Wildcarding in a Multi-TA HIBE Setting

Kent D. Boklan[*], Christopher Seaman[†]

June, 2008

WORKING DRAFT

Introduction & Motivation

Identity-based encryption is most often considered in the context of one-to-one communication within a single Trusted Authority (TA); every encrypted message is sent between two individual entities. In this paper we consider a coalition of TA's desiring secure one-to-many communication, with each TA retaining security of it's secret keys. This secrecy requirement is natural in the setting of dynamic coalition forming and dissolution.

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. In this paper, multiple recipients may decrypt the same message through the use of one or more "wildcards". A wildcard is a special character that may be used in an address in lieu of specifying a particular aspect of an identity, allowing anyone matching the non-wildcard portion to read the message.

In the single-TA case, the method of employing wildcards into hierarchical identity-based encryption structures (Abdallah et al., 2006) allows an individual at any level within one TA to send messages to entire levels within that TA. The hierarchy of a TA could be as simple as an email address, every entity under the TA (say "school.edu") has a name. Considered as an IBE setting, one might desire to send a single message to an

[*]Queens College

[†]CUNY Graduate Center

address of the form "name@school.edu", but a message pertaining to everyone at the school might be better sent to “*@school.edu” in a wildcard IBE setting.

In the multi-TA case, we could consider schools as separate Trusted Authorities. These TA’s could agree upon a protocol such as the one described in this paper to allow secure one-to-many hierarchical IBE. As such, a message addressed to the leadership of universities could be sent to "provost@*.edu" or a message for computer system administrators might be addresses “sysadmin@*.*”. 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 "*@school.edu" could be read by "alice@school.edu", "bob@school", and "eve@school.edu"; 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 TA or to all TA’s by wildcard; it is impossible to select multiple TA’s to receive a message without making the message readable across all TA’s.

In addition to one-to-many communication, this paper assumes that the coalition of TA’s may change over time and allows those changes without compromising the security of communication or any TA’s secret key. TA’s may be removed or added to the coalition with minimal configuration. Upon a change in coalition make-up the participating TA’s exchange public information, and based on non-public secret information they are able to communicate. The private keys of subordinate entities of each TA must be updated, but each TA can accomplish this through a single broadcast message exclusively readable to members of that TA. This flexible reconfiguration ability similarly allows a fixed set of TA’s in coalition to schedule secure reconfigurations with minimal communication long before any secret key has a chance to become stale or compromised.

An interesting consequence and possible drawback of using a hierarchical IBE is that messages sent to a subordinate entity may be decrypted by direct ancestors of that entity. Explicitly, a message to "bob@school.edu" could be read by the entity "school.edu", but not by "eve@school.edu" or by an entity at "university.edu". The ability to decrypt is possible because an entity’s ancestors are able to generate new secret keys for subordinates, so "school.edu" can make keys for "alice@school.edu", "bob@school.edu", ad infinitum. With the use of a centralized key distributor it may be possible to avoid this vulnerability by choosing keys in such a way as to isolate the levels of hierarchy from each other.

The Scheme

We extend the Boneh-Boyen hierarchical IBE model as adapted to include wildcards by Abdallah, et al. We assume that a group of n TA's, (TA_1, TA_2, $\ldots$, TA_n), wish to establish a coalition using a wildcard HIBE system. We further require that the TA's create a master secret such that no group of $(n-1)$ TA's may recover the secret. We assume that an individual node at the kth level of a TA's hierarchy has an identity consisting of a k-tuple of bit-string identifiers that we shall describe. We fix $k \leq L$. Each TA_i is considered to be on the first level of it's own hierarchy.

For example, TA_i may have identity ("TA Name") while a subordinate on the second tier of that TA may have identity ("TA_i Name", "2nd Tier Name_i"). We do not assume that every TA has the same depth, only that each has depth less than or equal to L. When referring to a particular identifier $(ID_{i,1}, ID_{i,2}, \ldots, ID_{i,k-1}, ID_{i,k})$ = ("TA_i Name", "2nd Tier Name", $\ldots$, "Direct Parent's Name" , "kth Tier Name") we use the notation $ID_{i,j}$ to refer to the jth identifier in the k-tuple of an entity under TA_i. We treat bit-string identifiers as names and integers interchangeably in the encryption process that we describe in this section.

Our setup requires groups $\mathbb{G}^+$, $\mathbb{Z}_q$, and $\mathbb{G}_T$, a bilinear map $e : \mathbb{G}^+ \times \mathbb{G}^+ \to \mathbb{G}_T$, a message space M, and a hash $H : \mathbb{G}_T \to M$. The security of this scheme is based on the difficulty of Bilinear Decision Diffie-Hellman problem for the map e, and also the discrete logarithm problem in $\mathbb{G}^+$. In our application we will assume $\mathbb{G}^+$ is a finite subgroup of points $E[m]$ on a supersingular elliptic curve E, that $\mathbb{G}_T$ is the finite field that is the image of $e(E[m], E[m])$, and that the message space M is $\{0,1\}^t$ for some fixed t. We assume that the TA's have agreed upon the elliptic curve over a fixed finite field and a point P of large prime order q.

To initialize our multi-TA wildcard setup, we assume that the TA's have an agreed upon order. The TA's in some manner independent of the security of the system (perhaps sequentially) choose elements of the $2L+2$-tuple:

$$\{g_1, g_2, u_{1,0}, u_{2,0}, \ldots, u_{L,0}, u_{1,1}, u_{2,1}, \ldots, u_{L,1}\}$$

where each component is a 'random' multiple of P. This information is made public. We require that discrete log problems associated with g_1 and g_2 are difficult.

In the Boneh-Boyen model, each first tier entity would get a key based upon a master secret of the form $\alpha \cdot g_2$.In our system, we do not have a single root authority and do not allow any TA or $(n-1)$ TA's to know this master secret. We require that each TA_i choose $\alpha_i, r_{i,j} \in \mathbb{Z}_q$ for $i, j \in \{1, \ldots, n\}$. Denoting each TA_j's identity as $ID_{j,1}$, TA$_i$ makes public the points:

$$\alpha_i \cdot g_1 \tag{1}$$

$$\alpha_i \cdot g_2 + r_{i,j} \cdot (u_{1,0} + ID_{j,1} \cdot u_{1,1}) \text{ for all } i \neq j \tag{2}$$

$$r_{i,j} \cdot g_1 \text{ for all } i \neq j \tag{3}$$

Each TA publishes one (1), $n - 1$ (2)'s, and $n - 1$ (3)'s. This makes for n (1)'s, $n^2 - n$ (2)'s, and $n^2 - n$(3)'s, or a total of $2n^2 - n$ public points based on secret information. As such, increasing the size of the coalition increases the number of public points each TA publishes, which in turn reduces the complexity of the discrete logarithm problem.

Each TA$_j$ derives their individual secret master key using their secret value $r_{j,j}$by adding elements of (2) and (3) as follows (note: summations run over the i index as j is fixed):

$$d_{TA_j} = (a_0, a_1) = (((\Sigma\alpha_i) \cdot g_2 + (\Sigma r_{i,j}) \cdot (u_{1,0} + ID_{j,1} \cdot u_{1,1})), (\Sigma r_{i,j}) \cdot g_1).$$

The multi-TA public key is then $(\Sigma\alpha_i \cdot g_1)$ formed by summing the public formula (1) together.

(Similar to the Boneh-Boyen architecture, each TA's secret key d_{TA_j} (as above) is related to a master secret$(\Sigma\alpha_i)\cdot g_2$ although no TA has knowledge of $(\Sigma\alpha_i) \cdot g_2$ or $\Sigma\alpha_i$ directly.

Under each TA$_j$, private keys for subordinate entities are generated from d_{TA_j}. For a subordinate on level k,the private key is given recursively from its immediate ancestor's private key $d_{\text{ancestor}} = (a_0, a_1, \ldots, a_{k-1})$. Using a randomly generated $r_k \in \mathbb{Z}_q$, the kth level subordinate's private key$d_{\text{subordinate}} = (a_0 + r_k \cdot (u_{k,0} + ID_{j,k} \cdot u_{k,1}), a_1, \ldots, a_{k-1}, r_k \cdot g_1)$.

Encryption and decryption work as in the standard wildcard hierarchical scheme. To send a message m to all level-$k$identities matching a pattern $\mathbf{P} = (P_1, \ldots, P_k)$ where each P_i is either an identifier or a wildcard[1], we say $i \in W(\mathbf{P})$ if P_i is a wildcard. (Fixed identities will be denoted $i \notin W(\mathbf{P})$.) The sender chooses a 'random' element $t \in \mathbb{Z}_q$ and outputs the ciphertext $C = (\mathbf{P}, C_1, C_{2,i}, C_3, C_{4,i,0}, C_{4,i,1})$ as i ranges as follows:

$$C_1 = t \cdot g_1$$

$$C_{2,i} = t \cdot (u_{i,0} + (P_i) \cdot u_{i,1}) \text{ for } i \notin W(\mathbf{P})$$

$$C_3 = m \otimes H(e((\Sigma\alpha_i) \cdot g_1, g_2)^t)$$

$$C_{4,i,j} = t \cdot u_{i,j} \text{ for } i \in W(\mathbf{P}) \text{ and } j \in \{0, 1\}$$

[1]For all messages sent to entities on a certain level within a fixed TA, any direct ancestor of an addressee may decrypt the message due to the derivative nature of the key generation process. Given a TA-wide central authority for key generation, it is possible to create private keys such that each subordinate entity's encrypted messages could be kept secret from their ancestor entities.

If the recipient list for the message is composed of exactly n wildcards on this k^{th} level, the cipher vector will be composed of $k + n + 2$ parts.

Decryption works by first compensating for the use of wildcards and then processing the message using the recipient's secret key. We note that anyone knowing $(\Sigma\alpha_i \cdot g_2)$ may decrypt the message by calculating $C_3 \otimes H(e(C_1, \Sigma\alpha_i \cdot g_2))$. No TA or subordinate knows this value. An intended recipient under TA$_j$ has an $ID = (ID_{j,1}, ID_{j,2}, \ldots, ID_{j,k})$ that matches the pattern $\mathbf{P} = (P_1, \ldots, P_k)$. This recipient may decrypt the message using their secret key $d_{ID} = (a_0, \ldots, a_k)$ by calculating a new C_2' element and the processing the message:

$$C'_{2,i} = C_{2,i} \text{ for } i \notin W(\mathbf{P})$$

$$C'_{2,i} = C_{4,i,0} + ID_{j,i} \cdot C_{4,i,1} \text{ for } i \in W(\mathbf{P})$$

$$m = C_3 \otimes H\left(\frac{e(C_1, a_0)}{\prod_{i=1}^{k} e(a_i, C'_{2,i})}\right)$$

1 Syntax

A multi-hierarchy WIBE consists of the following PPT algorithms/protocols:

- `Setup`$(1^k, TA)$: This algorithm is run once by a TA and outputs a master public key and master private key for that TA $(mpk_{TA}, msk_{TA}) \overset{\$}{\leftarrow}$ `Setup`$(1^k, TA)$.

  What's to prevent an attacker setting up his own TA under the name of a real coalition member and then hijacking the update coalition protocol? Are we going to assume trusted distribution of master public keys?

- `SetupCoalitionBroadcast`$(TA, msk_{TA}, (TA_1, mpk_{TA_1}), \ldots, (TA_n, mpk_{TA_n}))$: This algorithm creates a coalition between a set of TAs $C = (TA_1, \ldots, TA_n)$. This algorithm outputs a list of messages w_i to be sent to the TA TA_i $((TA_1, w_{TA_1}), \ldots, (TA_n, w_{TA_n}))$.

- `SetupCoalitionKeys`$(TA, msk_{TA}, (TA_1, mpk_{TA_1}, w_1), \ldots, (TA_n, mpk_{TA_n}, w_n))$: The algorithm completes the setup of the coalition. After every member TA_i of the coalition has provided a message w_i for TA. It outputs a message u_{TA} to be broadcast to every member of its hierarchy.

The system should be able to dynamically update the coalition. We may wish to change a coalition C into a coalition C'. We assume that members $C \cap C'$ execute the `UpdateCoalition` algorithms, while new members

$C \setminus C'$ execute the `JoinCoalition` algorithms. Excluded members $C' \setminus C$ are simply informed that they are no longer members of the coalition.

- `UpdateCoalitionBroadcast`$(TA, msk_{TA}, (TA_1, mpk_{TA_1}), \ldots, (TA_n, mpk_{TA_n}))$: This algorithm updates an existing coalition C contain TA to become a new coalition $C' = (TA, TA_1, \ldots, TA_n)$. This algorithm outputs a list of messages w_i to be sent to the TA TA_i. It should be noted that some w_i may be empty, particularly if $TA_i \in C$.

- `Join CoalitionBroadcast`$(TA, msk_{TA}, (TA_1, mpk_{TA_1}), \ldots, (TA_n, mpk_{TA_n}))$: A new authority TA which is joining an existing coalition to form a new coalition $C' = (TA, TA_1, \ldots, TA_n)$ uses this algorithm to produce a series of messages w_i to be sent to TA_i.

- `UpdateCoalitionKeys`$(TA, msk_{TA}, (TA_1, mpk_{TA_1}, w_1), \ldots, (TA_n, mpk_{TA_n}, w_n))$: The algorithm completes the updating of the coalition for existing members. After every member TA_i of the coalition has provided a (non-empty) message w_i for TA. It outputs a message u_{TA} to be broadcast to every member of its hierarchy.

- `JoinCoalitionKeys`$(TA, msk_{TA}, (TA_1, mpk_{TA_1}, w_1), \ldots, (TA_n, mpk_{TA_n}, w_n))$: This algorithm completes the joining of an existing coalition for new members. After every member TA_i of the coalition has provided a (non-empty) message w_i for TA, this algorithm outputs a message u_{TA} to be broadcast to every member of its hierarchy.

We now describe the algorithms required by the individual users.

- `Extract`$(\vec{ID}, ID', d_{\vec{ID}})$: This algorithm outputs a decryption key $d_{\vec{ID}||ID}$ for the identity $\vec{ID}||ID$. The basic level has $ID = TA$ and $d_{TA} = msk_{TA}$.

- `ExtractCoalitionKey`$((TA_1, \ldots, TA_n), u_{TA}, d_{ID})$: This algorithm outputs a user key c_{ID} for the coalition $C = \{TA, TA_1, \ldots, TA_n\}$ by combining the broadcast key u_{TA} and their decryption key d_{ID}.

- `UpdateCoalitionKey`$((TA_1, \ldots, TA_n), u_{TA}, c_{ID}, d_{ID})$: This algorithm outputs an updated user key c'_{ID} for the coalition $C = \{TA, TA_1, \ldots, TA_n\}$ by combining the broadcast key u_{TA} with the user's decryption key d_{ID} and existing coalition key c_{ID}.

- `Encrypt`$((TA_1, mpk_{TA_1}), \ldots, (TA_1, mpk_{TA_1}), P, m)$: This algorithm is used to encrypt a message m to entities satisfying the pattern P under the coalition formed by $(TA_1, \ldots, TA_n)$. It outputs a ciphertext C or the invalid symbol $\perp$.

- `Decrypt`: It does what you'd expect...

2 Security Model

The security model is parameterised by a bit b involves a PPT attacker $\mathcal{A}$ which is initially given the input 1^k and access to the following oracles:

- `CreateTA`(TA): The oracle computes $(mpk_{TA}, msk_{TA}) \overset{\$}{\leftarrow}$ `Setup`$(1^k, TA)$ for the TA identity TA and returns mpk_{TA}. This oracle can only be queried once for each identity TA.

- `SetupCoalitionBroadcast`$(TA, (TA_1, \ldots, TA_n))$: This oracle runs the `SetupCoalitionBroadcast` algorithm on the appropriate inputs and returns $(w_1, \ldots, w_n)$.

- `SetupCoalitionKeys`$(TA, (w_1, \ldots, w_n))$: This oracle can only be queried if TA has been queried to the `SetupCoalitionBroadcast` oracle with n TA's in the coalition. The oracles runs the `SetupCoalitionKeys` algorithm assuming that message w_i was sent by TA_i. Note that this does not imply that all the TAs believe that they're in the same coalition.

- `UpdateCoalition` oracles are similar to the above...

- `CorruptTA`(TA): The oracle returns msk_{TA} and records that TA is corrupt.

- `CorruptUser`$(TA, \vec{ID})$: This oracle returns $d_{\vec{ID}}$ for the identity $\vec{ID}$ under the authority TA. Note that given $d_{\vec{ID}}$ the attacker can compute any coalition key $c_{\vec{ID}}$.

- `UserDecrypt`$(TA, \vec{ID}, C^*)$: This oracle decrypts the ciphertext with the decryption key $d_{\vec{ID}}$.

- `CoalitionDecryption`$(TA, \vec{ID}, C^*)$: This oracle decrypts the ciphertext with the decryption key $c_{\vec{ID}}$.

- `Test`$(TA_1, \ldots, TA_n, P, m_0, m_1)$: This oracle takes as input two messages (m_0, m_1) of equal length. It encrypts the message m_b for the coalition using $(mpk_{TA_1}, \ldots, mpk_{TA_n})$ under the pattern P. This oracle may only be access once and outputs a ciphertext C^*. We will let C^* denote the challenge coalition $(TA_1, \ldots, TA_n)$.

The attacker terminates by outputting a bit b'. The attacker's advantage is defined to be:

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

The disallowed oracle queries: (1) a `CorruptTA` query for any TA in the test coalition , (2) a `CorruptUser` query for any user *ID* matching the pattern *P* under an authority *TA* in the test coalition if there has been a `SetupCoalitionKeys` or `UpdateCoalitionKeys` query for the test coalition, (3) a decrypt query for C^* and any user *ID* matching the pattern *P* under an authority *TA* in the test coalition if there has been a `SetupCoalitionKeys` or `UpdateCoalitionKeys` query for the test coalition.

Proofs of Security

Charting the Course

Proof of security in two steps. We will base multi-TA WIBE security (IND-smWID-CPA) on multi-TA HIBE security (IND-smID-CPA), which in turn will be based on the Bilinear Decision Diffie-Hellman problem (BDDH). In the construction of this Boneh-Boyen-based scheme, each TA_j has a private key $d_j = (\Sigma\alpha_i \cdot g_2 + \Sigma r_{i,j}(u_{0,0} + ID_{j,1} \cdot u_{0,1}), \Sigma r_{i,j} \cdot g_1)$ with summations over $i \in \{1...n\}$. These keys are constructed by taking input from each TA_i of the form (α_i)

HIBE to WIBE

Theorem:

If the Boneh-Boyen multi-*TA* HIBE is IND-smID-CPA secure then its respective WIBE is IND-smWID-CPA secure.

Proof:

The proof will follow by contradiction. Assume an adversary $\mathcal{A}$ with an advantage in the IND-smID-CPA game for the WIBE. We will construct another adversary $\mathcal{B}$ which, using $\mathcal{A}$ as a black box, will gain an advantage in the IND-sID-CPA game for the HIBE.

Initialization:

The challenger announces to $\mathcal{B}$ a set of n *TA*'s $(TA_1, ..., TA_n)$ and a maximum hierarchy depth L. $\mathcal{B}$ begins interacting with $\mathcal{A}$ and repeats the *TA*'s and depth to $\mathcal{A}$ verbatim. $\mathcal{A}$ responds with a challenge identity $P^* = (P_1, ..., P_K)$ with $P_1 \in \{TA_1, ..., TA_n\}$ or $P_1 = " * "$. Since $\mathcal{B}$ cannot have any wildcards in his challenge identity we will take the non-wildcard portions of P^* to make a fixed identity ID^*. We set $\mathcal{B}$'s challenge identity to be $ID^* = ID^*_i = P^*_{\pi(i)}$ where the map $\pi(i) = i - |W(P^*_{\leq i})| \forall i \notin W(P^*)$, dropping the wildcard portion. $\mathcal{B}$ announces this ID^* as his choice of challenge identity.

Setup:

The challenger runs `Setup` to generate HIBE parameters $\{g_1, g_2, u_{0,0}, ..., u_{L,1}\}$. Upon receiving these parameters, $\mathcal{B}$ sets his own $\hat{u}_{i,j} = u_{i,j} \forall i \notin W(P)$ and $\hat{u}_{i,j} = g_1 \forall i \in W(P)$ and announces to $\mathcal{A}$ WIBE parameters $\{g_1, g_2, \hat{u}_{0,0}, ... \hat{u}_{L,1}\}$.

Queries:

Any valid query made by $\mathcal{A}$ must be answered by $\mathcal{B}$, possibly after consulting her own oracles. The adversaries $\mathcal{A}$ and $\mathcal{B}$ have the same oracles available: `SetupCoalitionBroadcast`, `SetupCoalitionKeys`, `CorruptTA`, and `CorruptUser`.

Since the TA's available to form coalitions are identical for $\mathcal{A}$ and $\mathcal{B}$, any queries to the `SetupCoalitionBroadcast` and `SetupCoalitionKeys` oracles made by $\mathcal{A}$ may be repeated verbatim by $\mathcal{B}$. Queries made to the `CorruptTA` oracle may be made for any $TA \neq P_1$ when $P_1 \neq " * "$, if the challenge pattern has a wildcard on the TA-level ($P_1 = " * "$) then any TA is an ancestor of the challenge recipient and no `CorruptTA` queries may be made.

Queries to the `CorruptUser` oracle may be made of any node that is not an ancestor of the challenge pattern, i.e. a user $ID = (ID_1, ..., ID_j)$ may not be corrupted if $P_i \in ID_i, " * " \forall i \leq j$. To answer a `CorruptUser` query, $\mathcal{B}$ projects the identity $ID = (ID_1, ID_2, ..., ID_j)$ from the WIBE to the HIBE as $ID' = ID_{\pi(i)}$ and queries her `CorruptUser` oracle for $d_{ID'} = (a_0, a_1, ..., a_{pi(j)})$. $\mathcal{B}$ must now fill in the missing pieces of the key d_{ID}. First $\mathcal{B}$ sets $b_i = a_{\pi^{-1}(i)} \forall i > 0$. Then $\mathcal{B}$ chooses $r_i \leftarrow \mathbb{F}_p$ randomly and sets $b_i = r_i \cdot g_1$ for the missing values of $i > 0$. Finally $\mathcal{B}$ sets the value of $b_0 = a_0 \cdot \Pi r_i \hat{(} u_{i,0} + ID_i \cdot \hat{u}_{i,1})$ and answers $\mathcal{A}$'s query with $d_{ID} = (b_0, b_1, ..., b_j)$.

Challenge:

The final oracle available to both $\mathcal{A}$ and $\mathcal{B}$ is the `Test` oracle, which takes two messages m_0 and m_1 and returns the encrypted m_b for an unknown $b \in \{0, 1\}$. $\mathcal{B}$ allows $\mathcal{A}$ to choose the two messages and passes them on to his `Test` oracle. $\mathcal{B}$ must then remap elements of the ciphertext from the HIBE setting to the WIBE setting, recall the anatomy of a ciphertext C in the HIBE:

$$C_1 = t \cdot g_1$$
$$C_{2,i} = t \cdot (u_{i,0} + (P_i) \cdot u_{i,1})$$
$$C_3 = m \cdot e((\Sigma\alpha_i) \cdot g_1, g_2)^t$$

Note that there are no $C_{4,i,j}$ elements because addresses in the HIBE do not have wildcards. The challenge pattern P^* lives in the WIBE setting and is allowed to contain wildcards, $\mathcal{B}$ must adjust the ciphertext for any wildcards present as follows:

$$C'_1 = C_1$$
$$C'_{2,i} = C_{2,\pi(i)} \text{ for all } i \notin W(P^*)$$
$$C'_3 = C_3$$
$$C'_{4,i,j} = C_1 \text{ for all } i \in W(P^*) \text{ and } j \in \{0, 1\}$$

This is a valid ciphertext because of our choice of $\hat{u}_{i,j} = g_1$ for all $i \in W(P^*)$. This means that for $i \in W(P^*)$ the value needed for $C'_{4,i,j} = t \cdot u_{i,j} = t \cdot g_1 = C_1$. $\mathcal{B}$ returns this ciphertext to $\mathcal{A}$ as response to the $\mathcal{A}$'s `Test` query. $\mathcal{A}$ responds with a guess of $c \in \{0, 1\}$ as the proper value of b, which $\mathcal{B}$ repeats as her guess. Any advantage in the IND-smWID-CPA game that $\mathcal{A}$ has is then transferred onto $\mathcal{B}$.

BDDH to HIBE

Theorem:

If the Bilinear Decisional Diffie-Hellman assumption holds then the multi-*TA* Boneh-Boyen HIBE scheme is IND-smID-CPA secure.

Proof:

We will assume the existence of an adversary $\mathcal{A}$, with non-negligible advantage in the IND-smID-CPA game to construct a new adversary $\mathcal{B}$ who, using $\mathcal{A}$ as a black box, gains a non-negligible advantage in the BDDH game. To start the BDDH game $\mathcal{B}$ is given a 5-tuple $\{g, g^a, g^b, g^c, Z\}$ with $g \in \mathbb{G}_1$ and $Z \in \mathbb{G}_2$, and must decide whether $Z = e(g, g)^{abc}$ or if $Z = e(g, g)^z$ for a random $z \in \mathbb{F}_l$.

Initialization:

$\mathcal{B}$ begins interacting with $\mathcal{A}$ announcing a set of TA's $(TA_1, ..., TA_n)$ available to form coalitions and a maximum hierarchy depth L. $\mathcal{A}$ replies with a challenge identity $ID = (ID_1, ..., ID_j)$ for some $j < L$.

Setup:

$\mathcal{B}$ must construct a multi-TA HIBE for $\mathcal{A}$ and give parameters $g_1, g_2, u_{0,0}, u_{1,0}, ..., u_{L,0}, u_{0,1}, u_{1,1}, ..., u_{L,1}$ to $\mathcal{A}$.

Reconfiguration

Should the members of the coalition of TA's change, an interesting aspect of this system is that it is quickly reconfigurable. Reconfiguration of the system in play relies on the hierarchical organization of secret keys, the ability to broadcast messages to all members of a TA. The security of reconfiguration relies on only the security assumptions made previously: difficulty of the BDDH under e and the discrete logarithm problem in $\mathbb{G}^+$.

Once a new coalition is determined, the members choose β_i and $s_{i,j}$ to replace α_i and $r_{i,j}$ and publish values of $(\beta_i \cdot g_1)$ for all i and $(\beta_i \cdot g_2 + s_{i,j} \cdot (u_{1,0} + ID_{j,1} \cdot u_{1,1}))$for all $i \neq j$. As before, each TA$_j$ may then calculate their private key based on the withheld $i = j$ value, and replace the previous secret key $(a_0, a_1) = (((\Sigma\alpha_i) \cdot g_2 + \Sigma r_{i,j} \cdot (u_{1,0} + ID_{j,1} \cdot u_{1,1})), \Sigma r_{i,j} \cdot g_1)$ with the new secret key $d_{TA_j} = (b_0, b_1) = (\Sigma\beta_i \cdot g_2 + \Sigma s_{i,j} \cdot (u_{1,0} + ID_{j,1} \cdot u_{1,1}), \Sigma s_{i,j} \cdot g_1)$ (summation runs on the i index). The public key $\Sigma\beta_i \cdot g_1$ is also calculable from the public information during this setup and should be assumed available to all TA's and subordinates. To disseminate the new private information, the TA's could calculate an adjustment term:

$$(b_0 - a_0, b_1 - a_1) = ((\Sigma\beta_i - \Sigma\alpha_i) \cdot g_2 + (\Sigma s_{i,j} - \Sigma r_{i,j}) \cdot (u_{1,0} + ID_{j,1} \cdot u_{1,1}), (\Sigma s_{i,j} - \Sigma r_{i,j}) \cdot g_1)$$

The savings in reconfiguration costs come from TA's needing to do a round of communication at the highest level and then allowing a broadcast of information to subordinates rather than being required to send a separate message to each subordinate using their unique private keys.

Properties of the Scheme

Our setup enables a group of TA's to use identity based encryption in a mixed trust situation. The "super secret" $\Sigma\alpha_i \cdot g_2$ is inseparable from the secret values $r_{,i,j}$ and α_i that the TA's do not share. In this way, no TA has enough information to decrypt messages destined for another. It should be noted that we inherit from a hierarchical cryptosystem the property that every subordinate's key is derived from its superior. From this it follows that every superior may read messages sent to any subordinates in the hierarchy,not just immediate subordinates. If there were an entity above the TA's, corresponding to the "super secret",it would be able to read all messages sent in the system; for this reason we force this value to be difficult to recover by any TA or group of TA's.

Collaborative secret creation requires revealing many pieces of information but results in a highly collusion-resistant secret. The generation process requires each TA to reveal n pieces of information which are each related to the secret α_i that the TA must not reveal. From the security of each TA's α_i, any two TA's colluding have no advantage over a single TA in recovering the unknown α_i. Because no one but the target TA has this information, bringing more TA's into an attacking coalition would bring no advantage either. This can be seen by imagining the worst case scenario, $(n-1)$ TA's attacking the secret of the lone target TA. In this case, the attacker has knowledge of all α_i's and $r_{i,j}$'s except for α_k and $r_{k,k}$ for target TA$_k$. The attackers have knowledge of $((\Sigma\alpha_i) \cdot g_2 + r_{i,k} \cdot (u_{1,0} + ID_{k,1} \cdot u_{1,1}))$ for all $i \neq k$. From their secret and public pieces of information the attackers would need to calculate $(\alpha_k \cdot g_2 + r_{k,k} \cdot (u_{1,0} + ID_{k,1} \cdot u_{1,1}))$, but they cannot because $r_{k,k}$ is secret and $alpha_k$ is occluded by the discrete logarithm problem. Similarly, the use of independent secrets in the setup ensures that no TA can read another's messages without knowledge of either $\Sigma\alpha_i) \cdot g_2$ or the target TA's individual α_i. Solving either of these problems is equivalent to solving the Bilinear Diffie-Hellman problem. In the first case the attacker must be able to separate the two terms of the published $(\alpha_i \cdot g_2 + r_{i,j} \cdot (u_{1,0} + (ID_{j,1}) \cdot u_{1,1}))$, which relies on knowledge of $r_{i,j}$'s. In the second case the attacker must recover α_i, from solving a discrete logarithm on public information such as $\alpha_i \cdot g_1$.

The parameters used in encryption and decryption in this scheme does not depend on the TA membership of the sender or recipient(s) of a message (except in the sense of recipient identity). There are no TA-specific pieces of public information necessary as is sometimes the case in identity-based encryption with multiple TA's. As such, ciphertext messages may be unencumbered by information about the sender. The system is truly "identity-based" because a ciphertext depends only on the identity of the recipient(s). One interesting question in this scenario is the security of sending a message across all TA's. Is it possible for an attacker to masquerade as an additional TA by using the public information available? It is interesting to note that this scheme allows for broadcast of information to many (or all) identities within TA's. The incorporation of wildcards into the scheme comes at the (low) cost of having more public parameters for the crypto-system.