Broadcast Encryption with Multiple Trust Centers and Dynamic Coalitions

Christopher A. Seaman[1], Kent D. Boklan[2], and Alexander W. Dent[3]

[1] Graduate Center, City University of New York, USA
[2] Queens College, City University of New York, USA
[3] Royal Holloway, University of London, UK

Abstract. We demonstrate an extension of hierarchical identity-based encryption (HIBE) from the domain of a single Trusted Authority (TA) to a coalition of multiple independent Trusted Authorities with their own hierarchies. Coalitions formed under this scheme may be temporary or dynamic in membership without compromising security. In this paper we give an example instantiation with formal security model and a proof of security against selective-identity chosen-plaintext attacks in the standard model based upon the Bilinear Decisional Diffie-Hellman assumption (BDDH).

1 Introduction and Motivation

1.1 Identity-Based Encryption

Identity-based encryption (IBE) [**?**] has become an attractive alternative to public-key encryption schemes. A Public Key Infrastructure (PKI) allows for encryption only after obtaining a certified copy of the recipient's public key. Under identity-based encryption a message is encrypted using a personal identifier such as an email address rather than using a certified public key obtained from a PKI. The public key infrastructure is replaced by an identification system and a single public set of common parameters.

Identity-based encryption is most often considered in the context of one-to-one communication within a single Trusted Authority (TA). This TA is the root arbiter of trust within its domain, it is trusted with the secret key from which it derives all decryption keys for individuals in the organization. In an IBE system an encrypted message may only be sent to a single entity within the domain of that specific TA. This paper considers a coalition of TA's desiring secure one-to-many communication, with each TA retaining the security of its secret keys. This secrecy requirement is natural in the setting of dynamic coalitions.

1.2 Known Extensions of IBE

A hierarchical identity-based encryption (HIBE) scheme [**?**] is an extension of identity-based encryption with identities organized in a tree with a single root master authority. An identity in non-hierarchical IBE is a unique single identifier, e.g. "Bob Smith serial number 123456789". An identity in a HIBE scheme is an ordered collection of identifiers which can mirror the organizational structure of the TA, e.g. ("US Army", "2-16 Infantry", "Bob"). Consider a two-level TA called the "US Army" with an analogous but simplified hierarchical structure. The first level of the hierarchy will identify a particular grouping of personnel, our examples use groupings at either the regiment or mission level, with the secret key for a grouping, e.g. ("US Army", "2-16 Infantry"), being held by its commanding officer. The second level will identify a specific individual or role within the grouping, our examples use individuals' names for this level. In a hierarchical IBE, as in regular IBE, encryption is accomplished by using the identity of a recipient in place of using the recipient's certified public key.

The hierarchical structure is meant to mirror the trust or authority structure of the organization. Any entity in a HIBE can use its own secret key to generate a secret key for any subordinate. If the head of the 2-16 infantry battalion has the secret key for identity ("US Army", "2-16 Infantry") then he is able to generate a secret keys of the form ("US Army", "2-16 Infantry", "Name"). Messages may then be decrypted

by any 'ancestor' capable of generating a secret key for the addressee, but a subordinate (or 'child') is not capable of generating a secret key for or decrypting messages addressed to any ancestor. This characteristic of HIBEs makes it absolutely necessary that the root authority of the hierarchy have the full faith and confidence of all members of the hierarchy. The root authority holds the master key for the HIBE scheme, making it capable of generating a secret key for any entity in the hierarchy and decrypting any message sent using the scheme.

One-to-many secure communication is a powerful cryptographic tool. Take a MANETs setting for example, where transmission is much more expensive than computation, one-to-many communication allows for a single transmitted message to be read by any number of valid recipients within range. Multiple recipients may decrypt the same message through the use of one or more 'wildcards'. A wildcard is a special character, commonly denoted '*', that may be used in an address in lieu of specifying a particular aspect of an identity. Anyone with an identifier matching the non-wildcard portion is then able to read the message.

The method of employing wildcards into hierarchical identity-based encryption (WIBE) structures [**?**] allows any individual to send messages to entire levels of a TA's hierarchy. To send a message to every member of a regiment in a HIBE, separate messages must be sent for individual in addressed as ("US Army", "2-16 Infantry", "Samuel Adams"). In a wildcard HIBE setting a single message could be addressed to ("US Army", "2-16 Infantry", *) which could then be read by each member of the regiment. A message addressed to the leadership of regiments could be sent to ("US Army", *) allowing for quick dissemination of command-level materials. Wildcards are also useful in the middle level of a hierarchy, a message can be addressed as ("US Army", *, "John Locke") to reach a "John Locke" regardless of his current regiment.

The wildcard method of multicast communication is limited by the structure of the hierarchy: it cannot distinguish between entities within a single hierarchical level. For example, a single message to ("'US Army","Regiment", *) could be read by ("'US Army","Regiment", "Alice"), ("'US Army","Regiment", "Bob"), and ("'US Army","Regiment", "Eve"); however, it would not be possible to send a single message to Alice and Bob without also allowing Eve to read it. Likewise, a message may be addressed to a single specified regiment or to all regiments by wildcard. In general, wildcard-based encryption allows you to specify either a single identity or all identities at each level of the hierarchy; it does not allow you to choose a specific subset of identities within a level of the hierarchy.

Other researchers have considered the question of developing IBE systems with multiple trusted authorities. For example, a scheme proposed by Paterson and Srinivasan [**?**] and another by Boklan *et al.* [**?**] introduced multi-TA IBE systems which allow trusted authorities to interact in order to derive a shared IBE system. The Paterson and Srinivasan [**?**] paper constructed an IBE scheme which supported multiple trust authorities in a way which makes it infeasible for an attacker to determine which trust authority's public parameters was used to form the ciphertext – i.e. the ciphertext preserves the anonymity of the trust authority. However, the Paterson and Srinivasan scheme does not allow trust authorities to form trust coalitions. The Boklan *et al.* scheme [**?**] allows trust authorities to cooperate to form trust coalitions, in the sense that within the coalition a private key issued by TA_i for an identity $\boldsymbol{ID}$ can be translated into a private key issued by TA_j for the same identity. However, in order to achieve this functionality, the scheme requires that the coalition TAs setup their master public parameters and master private keys simultaneously. Furthermore, every TA can deduce the master private key of every other TA. This is clearly a severe disadvantage for any setting where the TAs share anything less than complete trust in each others' intentions and security procedures. Neither scheme supports hierarchical identity-based encryption or one-to-many communication.

1.3 Our Contributions

This paper extends the concept of multi-TA IBE schemes to include hierarchical structures. For a set of uniquely named trust authorities ("US Army", "US Navy", "UK Forces", etc.) each with its own master secret used to create decryption keys for members of its hierarchy, the scheme put forward in this paper allows groups of these TA's to form coalitions without divulging secret information. Communication within a coalition is respectful of the original trust authority's hierarchy organizational structure, e.g. messages addressed to ("US Army", "1-75 Ranger Regiment", "William 'Refrigerator' Perry") may be decrypted by the same individual when sent using the multi-TA coalition scheme or sent using the "US Army" HIBE.

The same wildcard techniques from Abdalla *et al.* [**?**] may be used to transition this coalition-based HIBE to a WIBE. Under this WIBE it is possible for anyone with the public parameters to encrypt a message to any wildcard-based pattern of identities. Under any coalition configuration, all members of the "US Army" TA could be addressed as ("US Army", *, *) even if the sender was not a member of that TA. A sender possessing the public parameters of a coalition does not need to lie in the domain of trust of any TA in order to be able to send a message. A message may only be decrypted by members of the same coalition it was encrypted under, and only by those holding a decryption key matching the addressed wildcard pattern. A message may be addressed to a single specified TA or to all TA's in a coalition by wildcard in the TA-level identity (*). It is only possible to select one or all TA's within a coalition; however, selecting a subset of member TA's could be achieved by forming another smaller coalition with these TA's. Note that a message broadcast to all TA's in a coalition remains secure against non-members of the coalition.

Coalition formation is achieved through communication at the root level of trust authorities. Once a set of authorities agree to be members of a coalition they publish information to make a coalition-specific public key and for each TA to adjust their keys for the coalition. This compatibility information may then be broadcast to subordinates of the TA hierarchy enabling them to adapt their decryption keys for use within the coalition.

In this paper we offer an example instantiation based upon the Boneh-Boyen IBE [**?**] as extended to a WIBE by Abdalla *et al.* [**?**]. The system requires a bilinear map with certain properties. The Weil or Tate pairing would be a suitable example of such a map. When applied to supersingular elliptic curves, these pairings are also efficient enough for practical purposes. Included is a proof of security against selective-identity chosen-plaintext attacks in the standard model based upon the Bilinear Decisional Diffie-Hellman assumption (BDDH). In specific instantiations, including the one presented, coalitions may be dynamically reformed more efficiently than forming a coalition from scratch.

1.4 Usage Scenario

As an example of our techniques, we propose the following highly simplistic usage scenario. We suppose that individuals in a force can be identified using the hierarchical identifier ("force name", "mission name", "individual name"). For example, an individual could be identified as ("US force", "Mission 12", "John Smith"). Our scheme allows an architecture in which a central facility for each force generates a master key-pair. The master public key is publicly distributed and is intended for long-term use. We will assume that all master public keys are known by all individuals in the scenario. The master private key is used by the central facility to derive keys for each hierarchical name ("force name", "mission name", "individual name").

These individual hierarchical encryption schemes can be used as independent WIBEs. In other words, any user in possession of a force's master public key and a hierarchical identifier ("force name", "mission name", "individual name") may send a message that can only be read by that individual. Furthermore, the hierarchical identifier may include wildcard characters denoted as '*'. So, for example, a user could send a message to every individual on a mission using the hierarchical identifier ("force name", "mission name", *) or to an individual on any mission using the hierarchical identifier ("force name", *, "individual name").

However, our scheme allows coalitions to be formed between forces. If two (or more) forces which to form a coalition, then the central facilities exchange some (public) information which then allow the central facility to send a broadcast message to every member of the force. This broadcast message allows the individuals to derive coalition decryption keys from their individual decryption keys. The individuals must be assured that the broadcast message is from their force's central facility, but the message is short and need not be confidential. These coalition keys allow individuals to decrypt messages sent to multiple forces simultaneously.

For example, for a particular mission, the US and UK force may form a coalition. This allows messages to be sent to hierarchical identifiers of the form ("force name", "mission name", "individual name") where the force name either identifies a specific force or is a wildcard. In other words, to send a message to all individuals involved in the mission, one would merely have to send the message using the hierarchical name (*, "mission name", *) using the master public keys for the US and UK forces. This message could only be decrypted by an individual on the mission who has received the (public) broadcast update from their central

facility which allows them to form the coalition key for the coalition containing the US forces and the UK forces.

Furthermore, these coalitions can be dynamic in nature. So, in the above scenario, if the German force was later included in the mission, then broadcast messages could be sent to update coalition decryption keys to allow the US, UK, and German forces to decrypt messages. The German force members would not be able to read earlier messages, which were encrypted for the US and UK forces coalition only. If the US force retired from the mission, then the coalition keys could be updated so that only the UK and German forces could decrypt messages.

2 Syntax

A Trusted Authority is the root of a domain of trust with responsibilities over the namespace of its organization. The US Army could be a TA with its root authority addressed as ("US Army") with levels of the hierarchy corresponding to successively finer groupings of members, the final level identifying a particular individual in the group specified. An example identity might then be ("US Army", "1st Infantry Division", "1st Brigade Combat Team","16th Infantry Regiment", "2nd Battalion", "Benjamin Button").

In general we will refer to a Trusted Authority TA_i as a hierarchy of identities of the form $\boldsymbol{ID} = (ID_1, ID_2, ..., ID_k)$ with the same first identity ($ID_1 = TA_i$) and maximum depth of L such that $k \leq L$. Given a population of TA's $\mathcal{U} = \{ TA_1, TA_2, ..., TA_n\}$ we define a coalition $\mathcal{C} = \{ TA_a, TA_b, ..., TA_k\} \subseteq \mathcal{U}$. We also define a pattern to be a vector of identities and wildcards, i.e. $P = (P_1, \ldots, P_k)$ where $P_i \in \{0,1\}^* \cup \{*\}$ for $1 \leq i \leq k$. We say that an identity $\boldsymbol{ID} = (ID_1, \ldots, ID_k)$ matches a pattern $P = (P_1, \ldots, P_k)$, written $\boldsymbol{ID} \in_* P$, if $P_i \in \{ID_i, *\}$ for all $1 \leq i \leq k$. A multi-hierarchy WIBE consists of the following PPT algorithms/protocols:

- `CreateTA`: This algorithm creates the master public/private keys for a trust authority with a particular name ("TA_i"). This algorithm would be run by the central facility of a particular trust authority. The master public key is publicly distributed to everyone that would potentially send messages to the TA's hierarchy or form a coalition with the trust authority. The private key must be kept secret and is only used to create decryption keys for individuals in the hierarchy. Formally the algorithm takes as input the proposed name for the trusted authority ("TA_i") and outputs a master key-pair (pk_i, sk_i), written $(pk_i, sk_i) \xleftarrow{\$} \texttt{CreateTA}(\text{"}TA_i\text{"})$.
- `SetupCoalitionBroadcast`: This algorithm allows a central facility to share the information necessary to initialize coalitions. Once a set of trust authorities agree to setup a coalition between them, then each trust authority runs this algorithm to produce the information which allows the other trust authorities in the coalition to produce coalition keys for the members of its hierarchy. For some coalition $\mathcal{C} \subseteq \mathcal{U}$ containing TA_i, trust authority TA_i uses it's secret key and the public keys of participating authorities to generate public parameters specific to each other authority. These parameters allow the other coalition members to include TA_i in a coalition. This is written as $(w_{i,j} : TA_j \in \mathcal{C} \setminus TA_i) \xleftarrow{\$} \texttt{SetupCoalitionBroadcast}(TA_i, sk_i, \mathcal{C}, \{pk_j : TA_j \in \mathcal{C}\})$, where the information $w_{i,j}$ is sent from TA_i to TA_j. These values are stored by TA_j for later use in the `SetupCoalitionKeys` algorithm.
- `SetupCoalitionKeys`: This algorithm completes the setup of the coalition $\mathcal{C} = \{ TA_a, TA_b, ..., TA_k\} \subseteq \mathcal{U}$. After every member TA_i of the coalition has provided a message $w_{i,j}$ to TA_j, trust authority TA_j uses this algorithm to combine those messages to allow creation of coalition-specific secret keys. It outputs a message v_j to be broadcast to every member of TA_j's hierarchy in order who then run the `ExtractCoalitionKey` algorithm to obtain their individual secret keys specific to coalition $\mathcal{C}$. Written, $v_j \xleftarrow{\$} \texttt{SetupCoalitionKeys}(TA_j, sk_j, \mathcal{C}, \{w_{i,j} : TA_i \in \mathcal{C} \setminus TA_j\})$.

We now describe the algorithms required by the individual users.

- `Extract`: This algorithm is used by an individual in the domain of a trust authority to generate private keys for their subordinates. This key-generating entity may be a member of any level of the hierarchy,

except for the deepest level allowed (L), including the root authority. The keys generated are specific to the TA's HIBE, although they may later be adjusted for use in a coalition environment. For entity $\boldsymbol{ID} = (ID_1, ID_2, ..., ID_k)$ extracting a private key for subordinate $\boldsymbol{ID} \| ID' = (ID_1, ID_2, ..., ID_k, ID')$ the algorithm outputs $d_{\boldsymbol{ID}\|ID'} \xleftarrow{\$} \texttt{Extract}(\boldsymbol{ID}, d_{\boldsymbol{ID}}, ID')$.

- `ExtractCoalitionKey`: Users in a trust authority's hierarchy may use this algorithm to adapt their TA-specific HIBE private key for use within a coalition. To accomplish this their TA, TA_i, must provide an adjustment parameter v_i to be combined with the user's private key $d_{\boldsymbol{ID}}$. The adjustment parameter is specific to a particular coalition $\mathcal{C}$ containing TA_i, it is generated by the `SetupCoalitionKeys` algorithm. A user generates its coalition key as $c_{\boldsymbol{ID}} \leftarrow \texttt{ExtractCoalitionKey}(d_{\boldsymbol{ID}}, v_i)$. If the coalition key is undefined, then we assume that $c_{\boldsymbol{ID}} = d_{\boldsymbol{ID}}$, i.e. the decryption key for the coalition which contains only the user's trust authority.
- `Encrypt`: This algorithm can be used by an individual to encrypt a message m to any individual whose identity matches a pattern P in the trust domain of a coalition $\mathcal{C}$. This is computed as $C \xleftarrow{\$} \texttt{Encrypt}(P, m, \mathcal{C}, \{pk_i : TA_i \in \mathcal{C}\})$.
- `Decrypt`: This algorithm can be used to decrypt a ciphertext C under a coalition key $c_{\boldsymbol{ID}}$ and outputs either a message m or the error symbol $\perp$. We write this operation as $\texttt{Decrypt}(c_{\boldsymbol{ID}}, C)$. If no coalition is currently defined, then $c_{\boldsymbol{ID}} \leftarrow d_{\boldsymbol{ID}}$.

For correctness, we require that the decryption algorithm "undoes" the action of the encryption algorithm. Formally, we consider any collection of TA's $\mathcal{C} = \{TA_1, \ldots, TA_n\}$, with $(pk_i, sk_i) \xleftarrow{\$} \texttt{CreateTA}(TA_i)$, which form a coalition by computing

$$\{w_{i,j} : TA_j \in \mathcal{C} \setminus \{TA_i\}\} \xleftarrow{\$} \texttt{SetupCoalitionBroadcast}(TA_i, sk_i, \mathcal{C}, \{pk_j : TA_j \in \mathcal{C}\}) \text{ for all } TA_i \in \mathcal{C}$$
$$v_j \xleftarrow{\$} \texttt{SetupCoalitionKeys}(TA_j, sk_j, \mathcal{C}, \{w_{i,j} : TA_i \in \mathcal{C} \setminus \{TA_j\}\}) \text{ for all } TA_j \in \mathcal{C}\,.$$

Then for any identity $\boldsymbol{ID}$ and pattern P satisfying $\boldsymbol{ID} \in_* P$ and $ID_1 = TA_i \in \mathcal{C}$, we form the coalition key $c_{\boldsymbol{ID}}$ for $\boldsymbol{ID}$ by computing

$$d_{\boldsymbol{ID}} \xleftarrow{\$} \texttt{Extract}(TA_i, sk_i, \boldsymbol{ID})$$
$$c_{\boldsymbol{ID}} \xleftarrow{\$} \texttt{ExtractCoalitionKey}(d_{\boldsymbol{ID}}, v_i)\,.$$

In this situation, we demand that if $C \xleftarrow{\$} \texttt{Encrypt}(P, m, \mathcal{C}, \{pk_i : TA_i \in \mathcal{C}\})$ then $\texttt{Decrypt}(c_{\boldsymbol{ID}}, C) = m$.

3 Boneh-Boyen-Based Instantiation

We instantiate the notion of multi-TA WIBEs by extending the Boneh-Boyen WIBE [**?**]. Our scheme is utilizes two (multiplicatively-written) cyclic groups $\mathbb{G}$ and $\mathbb{G}_T$ with prime order p for which there exists a non-degenerate bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, i.e. we require that if g is a generator of $\mathbb{G}$ then $e(g, g) \neq 1$ and $e(g^a, g^b) = e(g, g)^{ab}$ for all $a, b \in \mathbb{Z}_p$. Furthermore, we assume that this map is efficiently computable. In practice, we can instantiate $\mathbb{G}$ as a prime-order subgroup of a supersingular elliptic curve group and $\mathbb{G}_T$ as a multiplicative subgroup of $\mathbb{F}_{p^r}$ for some value r, where the bilinear map is instantiated using the Weil or Tate pairing. For more information on this topic, the reader is referred to Blake, Serrousi and Smart [**?**, **?**] or Galbraith, Paterson and Smart [**?**].

Our scheme requires some general parameters which are used by all trust authorities. These include descriptions of the groups $(\mathbb{G}, \mathbb{G}_T)$, the prime p and a description of the bilinear map e. However, we require that the general parameters include some randomly generated group elements. In particular, if the maximum hierarchy depth under each trust authority is $L - 1$, then we require the group elements:

$$g_1, g_2 \xleftarrow{\$} \mathbb{G}^* \qquad\qquad u_{i,j} \xleftarrow{\$} \mathbb{G}^* \text{ for } 1 \leq i \leq L \text{ and } j \in \{0, 1\}\,.$$

It is vitally important that the group elements are generated randomly and that no extra information about the group elements are revealed. In particular, the discrete logarithms of these group elements must be

unknown. The group descriptions can be provided by standardization body in a manner similar to the NIST curves [**?**]. The group elements can be chosen using well-known techniques, for example by mapping a randomly chosen bitstring onto the curve. We let *param* denote these general parameters. Our scheme will regard identities as elements of $\mathbb{Z}_p$ and is described as follows:

- `CreateTA`(TA_i): The TA generates $\alpha_i \xleftarrow{\$} \mathbb{Z}_p$ and computes master public key $pk_i \leftarrow g_1^{\alpha_i}$. The TA's private key is defined to be $sk_i \leftarrow g_2^{\alpha_i}$.
- `Extract`$(\boldsymbol{ID}, ID', d_{\boldsymbol{ID}})$: Recall that we regard a hierarchical identity as beginning with the trusted authority's identity; hence an identity tuple $\boldsymbol{ID}$ the first element ID_1 will always equal that identity's trusted authority TA_i. If the trusted authority $(TA_i) = \boldsymbol{ID}$ wishes to derive a key for a directly subordinate identity (TA_i, ID'), the trusted authority generates the private key $d_{(TA_i,ID')} = (h, a_1, a_2)$, where

$$r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p \qquad h \leftarrow g_2^{\alpha_i}(u_{1,0} \cdot u_{1,1}^{ID_1})^{r_1}(u_{2,0} \cdot u_{2,1}^{ID_2})^{r_2} \qquad a_1 \leftarrow g_1^{r_1} \qquad a_2 \leftarrow g_1^{r_2}\,.$$

  An entity with identity $\boldsymbol{ID} = (ID_1, ID_2, ..., ID_k)$ holding private key $d_{\boldsymbol{ID}} = (h, a_1, a_2, ..., a_k)$ may derive a private key for identity $(ID_1, ID_2, ..., ID_k, ID')$. This private key is generated as $d_{(ID_1,ID_2,...,ID_k,ID')} = (h', a_1, a_2, ..., a_k, a_{k+1})$, where a_1 through a_k are retained and

$$r \xleftarrow{\$} \mathbb{Z}_p \qquad h' \leftarrow h(u_{k+1,0} \cdot u_{k+1,1}^{ID'})^r \qquad a_{k+1} \leftarrow g_{k+1}^r\,.$$

- `SetupCoalitionBroadcast`$(TA_i, \mathcal{C})$: For $TA_j \in \mathcal{C}$, TA_i randomly generates $r_j \xleftarrow{\$} \mathbb{Z}_p$ and computes

$$w_{i,j,0} \leftarrow g_2^{\alpha_i}(u_{0,0} \cdot u_{0,1}^{TA_j})^{r_j} \qquad w_{i,j,1} \leftarrow g_1^{r_j}$$

  where $TA_j \in \mathbb{Z}_p$ is the identity of TA_j. The algorithm sets $w_{i,j} = (w_{i,j,0}, w_{i,j,1})$ and outputs a list of TA/message pairs $w_{i,j}\ \forall TA_j \in \mathcal{C} \setminus TA_i$.
- `SetupCoalitionKeys`$(TA_i, sk_i, \mathcal{C}, \{w_{j,i} : TA_j \in \mathcal{C}\})$: After receiving all the messages $w_{j,i}$ from every $TA_j \in \mathcal{C}$, the algorithm outputs the TA_i-specific, coalition-specific adjustment parameter v_i as

$$v_i \leftarrow (\prod_{TA_j \in \mathcal{C} \setminus TA_i} w_{j,i,0}, \prod_{TA_j \in \mathcal{C} \setminus TA_i} w_{j,i,1})$$

- `ExtractCoalitionKey`$(\mathcal{C}, v_i, d_{\boldsymbol{ID}})$: Parse v_i as $(v_{i,0}, v_{i,1})$. A user with private key $(h, a_0, a_1, a_2, \ldots, a_\ell)$ can form a coalition key $(h', a_0', a_1, a_2, \ldots, a_\ell)$ where

$$h' \leftarrow h \cdot v_{i,0} = h \prod_{TA_j \in \mathcal{C} \setminus TA_i} w_{j,0} = g_2^{\sum \alpha_j}(u_{0,0} \cdot u_{0,1}^{TA})^{\sum r_j}$$

$$a_0' \leftarrow a_0 \cdot v_{i,1} = a_0 \prod_{TA_j \in \mathcal{C} \setminus TA_i} w_{j,1} = g_1^{r + \sum r_j}$$

- `Encrypt`$(P, m, \mathcal{C}, \{pk_j : TA_j \in \mathcal{C}\})$: Let ℓ be the depth of the pattern and $W(P)$ be the set of levels which have wildcard characters. The sender chooses $t \xleftarrow{\$} \mathbb{Z}_p$ and computes the ciphertext $C = (C_1, C_{2,1}, ..., C_{2,\ell}, C_3)$ where

$$C_1 \leftarrow g_1^t$$

$$C_{2,i} \leftarrow \begin{cases} (u_{i,0} \cdot u_{i,1}^{P_i})^t & \text{if } i \notin W(P) \\ (u_{i,0}^t, u_{i,1}^t) & \text{if } i \in W(P) \end{cases}$$

$$C_3 \leftarrow m \cdot e(\prod_{TA_j \in \mathcal{C}} pk_j, g_2)^t$$

  The ciphertext produced here is coalition-specific as it depends on the public keys of all the trust authorities in the coalition. A wildcard may be used at the TA identifier level (i.e. $P_1 = *$) to address all the trust authorities in the coalition.

– $\texttt{Decrypt}(\boldsymbol{ID}, c_{\boldsymbol{ID}}, C)$: Parse $\boldsymbol{ID}$ as $(ID_1, \ldots, ID_\ell)$, c_{ID} as $(h, a_1, \ldots, a_\ell)$, and C as $(C_1, C_{2,1}, \ldots, C_{2,\ell}, C_3)$. For each $i \in W(P)$, parse $C_{2,i}$ as $(\tilde{u}_{i,0}, \tilde{u}_{i,1})$. We recover a complete HIBE ciphertext by setting $C'_{2,i} \leftarrow C_{2,i}$ if $i \notin W(P)$, and $C'_{2,i} \leftarrow \tilde{u}_{i,0} \cdot \tilde{u}_{i,1}^{ID_i}$ if $i \in W(P)$. Recover

$$m' \leftarrow C_3 \frac{\prod_{i=1}^{\ell} e(a_i, C'_{2,i})}{e(C_1, h)}$$

and return m'.

A multi-trust-authority HIBE scheme may be designed analogously except that we remove the possibility of the pattern containing wildcards. For both schemes, we note that we can quickly reconfigure a coalition by re-using existing values of $w_{i,j}$ between trust authorities that have previously been exchanged coalition messages.

4 Security Results

4.1 Security Model

We provide a definition for the selective-identity IND-CPA security of a multi-TA WIBE. This is a weak notion of security, but serves as a basis for assessing the security of Boneh-Boyen scheme and as a basis for developing more complex security models. A selective-identity IND-CPA security model for a multi-TA HIBE can be defined in an analogous way.

The security model is presented as a game played between a probabilistic polynomial-time attacker $(\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$ and a hypothetical challenger. The sID-IND-CPA security game runs as follows:

1. The attacker runs $\mathcal{A}_0(1^k)$. The attacker outputs a challenge coalition $\mathcal{C}^*$ and a challenge patter $P^* = (P_1^*, \ldots, P_k^*)$, where $P_1^* \in \mathcal{C}^*$ or $P_1^* = *$, along with some state information *state*.
2. The challenger generates any general parameters *param* and public/private keys for each trust authority in the challenge coalition $(pk_i, sk_i) \xleftarrow{\$} \texttt{CreateTA}(\text{“}TA_i\text{”})$ (for $TA_i \in \mathcal{C}^*$).
3. The attacker runs $\mathcal{A}_1$ on the input $(param, PK, state)$ where $PK = \{pk_i : TA_i \in \mathcal{C}^*\}$. $\mathcal{A}_1$ may query the following oracles during its execution:
   – $\texttt{CreateTA}(TA)$: 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.
   – $\texttt{SetupCoalitionBroadcast}(TA_i, \mathcal{C})$: This oracle runs the $\texttt{SetupCoalitionBroadcast}$ algorithm for coalition $\mathcal{C}$ containing TA_i and returns messages $w_{i,j} : TA_j \in \mathcal{C} \setminus TA_i$.
   – $\texttt{SetupCoalitionKeys}(TA_i, \mathcal{C}, \{w_{j,i} : TA_j \in \mathcal{C} \setminus TA_i\})$: The oracle runs the $\texttt{SetupCoalitionKeys}$ algorithm assuming that message $w_{j,i}$ was sent by TA_j and returns the resulting message v_i. Note that this does not imply that all the TAs believe that they're in the same coalition or that $w_{i,j}$ has been return by the $\texttt{SetupCoalitionBroadcast}$ oracle.
   – $\texttt{Corrupt}(\boldsymbol{ID})$: 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.

   The attacker $\mathcal{A}_1$ terminates with the output of two equal-length messages (m_0, m_1) and some state information *state*.
4. The challenger randomly chooses a bit $b \xleftarrow{\$} \{0, 1\}$ and computes the ciphertext $C^* \xleftarrow{\$} \texttt{Encrypt}(P^*, m, \mathcal{C}^*, PK)$ where $PK = \{pk_i : TA_i \in \mathcal{C}^*\}$.
5. The attacker runs $\mathcal{A}_2$ in the input $(C^*, state)$. The attacker $\mathcal{A}_2$ may query the same oracle in the previous phase of the security game. The attacker terminates with the output of a bit b'.

The attacker wins the game if $b = b'$ and the attacker did not make a $\texttt{Corrupt}$ query for any $\boldsymbol{ID}$ for which there exists $\boldsymbol{ID'}$ such that $\boldsymbol{ID} \| \boldsymbol{ID'} \in_* P^*$. An attacker which did not make such an oracle query is defined to have an advantage:

$$Adv_{\mathcal{A}}^{\texttt{IND}}(k) = |\Pr[b' = 1 | b = 1] - \Pr[b' = 1 | b = 0]| .$$

4.2 The Security of the Boneh-Boyen Multi-TA WIBE

The intuition behind the multi-TA Boneh-Boyen scheme is that any coalition of trust authorities can be viewed as an extended hierarchy with a "ghost" trust authority. Each actual trust authority is represented as a first-level identity under this ghost TA and, through communication with the other trust authorities in the coalition, is able to determine a private key for their first-level identity under the ghost TA. More specifically, if we consider a coalition $\mathcal{C} = \{TA_1, \ldots, TA_n\}$ in which each TA has a private key $sk_i = g_2^{\alpha_i}$, then the ghost TA will have a private key $g_2^{\sum \alpha_i}$. Upon forming the coalition, the trust authority TA_j receives the messages

$$w_{i,j,0} \leftarrow g_2^{\alpha_i}(u_{0,0} \cdot u_{0,1}^{TA_j})^{r_i} \qquad w_{i,j,1} \leftarrow g_1^{r_i}$$

from each $TA_i \in \mathcal{C} \setminus \{TA_j\}$. This allows TA_j to form the private key

$$h \leftarrow g_2^{\sum_i \alpha_i}(u_{0,0} \cdot u_{0,1}^{TA_j})^{\sum_{i \neq j} r_i} \qquad a_1 \leftarrow g_1^{\sum_{i \neq j} r_i}$$

which is precisely the key that would be obtained if the ghost TA were to distribute a private key to the identity TA_j. The security of multi-TA scheme then essentially follows from the security of the single-TA WIBE scheme, although care must be taken to show that the broadcast messages $w_{i,j}$ and v_i do not leak information about the private keys to the attacker.

The Boneh-Boyen multi-TA WIBE scheme we have presented is secure in the sID-IND-CPA security model under the Bilinear Decision Diffie-Hellman (BDDH) problem. This can be defined as:

Definition 1. *Let p be a prime of length k, let $\mathbb{G}$ and $\mathbb{G}_T$ be cyclic groups of order p, let g a generator for $\mathbb{G}$, and let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a bilinear map. Let D_k be the distribution $\boldsymbol{x} \leftarrow (g, g^a, g^b, g^c, e(g,g)^{abc})$ where $a, b, c \xleftarrow{\$} \mathbb{Z}_p$. Let R_k be the distribution $\boldsymbol{x} \leftarrow (g, g^a, g^b, g^c, Z)$ where $a, b, c \xleftarrow{\$} \mathbb{Z}_p$ and $Z \xleftarrow{\$} \mathbb{G}_T$. An algorithm $\mathcal{A}$ has advantage $Adv_{\mathcal{A}}^{BDDH}(k)$ against the Bilinear Decision Diffie-Hellman problem if*

$$Adv_{\mathcal{A}}^{BDDH}(k) = |\Pr[\mathcal{A}(\boldsymbol{x}) = 1 \,|\, \boldsymbol{x} \xleftarrow{\$} D_k] - \Pr[\mathcal{A}(\boldsymbol{x}) = 1 \,|\, \boldsymbol{x} \xleftarrow{\$} R_k]|\,.$$

We shall assume that it is infeasible for any probabilistic polynomial-time attacker to solve the BDDH problem on the groups $(\mathbb{G}, \mathbb{G}_T)$. We may then prove the following theorems:

Theorem 1. *Suppose that there exists an attacker $\mathcal{A}$ against the selective-identity multiple TA Boneh-Boyen WIBE that runs in time t and with advantage $Adv_{\mathcal{A}}^{WIBE}(k)$, then there exists an attacker $\mathcal{B}$ against the selective-identity multiple TA Boneh-Boyen HIBE that runs in time t' and with advantage $Adv_{\mathcal{B}}^{HIBE}(k)$ where $t \approx t'$ and $Adv_{\mathcal{A}}^{WIBE}(k) = Adv_{\mathcal{B}}^{HIBE}(k)$.*

Theorem 2. *If there exists an attacker $\mathcal{A}$ against the selective-identity multi-TA IND-WID-CPA secure of the Boneh-Boyen HIBE that runs in time t, makes at most q_K queries to the `Corrupt` oracle, and has advantage $Adv_{\mathcal{A}}^{HIBE}(k)$, then there exists an algorithm $\mathcal{B}$ that solves the DBDH problem that runs in time $t' = O(t)$ and has advantage $Adv_{\mathcal{B}}^{BDDH}(k) \geq Adv_{\mathcal{A}}^{HIBE}(k)/2 - q_K/2p$.*

The proofs of these theorems are given in Appendix **??** but we can conclude the following:

Corollary 1. *If the BDDH problem is difficult, then the Boneh-Boyen multi-TA WIBE scheme is secure in the sID-IND-CPA security model.*

A Security Proofs

Theorem 3. *Suppose that there exists an attacker $\mathcal{A}$ against the sID-IND-CPA security of the multi-TA Boneh-Boyen WIBE that runs in time t and with advantage $Adv_{\mathcal{A}}^{WIBE}(k)$, then there exists an attacker $\mathcal{B}$ against the sID-IND-CPA security of the multi-TA Boneh-Boyen HIBE that runs in time t' and with advantage $Adv_{\mathcal{B}}^{HIBE}(k)$ where $t \approx t'$ and $Adv_{\mathcal{A}}^{WIBE}(k) = Adv_{\mathcal{B}}^{HIBE}(k)$.*

Proof We directly describe the algorithm $\mathcal{B}$ which breaks the HIBE using the algorithm $\mathcal{A}$ as a subroutine. Before we begin, we define some useful notation. If $P = (P_1, \ldots, P_k)$ is a pattern, then

$$W(P) = \{1 \leq i \leq k : P_i = *\} \qquad W(P_{\leq j}) = \{1 \leq i \leq \min\{j, k\} : P_i = *\}$$

The algorithm $\mathcal{B}$ runs as follows:

1. $\mathcal{B}$ runs $\mathcal{A}_0$ on the security parameter. $\mathcal{A}_0$ responds by outputting a description of the challenge coalition $TA^* = (TA_1^*, \ldots, TA_n^*)$ and the challenge pattern $P^* = (P_1^*, \ldots, P_{\ell^*}^*)$. Let π be a map which identifies the number of non-wildcard entries in the first i layers of P^*, i.e. $\pi(i) = i - |W(P_{\leq i}^*)|$. $\mathcal{B}$ outputs the challenge coalition TA^* and the challenge identity $\hat{ID}^* = (\hat{ID}_1^*, \ldots, \hat{ID}_{\pi(\ell^*)}^*)$ where $\hat{ID}_{\pi(i)}^* = P_i^*$ for $i \notin W(P^*)$.
2. $\mathcal{B}$ responds with HIBE parameters $param = (\hat{g}_1, \hat{g}_2, \hat{u}_{1,0}, \ldots, \hat{u}_{L,1})$. $\mathcal{B}$ generates WIBE parameters as follows:

$$\begin{array}{ll} (g_1, g_2) \leftarrow (\hat{g}_1, \hat{g}_2) & \\ u_{i,j} \leftarrow \hat{u}_{\pi(i),j} & \text{for } i \notin W(P^*), j \in \{0,1\} \\ u_{i,j} \leftarrow g_1^{\beta_{i,j}} & \text{for } i \in W(P^*), j \in \{0,1\} \text{ where } \beta_{i,j} \xleftarrow{\$} \mathbb{Z}_p \\ u_{i,j} \leftarrow \hat{u}_{i-|W(P^*)|,j} & \text{for } i \in \{\ell^* + 1, \ldots, L\}, j \in \{0,1\} \end{array}$$

3. $\mathcal{B}$ executes $\mathcal{A}_1$ on the public parameters $(g_1, g_2, u_{0,0}, \ldots, u_{L,1})$. $\mathcal{A}$ may make the following oracle queries:
   - `CreateTA`: $\mathcal{B}$ forwards this request to its own oracle and returns the response.
   - `SetupCoalitionBroadcast`: $\mathcal{B}$ forwards this request to its own oracle and returns the response.
   - `SetupCoalitionKeys`: $\mathcal{B}$ forwards this request to its own oracle and returns the response.
   - `Corrupt`: To an extract a decryption key for an identity $ID = (ID_1, \ldots, ID_\ell)$ which does not match the challenge pattern, $\mathcal{B}$ computes the projection of the identity onto the HIBE identity space to give a projected identity $\hat{ID} = (\hat{ID}_1 \ldots, \hat{ID}_{\hat{\ell}})$.
     - If $\ell \leq \ell^*$ then $\hat{\ell} \leftarrow \pi(\ell)$ and $\hat{ID}_{\pi(i)} \leftarrow ID_i$ for $i \notin W(P_{\leq \ell}^*)$. Since ID does not match the challenge pattern for the WIBE, we have that $\hat{ID}$ does not match the challenge identity for the HIBE. $\mathcal{B}$ queries its `Corrupt` oracle on $\hat{ID}$ and receives $(\hat{h}, \hat{a}_0, \ldots, \hat{a}_{\hat{\ell}})$ in response. $\mathcal{B}$ now "retro-fits" to find a complete key, by setting

$$\begin{array}{ll} a_0 \leftarrow \hat{a}_0 & \\ a_i \leftarrow \hat{a}_{\pi(i)} & \text{for } 1 \leq i \leq \ell \text{ and } i \notin W(P_{\leq \ell}^*) \\ a_i \leftarrow g_1^{r_i} & \text{for } 1 \leq i \leq \ell \text{ and } i \in W(P_{\leq \ell}^*) \text{ where } r_i \xleftarrow{\$} \mathbb{Z}_p \\ h \leftarrow \hat{h} \prod_{i=1, i \in W(P_{\leq \ell}^*)}^{\ell} (u_{i,0} \cdot u_{i,1}^{ID_i})^{r_i} & \end{array}$$

       and returning the key $(h, a_0, \ldots, a_\ell)$.
     - If $\ell > \ell^*$, then $\hat{\ell} = \ell - |W(P^*)|$, $\hat{ID}_{\pi(i)} \leftarrow ID_i$ for $1 \leq i \leq \ell^*$ and $i \notin W(P^*)$, and $\hat{ID}_{i-|W(P^*)|} \leftarrow ID_i$ for $\ell^* < i \leq \ell$. Since ID does not match the challenge pattern for the WIBE, we have that $\hat{ID}$ does not match the challenge identity for the HIBE. $\mathcal{B}$ queries its `Corrupt` oracle on $\hat{ID}$ and receives $(\hat{h}, \hat{a}_0, \ldots, \hat{a}_{\hat{\ell}})$ in response. $\mathcal{B}$ now "retro-fits" to find a complete key, by setting

$$\begin{array}{ll} a_0 \leftarrow \hat{a}_0 & \\ a_i \leftarrow \hat{a}_{\pi(i)} & \text{for } 1 \leq i \leq \ell^* \text{ and } i \notin W(P^*) \\ a_i \leftarrow g_1^{r_i} & \text{for } 1 \leq i \leq \ell^* \text{ and } i \in W(P^*) \text{ where } r_i \xleftarrow{\$} \mathbb{Z}_p \\ a_i \leftarrow \hat{a}_{i-|W(P^*)|} & \text{for } \ell^* < i \leq \ell \\ h \leftarrow \hat{h} \prod_{i=1, i \in W(P^*)}^{\ell^*} (u_{i,0} \cdot u_{i,1}^{ID_i})^{r_i} & \end{array}$$

       and returning the key $(h, a_0, \ldots, a_\ell)$.

   $\mathcal{A}_1$ terminates with the output of two equal-length messages (m_0, m_1).
4. $\mathcal{B}$ outputs the messages (m_0, m_1) and receives the ciphertext $(C_1^*, \hat{C}_{2,1}^*, \ldots, \hat{C}_{2,\pi(\ell^*)}^*, C_3^*)$. $\mathcal{B}$ retro-fits this to form the challenge ciphertext for $\mathcal{A}$ by setting

$$\begin{array}{ll} C_{2,i}^* \leftarrow \hat{C}_{2,\pi(i)}^* & \text{for } 1 \leq i \leq \ell^*, i \notin W(P^*) \\ C_{2,i}^* \leftarrow (C_1^{*\beta_{i,0}}, C_1^{*\beta_{i,1}}) & \text{for } 1 \leq i \leq \ell^*, i \in W(P^*) \end{array}$$

5. $\mathcal{B}$ executes $\mathcal{A}_2$ on the ciphertext $(C_1^*, C_{2,0}^*, \ldots, C_{2,\ell^*}^*, C_3^*)$ and answers all oracle queries as before. $\mathcal{A}_2$ terminates by outputting a big b' as its guess for the challenge bit b.
6. $\mathcal{B}$ outputs the bit b'.

The algorithm $\mathcal{B}$ correctly simulates the oracles to which $\mathcal{A}$ has access; furthermore, $\mathcal{B}$ wins the HIBE game if and only if $\mathcal{A}$ wins the game. Hence, we have the theorem. □

Theorem 4. *If there exists an attacker $\mathcal{A}$ against the sID-IND-CPA security of the multi-TA Boneh-Boyen HIBE that runs in time t, makes at most q_K queries to the* `Corrupt` *oracle, and has advantage $Adv_{\mathcal{A}}^{HIBE}(k)$, then there exists an algorithm $\mathcal{B}$ that solves the DBDH problem that runs in time $t' = O(t)$ and has advantage $Adv_{\mathcal{B}}^{BDDH}(k) \geq Adv_{\mathcal{A}}^{HIBE}(k) - q_K/p$.*

Proof We directly describe the algorithm $\mathcal{B}$ against the DBDH problem:

1. $\mathcal{B}$ receives the input (g, g^a, g^b, g^c, Z).
2. $\mathcal{B}$ runs $\mathcal{A}_0$ to obtain the challenge coalition $TA^* = \{TA_1^*, \ldots, TA_{n^*}^*\}$ and the challenge identity $\boldsymbol{ID}^* = (ID_1^*, \ldots, ID_{\ell^*}^*)$ under the challenge trust authority $ID_1^* = TA_1^*$. (We assume, without loss of generality, that the challenge identity is under the trusted authority TA_1^*.)
3. If $\ell^* < L$ then $\mathcal{B}$ randomly generates $ID_{\ell^*+1}^*, \ldots, ID_L^* \xleftarrow{\$} \mathbb{Z}_p$.
4. $\mathcal{B}$ computes the challenge parameters

$$\begin{gathered} g_1 \leftarrow g \qquad g_2 \leftarrow g^b \qquad k_{i,j}, \alpha_j \xleftarrow{\$} \mathbb{Z}_p^* \text{ for } 0 \leq i \leq L, j \in \{0,1\} \\ pk_1 \leftarrow g^a / g^{\sum_{j=2}^{n^*} \alpha_j} \qquad pk_j \leftarrow g^{\alpha_j} \text{ for } 2 \leq j \leq n^* \\ u_{i,0} \leftarrow g_1^{k_{i,0}} \cdot (g^a)^{-ID_i^* k_{i,1}} \qquad u_{i,1} \leftarrow (g^a)^{k_{i,1}} \qquad \text{for } 1 \leq i \leq L \end{gathered}$$

   We will set TA_1^*'s public key to be $pk_1 = g^a / g^{\sum_{j=2}^{n^*} \alpha_j}$ and TA_i^*'s public/private key pair to be $pk_i = g_1^{\alpha_i}$ and $sk_i = g_2^{\alpha_i}$ for $2 \leq i \leq n^*$.
5. $\mathcal{B}$ runs $\mathcal{A}_1$ on the public parameters $(g_1, g_2, u_{1,0}, u_{1,1}, \ldots, u_{L,0}, u_{L,1})$. If $\mathcal{A}_1$ makes an oracle queries, then $\mathcal{B}$ answers queries as follows:
   - `CreateTA`: Suppose $\mathcal{A}$ requests the public key for TA. If $TA \in TA^*$ then $\mathcal{B}$ returns the public key pk_i derived above. Otherwise, $\mathcal{B}$ generates $\alpha_{TA} \xleftarrow{\$} \mathbb{Z}_p$ and returns the public key $g_1^{\alpha_{TA}}$, while storing TA for future use.
   - `Corrupt`: Suppose $\mathcal{A}$ requests the decryption key for $\boldsymbol{ID}$. If $\boldsymbol{ID} = (TA)$, we must have $TA \notin TA^*$ for this to be a valid query; hence, $\mathcal{B}$ returns $g_2^{\alpha_{TA}}$ where α_{TA} is the value generated during the `CreateTA` query. If $\boldsymbol{ID}$ as a subordinate user of some trust authority TA, we require that $TA \neq TA_1^*$ or $\boldsymbol{ID}$ is not ancestor of $\boldsymbol{ID}^*$. Let $\boldsymbol{ID} = (ID_1, \ldots, ID_\ell)$. If $TA \neq TA_1^*$ then we may extract a decryption key using the extract algorithm and the master secret key of TA. If $TA = TA_1^*$ and there exists an index $1 \leq j \leq L$ such that $ID_j \neq ID_j^*$, then $\mathcal{B}$ generates $r_1, \ldots, r_\ell \xleftarrow{\$} \mathbb{Z}_p$ and computes the decryption key $(h, a_1, \ldots, a_j)$ for $(ID_1, \ldots, ID_j)$ as

$$\begin{aligned} h &\leftarrow g_2^{-\frac{k_{j,0}}{k_{j,1}(ID_j - ID_j^*)}} \cdot g_2^{-\sum_{j=2}^{n^*} \alpha_j} \cdot \prod_{i=1}^{j} \left(u_{i,0} \cdot u_{i,1}^{ID_i}\right)^{r_i} \\ a_i &\leftarrow g_1^{r_i} \text{ for } 1 \leq i \leq j-1 \\ a_j &\leftarrow g_2^{-\frac{1}{k_{j,1}(ID_j - ID_j^*)}} \cdot g_1^{r_j} \end{aligned}$$

     $\mathcal{B}$ computes the decryption key for $\boldsymbol{ID}$ using the key derivation algorithm and returns the result. If no such j exists then $\mathcal{B}$ aborts.
   - `SetupCoalitionBroadcast`: Suppose that $\mathcal{A}$ makes a request for TA to send messages to the coalition $\mathcal{C}$. For this to be a valid request we must have $TA \in \mathcal{C}$. If $TA \neq TA_1^*$, $\mathcal{B}$ can compute the private key

directly; hence, $\mathcal{B}$ can return the correct value using the appropriate algorithm. For $TA = TA_1^*$, $\mathcal{B}$ generates $r_1 \xleftarrow{\$} \mathbb{Z}_p$ and computes for each $TA_i \in \mathcal{C} \setminus \{TA\}$

$$w_{i,0} \leftarrow g_2^{-\frac{k_{1,0}}{k_{1,1}(TA_i - TA_1^*)}} \cdot g_2^{-\sum_{j=2}^{n^*} \beta_j} \cdot \left(u_{1,0} \cdot u_{1,1}^{TA_i}\right)^{r_1}$$

$$w_{i,1} \leftarrow g_2^{-\frac{1}{k_{1,1}(TA_i - TA_1^*)}} g_1^{r_1}$$

and sets $w_i \leftarrow (w_{i,0}, w_{i,1})$. $\mathcal{B}$ returns the list $\{w_i : TA_i \in \mathcal{C} \setminus \{TA\}\}$.

– `SetupCoalitionKeys`: The output of this oracle can be returned directly as it is independent of any private key values.

$\mathcal{A}_1$ terminates with the output of two equal-length messages (m_0, m_1).

6. $\mathcal{B}$ chooses a random bit $b \xleftarrow{\$} \{0,1\}$ and computes the ciphertext

$$C^* \leftarrow (g^c, (g^c)^{k_{1,0}}, \ldots, (g^c)^{k_{\ell^*,0}}, m_b \cdot Z)\,.$$

7. $\mathcal{B}$ executes $\mathcal{A}_2$ on the input C^* and answers all of $\mathcal{A}_2$'s oracle queries as in the previous phase. $\mathcal{A}$ terminates with the output of a bit b'.
8. If $b = b'$ then $\mathcal{B}$ outputs 1. Otherwise, outputs 0.

The `Corrupt` oracle for subordinates works perfectly providing that $\mathcal{B}$ does not abort. The simulator only aborts occurs if $\boldsymbol{ID} = (ID_1^*, \ldots, ID_\ell^*)$ for some $\ell > \ell^*$. In particular, this means that $ID_{\ell^*+1} = ID_{\ell^*+1}^*$, which occurs with probability $1/p$ as this value is information theoretically hidden from $\mathcal{A}$. Hence, the probability that this does not occur in the entire execution of $\mathcal{A}$ is q_K/p where q_K is the number of queries to the `Corrupt` oracle. To show that if the simulator doesn't abort, the `Corrupt` returns a correct key, it suffices to show that

$$sk_1 \left(u_{j,0} \cdot u_{j,1}^{ID_j}\right)^r = g_2^{\frac{-k_{j,0}}{k_{j,1}(ID_j - ID_j^*)}} \cdot g_2^{-\sum_{i=2}^{n^*} \alpha_i} \qquad \text{for} \qquad r = -\frac{b}{k_{j,1}(ID_j - ID_j^*)}\,.$$

We note that $sk_1 = g_2^{a - \sum_{i=2}^{n^*} \alpha_i}$. Therefore, we have:

$$\begin{aligned}
sk_1 \left(u_{j,0} \cdot u_{j,1}^{ID_j}\right)^r &= g_2^{a-\sum_{i=2}^{n^*} \alpha_i} \left(g^{k_{j,0}} \cdot (g^a)^{-k_{j,1} ID_j^*} \cdot (g^a)^{k_{j,1} ID_j}\right)^{-\frac{b}{k_{j,1}(ID_j - ID_j^*)}} \\
&= g^{ab} g_2^{-\sum_{i=2}^{n^*} \alpha_i} \left(g^{k_{j,0}} \cdot (g^a)^{k_{j,1}(ID_j - ID_j^*)}\right)^{-\frac{b}{k_{j,1}(ID_j - ID_j^*)}} \\
&= g^{ab} g_2^{-\sum_{i=2}^{n^*} \alpha_i} (g^b)^{\frac{-k_{j,0}}{k_{j,1}(ID_j - ID_j^*)}} g^{-ab} \\
&= g_2^{\frac{-k_{j,0}}{k_{j,1}(ID_j - ID_j^*)}} g_2^{-\sum_{i=2}^{n^*} \alpha_i}
\end{aligned}$$

Hence, $\mathcal{B}$'s simulation returns a correct decryption key. A similar calculation shows that the `SetupCoalitionBroadcast` algorithm gives correct broadcast messages for TA_1^*. All other oracles that $\mathcal{B}$ provides correctly simulate the security model for $\mathcal{A}$.

We now assume that $\mathcal{B}$ does not abort. If $Z = e(g,g)^{abc}$ then the challenge ciphertext is a correct encryption of m_b. This is because an encryption using the random value c would have

$$\begin{aligned}
C_1 &= g_1^c = g^c \\
C_{2,i} &= (u_{i,0} \cdot u_{i,1}^{ID_i^*})^c = (g^c)^{k_{i,0}} \qquad \text{for } 1 \le i \le \ell^* \\
C_3 &= m_b \cdot e(\prod_{i=1}^{n^*} pk_i, g_2)^c = m_b \cdot e(g^a, g^b)^c = m_b \cdot e(g,g)^{abc}
\end{aligned}$$

The probability that $\mathcal{B}$ outputs 1 in this situation is the probability that $b = b'$ in the sID-IND-CPA game for the attacker $\mathcal{A}$. This probability can be shown to be $(Adv_{\mathcal{A}}^{\mathtt{HIBE}}(k) - 1)/2$. If Z is random then the challenge ciphertext information theoretically hides b and so the probability that $\mathcal{B}$ outputs 1 in this situation is $1/2$. Hence, the probability that $\mathcal{B}$ wins the DBDH is at least $Adv_{\mathcal{A}}^{\mathtt{HIBE}}(k)/2 - q_K/2p$. □