $\mathcal{A}$'s. If $\mathcal{B}$ correctly guesses these values and $\mathcal{A}$ never forces $\mathcal{B}$ to make an illegal query then $\mathcal{B}$ has advantage:

$$\epsilon' \geq \left(\frac{\epsilon}{2^n L(q_H+1)^L}\right)\left(1-\frac{(q_H+1)}{|ID|}\right)^L$$

□