Broadcast Encryption with Multiple Trust Authorities

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

[1] Department of Computer Science,
Queens College, City University of New York, USA
[2] Information Security Group,
Royal Holloway, University of London, UK
[3] Department of Mathematics,
Graduate Center, City University of New York, USA

Abstract. In this paper we extend the notion of hierarchical identity-based encryption with wildcards (WIBE) from the domain of a single Trusted Authority (TA) to a setting with multiple, independent Trusted Authorities each with their own WIBE. In this multi-trust-authority WIBE environment, a group of TA's may form coalitions, enabling secure communication across domains. These coalitions can be created in an *ad-hoc* fashion and membership of one coalition does not give a trust authority any advantage in decrypting a ciphertext for a different coalition. This allows the broadcast of confidential messages to large groups of users within a coalition with a single ciphertext. We provide a full syntax and security model for multi-trust-authority WIBEs, and give a constructions based on the Boneh-Boyen WIBE scheme for both passive and active attackers.

1 Introduction

Identity-based encryption [12] is a type of public-key encryption in which a user's public key is set to be equal to some bitstring which uniquely identifies them within some context. This removes the need for costly and complex public-key infrastructures. Since a user's public key can now be deduced from the user's identity, the user's private decryption key can only be produced by a trusted authority (TA) with some special "trapdoor" information. This TA will release the private key to a user in the TA's trust domain after that user has proven its identity to the TA. A hierarchical identity-based encryption (HIBE) scheme [9] is an extension of identity-based encryption in which identities are arranged in a tree structure with the trust authority at the root. Users are associated with nodes in the tree and any user that has been issued with a private key can deduce a private key for any subordinate user in the tree structure. This reduces the key management requirements for the TA in a manner similar to the use of certificate chains in a public-key infrastructure.

A WIBE scheme is a HIBE scheme that allows one-to-many communication within the domain of the trust authority [1]. In a WIBE scheme, encryption is

M. Abdalla and P.S.L.M. Barreto (Eds.): LATINCRYPT 2010, LNCS 6212, pp. 1–19, 2010.

performed with respect to a pattern of bitstrings and wildcards, and a ciphertext can only be decrypted by a user whose identity matches the pattern at every non-wildcard level. This allows broadcast encryption to large numbers of users simultaneously. However, encryption is still restricted to a single trust domain. We extend the concept of a WIBE to a situation with multiple trusted authorities. In a multiple-trust-authority WIBE (MTA-WIBE) scheme, trust authorities can communicate to form coalitions that enable broadcast encryption across multiple trust domains. This allows simple broadcast encryption to large numbers of recipients in multiple trust domains. Our scheme model has the following properties:

– Trust authorities can initially implement independent instances of the MTA-WIBE scheme (without communication with other trust authorities).
– Trust authorities form coalitions through a two-stage process in which trust authorities first exchange *public* messages and then broadcast *public* "key update" messages to users in their trust domain. These key update messages allow users to form *coalition decryption keys* (based on their existing WIBE decryption keys) and so decrypt MTA-WIBE ciphertexts.
– Encryption of a message for a specific coalition is achieved using only the public parameters of the trust authorities in the coalition and does not require the sender to obtain extra information about the coalition.
– Messages are encrypted for a given coalition and under a given pattern. The resulting ciphertext can only be decrypted by users whose identities match the pattern and whose trust authority is currently in the coalition.
– Coalitions are secure in the sense that membership of one coalition does not allow a trust authority to deduce information about a message encrypted for a coalition which does not include that trust authority.

We provide a full syntax and security model for an MTA-WIBE scheme. We give a selective-identity instantiation based on the Boneh-Boyen WIBE scheme [1,4]. We also give a generic method to transform a passively secure scheme into an actively secure scheme via a novel implementation of the Boneh-Katz transform [5]. This is the most efficient generic transform for creating IND-CCA2 secure WIBEs or MTA-WIBEs from IND-CPA WIBEs or MTA-WIBEs in the literature. (The Boneh-Katz transform's application to the construction of secure WIBEs may be of independent interest.)

Usage Scenarios. We believe that MTA-WIBEs are useful in a variety of contexts. We believe that the uses of MTA-WIBEs by organisations involved in a joint project are obvious. However, we also believe that there are number of more unusual usage scenarios. For example, suppose a number of companies produce sensors for use in an ad-hoc network (including, e.g., "IBM") and suppose that these sensors can be classified according to the function they perform (including, e.g., "climate sensor"). We assume that there is a common naming structure for these sensors (e.g. (MANUFACTURER, SENSOR TYPE, PROJECT)). If the manufacturers agree to form a coalition, then a message encrypted using the coalition parameters and the pattern (*, "climate sensor", "Project Intercept") could be decrypted by any climate sensor on Project Intercept. Alternatively, a message encrypted using the pattern ("IBM", "climate sensor", *) can only be decrypted by IBM's climate sensors. This provides a method to address all sensors within a project (for project information distribution) or to address all sensors produced by an individual manufacturer (e.g. for software patching).

Related Work. Several researchers have considered the problem of developing broadcast or multi-receiver encryption schemes based on HIBE-like encryption schemes. The idea appears to have originated in the literature with the work of Dodis and Fazio [8]; other examples were given by Abdalla *et al.* [2] and Chatterjee and Sarkar [7,10]. These schemes concentrate on either sending a message to a (small) named set of users or to all users except for a (small) named set of revoked users. None of these schemes support multiple trust authorities or the "pattern-based" encryption capability provided by a WIBE. Our work extends the concept of a "pattern-based" encryption system of Abdalla *et al.* [1] to a situation with multiple trust authorities.

Other researchers have considered the question of developing IBE systems with multiple trusted authorities. Paterson and Srinivasan [11] 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. A scheme of Boklan *et al.* [3] 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 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 trust authorities setup their master private keys simultaneously. Furthermore, every trust authority can deduce the master private key of every other trust authority. This is clearly a disadvantage in any setting where the trust authorities share anything less than complete trust in each other. Unlike our scheme, no prior scheme simultaneously supports multiple trust authorities, secure *ad-hoc* coalitions, hierarchical identity-based encryption, and one-to-many communication.

Trivial Solutions. We note that the coalition functionality is only useful for broadcast encryption across the entire coalition (i.e. when the message is to be sent to a set of users which include members of distinct organisations). Broadcast encryption to a set of users contained within a single organisation can be achieved through the use of that organisation's existing WIBE. Since organisations must communicate before setting up a coalition, one could implement a trivial solution in which the organisations generate an independent WIBE to represent the coalition. Broadcast encryption across the coalition can then be achieved using this WIBE. This scheme would offer similar functionality to our scheme but has significant disadvantages:

- A sender (in possession of the public parameters for all members of the coalition) may not be able to deduce the public parameters for the coalition. In our scheme, knowledge of the public parameters for individual trust authorities allows cross-domain broadcast encryption.
- In the trivial scheme, all users within a trust domain must obtain new coalition keys from their TA individually. This requires a large amount of secure communication between the TA and the users. Our scheme allows users to deduce coalition decryption keys from a (short) publicly-broadcast "key update" message from the TA. This simplifies the key management architecture and massively reduces the workload of the TA.

2 Multiple-Trust-Authority WIBEs

Throughout the paper we will use standard notation for algorithms and assignment – see Appendix A.1.

2.1 Syntax

A Trusted Authority is the root of a domain of trust with responsibilities over the namespace of its organization. In general we will refer to a Trusted Authority TA as a hierarchy of identities of the form $ID = (ID_0, ID_1, \ldots, ID_k)$ with the same first identity ($ID_0 = TA$) and maximum depth of L. Given a population of TA's $\mathcal{U} = \{TA_1, TA_2, \ldots, TA_n\}$ we define a coalition $\mathcal{C} = \{TA_a, TA_b, \ldots, TA_\ell\} \subseteq \mathcal{U}$. We define a pattern to be a vector of identities and wildcards, i.e. $P = (P_0, \ldots, P_k)$ where $P_i \in \{0,1\}^* \cup \{*\}$ for $1 \leq i \leq k$. We say that an identity $ID = (ID_0, \ldots, ID_k)$ matches a pattern $P = (P_0, \ldots, P_{k'})$, written $ID \in_* P$, if $k \leq k'$ and $P_i \in \{ID_i, *\}$ for all $0 \leq i \leq k$. For a pattern $P = (P_0, \ldots, P_k)$, we define $W(P)$ to be the set of wildcard levels $W(P) = \{0 \leq i \leq k : P_i = *\}$ and $\bar{W}(P)$ to be the set of non-wildcard levels $\bar{W}(P) = \{0 \leq i \leq k : P_i \neq *\}$.

A multi-trust-authority WIBE (MTA-WIBE) consists of a number PPT algorithms. The following algorithms may be used by a TA:

- `Setup`: This algorithm produces public parameters used by all trust authorities. This is written $param \xleftarrow{\$} \mathtt{Setup}(1^k)$. These public parameters are assumed to be an implicit input to all other algorithms.
- `CreateTA`: This algorithm creates the master public/private keys for a trust authority with a particular name ("TA_i"). 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 $(pk_i, sk_i) \xleftarrow{\$} \mathtt{CreateTA}(\text{"}TA_i\text{"})$.
- `CoalitionBroadcast`: Once a set of trust authorities agree to setup a coalition between them, 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 their hierarchy. For some coalition $\mathcal{C} \subseteq \mathcal{U}$ containing TA_i, trust authority TA_i uses its secret key and the public keys of participating authorities to generate public parameters specific to

every other authority. This is written as $W_i \xleftarrow{\$}$ `CoalitionBroadcast`$(TA_i, sk_i, \mathcal{C}, PK)$, where $PK = \{pk_j : TA_j \in \mathcal{C}\}$ is the set of master public keys in the coalition and $W_i = \{w_{i,j} : TA_j \in \mathcal{C} \setminus TA_i\}$ is the set of key update elements. Each $w_{i,j}$ is sent from TA_i to TA_j through a *public* channel.
- `CoalitionUpdate`: 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 (publicly) broadcast to every member of TA_j's hierarchy (who then run the `CoalitionExtract` algorithm). This is written $v_j \xleftarrow{\$}$ `CoalitionUpdate`$(TA_j, sk_j, \mathcal{C}, PK, \hat{W}_j)$ where $PK = \{pk_i : TA_i \in \mathcal{C}\}$ and $\hat{W}_j = \{w_{i,j} : TA_i \in \mathcal{C} \setminus TA_j\}$ is the coalition parameters received by TA_j.

The security of the scheme does depend upon some user running the `Setup` algorithm correctly, deleting any internally-used variables, and widely distributing the result. Everyone must trust this user has not abused its privileges. This is a common problem in situations where several users have to share common parameters. In most scenarios, simple techniques can be used to provide some measure of assurance of the security of the common parameters, e.g. independent elements may be created by hashing publicly known, random data, such as astronomical signals. These techniques are considered out of scope for this paper, although the security proofs could be trivially altered to include this phase.

We also note that our security model allows the coalition broadcast messages $w_{i,j}$ and key update message v_j to be broadcast publicly. The key update message v_j may be publicly broadcast to all users, but we require that this message is sent via an integrity protected/origin authenticated channel. The $w_{i,j}$ messages exchanged between TA's do not require integrity protection or origin authentication. We now describe the algorithms required by the individual users.

- `Extract`: This algorithm is used by an individual to generate private keys for their subordinates (entities on the level below them in the hierarchical structure). The keys generated are specific to the TA's WIBE, although they may later be adjusted for use in a coalition environment. For entity $ID = (ID_0, ID_1, \ldots, ID_k)$ extracting a private key for subordinate $ID^\dagger = (ID_0, ID_1, \ldots, ID_k, ID')$ the algorithm outputs $d_{ID^\dagger} \xleftarrow{\$}$ `Extract`(ID, d_{ID}, ID').
- `CoalitionExtract`: Users in a trust authority's hierarchy may use this algorithm to adapt their TA-specific WIBE 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_{ID}. A user generates its coalition key as $c_{ID} \leftarrow$ `CoalitionExtract`(d_{ID}, v_i).
- `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 coalition $\mathcal{C}$. This is computed as $C \xleftarrow{\$}$ `Encrypt`$(\mathcal{C}, PK, P, m)$ where $PK = \{pk_i : TA_i \in \mathcal{C}\}$. We assume that if the message is encrypted for users in a single hierarchy (i.e. $P_0 \neq *$) then the coalition parameters PK used to encrypt the message will just be the parameters of the (single) trust authority P_0 (i.e. $\mathcal{C} = \{P_0\}$

when $ID_0 = P_0$). The coalition $\mathcal{C}$ should contain the identities of multiple TAs only when a message is to be sent to users in each TA's hierarchy (i.e $|\mathcal{C}| \geq 2$ if and only if $ID_0 = *$).
- `Decrypt`: This algorithm can be used to decrypt a ciphertext C under a coalition key c_{ID} and outputs either a message m or the error symbol $\perp$. We write this operation as $\mathtt{Decrypt}(ID, c_{ID}, C)$. If no coalition is currently defined, then $c_{ID} \leftarrow d_{ID}$.

It is, of course, possible to extend the MTA-WIBE syntax so that coalition update values v_j are produced after a protocol interaction between trust authorities in the coalition, but we use the simpler broadcast case as it is sufficient for our instantiation. We require that the scheme is correct in the obvious sense that decryption "undoes" encryption for correctly generated trust authorities and coalitions.

2.2 Security Model

We provide a security model for an MTA-WIBE. We begin by defining a selective-identity sID-IND-CPA model. This is a game played between a PPT attacker $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ and a hypothetical challenger: (1) the attacker runs $(\mathcal{C}^*, P^*, \omega) \stackrel{\$}{\leftarrow} \mathcal{A}_0(1^k)$ where $\mathcal{C}^*$ is the list of TA identifiers in the challenge coalition, P^* is the challenge pattern, and ω is state information; (2) the challenger generates $param \stackrel{\$}{\leftarrow} \mathtt{Setup}(1^k)$ and $(pk_i, sk_i) \stackrel{\$}{\leftarrow} \mathtt{CreateTA}(TA_i)$ for all $TA_i \in \mathcal{C}^*$; (3) the attacker outputs a bit $b' \stackrel{\$}{\leftarrow} \mathcal{A}_1(param, PK, \omega)$ where $PK \leftarrow \{pk_i : TA_i \in \mathcal{C}^*\}$. During $\mathcal{A}_1$'s execution, it may access the following oracles:

- $\mathtt{CreateTA}(TA)$: The oracle computes $(pk, sk) \stackrel{\$}{\leftarrow} \mathtt{CreateTA}(TA)$ and returns pk. This oracle can only be queried for values of TA that do not already have an associated public key. This TA is labelled "honest". (All TA's in $\mathcal{C}^*$ are also labelled "honest".)
- $\mathtt{SubmitTA}(TA, pk)$: This oracle associates the identity TA with the public key pk. It is used to model rogue TAs. This oracle can only be queried for values of TA that do not already have an associated public key. This TA is labelled "corrupt".
- $\mathtt{CoalitionBroadcast}(TA, \mathcal{C})$: This oracle computes the coalition key update set $W \stackrel{\$}{\leftarrow} \mathtt{CoalitionBroadcast}(TA, sk, \mathcal{C}, PK)$ where TA is "honest", sk is the private key associated with TA, $\mathcal{C}$ is a coalition containing TA, and PK is the set of public keys for trust authorities in $\mathcal{C}$. The oracle returns the set $W = \{w_j : TA_j \in \mathcal{C} \setminus TA\}$.
- $\mathtt{CoalitionUpdate}(TA, \mathcal{C}, \hat{W})$: The oracle computes the adjustment parameter $v \stackrel{\$}{\leftarrow} \mathtt{CoalitionUpdate}(TA, sk, \mathcal{C}, PK, \hat{W})$ where TA is honest, sk is the private key associated with TA, $\mathcal{C}$ is a coalition containing TA, PK is the set of public keys for $\mathcal{C}$, and $\hat{W} = \{w_i : TA_i \in \mathcal{C} \setminus TA\}$ is the set of key update messages that purport to be from TA_i. The oracle returns the value v. Note that we do not require that $\hat{W}$ corresponds to the elements returned by `CoalitionBroadcast`.

– $\texttt{Corrupt}(ID)$: The oracle returns d_{ID} for the identity ID. Note that if $ID = TA$ then this method returns the private key sk associated with the trust authority TA. This oracle can only be queried for situations where TA is "honest". If $ID = TA$ then TA is labelled "corrupt".
– $\texttt{Encrypt}(m_0, m_1)$: The oracle returns $C^* \xleftarrow{\$} \texttt{Encrypt}(\mathcal{C}^*, PK, P^*, m_b)$ where PK is the set of public keys associated with trust authorities $TA \in \mathcal{C}^*$ and $b \xleftarrow{\$} \{0, 1\}$. This oracle can only be queried once and only if $|m_0| = |m_1|$.

The attacker is forbidden from corrupting an identity $ID \in_* P^*$ under a trust authority $TA \in \mathcal{C}^*$. The attacker's advantage is defined to be

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

We define extended notions of security in the usual way. The IND-CPA notion of security is identical to the sID-IND-CPA notion of security except that there is no $\mathcal{A}_0$ algorithm. The algorithm $\mathcal{A}_1$ takes 1^k as input and the encryption oracle changes so that it works as follows:

– $\texttt{Encrypt}(\mathcal{C}^*, P^*, m_0, m_1)$: The oracle returns $C^* \xleftarrow{\$} \texttt{Encrypt}(\mathcal{C}^*, PK, P^*, m_b)$ where PK is the set of public keys associated with trust authorities $TA \in \mathcal{C}^*$ and $b \xleftarrow{\$} \{0, 1\}$. This oracle can only be queried once, only if $|m_0| = |m_1|$, and only on coalitions $\mathcal{C}^*$ where every $TA \in \mathcal{C}^*$ is honest.

The IND-CCA2 notion of security is identical to the IND-CPA notion of security except that $\mathcal{A}_1$ has access to a decryption oracle:

– $\texttt{Decrypt}(\mathcal{C}, ID, C)$: This oracle checks whether the coalition key c_{ID} has been defined for the coalition $\mathcal{C}$ (via a $\texttt{CoalitionUpdate}$ oracle query). If not, the oracle returns $\perp$. Otherwise, the oracle returns $\texttt{Decrypt}(ID, c_{ID}, C)$. This oracle can only be queried on identities for which the trust authority $TA = ID_0$ is honest.

The attacker is forbidden from submitting $(\mathcal{C}^*, ID, C^*)$ to the decryption oracle for any identity $ID \in_* P^*$.

We note that this model allows for "rogue TAs" whose parameters are generated maliciously, rather than by the $\texttt{CreateTA}$ oracle, as there is no requirement that $\mathcal{C}$ contain TA identities generated by the $\texttt{CreateTA}$ oracle *except* for the coalition submitted to the $\texttt{Encrypt}$ oracle. The $\texttt{SubmitTA}$ oracle allows the attacker to define a public key for a rogue TA although this public key is only used by the $\texttt{CoalitionBroadcast}$ oracle. We also note that the inability to query an oracle to obtain the coalition key c_{ID} does not represent a weakness in the model (assuming that c_{ID} and d_{ID} are secured in a similar manner) as c_{ID} can always be formed from d_{ID} and the public value v.

3 Boneh-Boyen MTA-WIBE

We present a selective-identity IND-CPA secure MTA-WIBE based on the Boneh-Boyen MTA-WIBE. The scheme is given in Figure 1. Our scheme makes use

$\mathtt{Setup}(1^k)$:
  $g_1, g_2, u_{i,j} \xleftarrow{\$} \mathbb{G}^*$ for $0 \le i \le L$, $j \in \{0,1\}$
  $param \leftarrow (g_1, g_2, u_{0,0}, \ldots, u_{L,1})$
  Return $param$

$\mathtt{CreateTA}(TA)$:
  $\alpha \xleftarrow{\$} \mathbb{Z}_p$
  $pk \leftarrow g_1^\alpha$; $sk \leftarrow g_2^\alpha$
  Return (pk, sk)

$\mathtt{CoalitionBroadcast}(TA, sk, \mathcal{C}, PK)$:
  For each $TA_j \in \mathcal{C}$:
    $r_j \xleftarrow{\$} \mathbb{Z}_p$
    $w_{j,0} \leftarrow g_2^\alpha (u_{0,0} \cdot u_{0,1}^{TA_j})^{r_j}$
    $w_{j,1} \leftarrow g_1^{r_j}$
    $w_j \leftarrow (w_{j,0}, w_{j,1})$
  $W \leftarrow \{w_j \,:\, TA_j \in \mathcal{C} \setminus \{TA\}\}$
  Return W

$\mathtt{CoalitionUpdate}(TA, sk, \mathcal{C}, PK, \hat{W})$:
  Parse $\hat{W}$ as $\{w_j \,:\, TA_j \in \mathcal{C}\}$
  Parse w_j as $(w_{j,0}, w_{j,1})$
  $v_0 \leftarrow \prod_{TA_j \in \mathcal{C}\setminus\{TA\}} w_{j,0}$
  $v_1 \leftarrow \prod_{TA_j \in \mathcal{C}\setminus\{TA\}} w_{j,1}$
  $v \leftarrow (v_0, v_1)$
  Return v

$\mathtt{CoalitionExtract}(d_{ID}, v)$:
  Parse v as (v_0, v_1)
  Parse d_{ID} as $(h, a_0, \ldots, a_k)$
  $h' \leftarrow h \cdot v_0$
  $a_0' \leftarrow a_0 \cdot v_1$
  Return $c_{ID} \leftarrow (h', a_0', a_1, \ldots, a_k)$

$\mathtt{Extract}(ID, d_{ID}, ID')$:
  If $ID = \varepsilon$ then
    $r_0, r_1 \xleftarrow{\$} \mathbb{Z}_p$
    $h \leftarrow g_2^\alpha (u_{0,0} \cdot u_{0,1}^{ID_0})^{r_0} (u_{1,0} \cdot u_{1,1}^{ID_1})^{r_1}$
    $a_0 \leftarrow g_1^{r_0}$; $a_1 \leftarrow g_1^{r_1}$
    Return (h, a_0, a_1)
  Else
    Parse d_{ID} as $(h, a_0, \ldots, a_k)$
    $r_{k+1} \xleftarrow{\$} \mathbb{Z}_p$
    $h' \leftarrow h(u_{k+1,0} u_{k+1,1}^{ID'})^{r_{k+1}}$
    $a_{k+1} \leftarrow g_1^{r_{k+1}}$
    Return $(h', a_0, \ldots, a_{k+1})$

$\mathtt{Encrypt}(\mathcal{C}, PK, P, m)$:
  $t \xleftarrow{\$} \mathbb{Z}_p$
  $C_1 \leftarrow g_1^t$
  For each $i \in W(P)$ set $C_{2,i} \leftarrow (u_{i,0}^t, u_{i,1}^t)$
  For each $i \in \bar{W}(P)$ set $C_{2,i} \leftarrow (u_{i,0} \cdot u_{i,1}^{P_i})^t$
  $C_3 \leftarrow m \cdot e(\prod_{TA_j \in \mathcal{C}} pk_j, g_2)^t$
  Return $(C_1, C_{2,1}, \ldots, C_{2,\ell}, C_3)$

$\mathtt{Decrypt}(ID, c_{ID}, C)$:
  Parse c_{ID} as $(h, a_0, \ldots, a_\ell)$
  Parse C as $(C_1, C_{2,1}, \ldots, C_{2,\ell}, C_3)$
  For each $i \in W(P)$ then
    Parse $C_{2,i}$ as $(\tilde{u}_0, \tilde{u}_1)$
    $C'_{2,i} \leftarrow \tilde{u}_0 \cdot \tilde{u}_1^{ID_i}$
  For each $i \in \bar{W}(P)$ set $C'_{2,i} \leftarrow C_{2,i}$
  $m' \xleftarrow{\$} C_3 \cdot \prod_{i=1}^{\ell} e(a_i, C'_{2,i})/e(C_1, h)$
  Return m'

Fig. 1. The Boneh-Boyen MTA-WIBE. Recall that any identity ID has $ID_0 = TA$. The `Extract` algorithm differentiates between initial key extraction by the TA and hierarchical extraction by a user. The `Decrypt` algorithm assumes that the depth of the decryption key and the depth of the ciphertext are equal. If the depth of the decryption key is shorter than the depth of the ciphertext, then the user can extract a key of a correct length and use the decryption algorithm.

of two prime-order groups $(\mathbb{G}, \mathbb{G}_T)$ and an efficiently computable bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. We assume that the size of the prime p is determined by the security parameter k.

We prove this algorithm secure in the sID-IND-CPA security model. The intuition behind the proof is that any coalition of trust authorities can be viewed as an extended hierarchy with a "ghost" trust authority at the top level. 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 the 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.

We prove this theorem in two steps. The first step removes the wildcards to form an sID-IND-CPA secure MTA-HIBE (i.e. a WIBE scheme which doesn't support wildcards).

Theorem 1. *Suppose that there exists a PPT attacker $\mathcal{A}$ against the sID-IND-CPA security of the multi-TA Boneh-Boyen WIBE with advantage $Adv_{\mathcal{A}}^{\mathit{WIBE}}(k)$. Then there exists a PPT attacker $\mathcal{B}$ against the sID-IND-CPA security of the multi-TA Boneh-Boyen HIBE with advantage $Adv_{\mathcal{B}}^{\mathit{HIBE}}(k) = Adv_{\mathcal{A}}^{\mathit{WIBE}}(k)$.*

Theorem 1 is proven using the projection technique of Abdalla *et al.* [1]. For completeness, the proof is given in Appendix B. The more interesting step is to show that the HIBE is secure. This is shown relative to the DBDH assumption:

Definition 1. *Let $(\mathbb{G}, \mathbb{G}_T)$ be groups of cyclic groups of prime order $p(k)$ with a bilinear map e and let g be a generator of $\mathbb{G}$. Let D_k be the distribution $\boldsymbol{x} \leftarrow (g, g^a, g^b, g^c, e(g,g)^{abc})$ for $a, b, c \xleftarrow{\$} \mathbb{Z}_p$. Let R_k be the distribution $\boldsymbol{x} \leftarrow (g, g^a, g^b, g^c, Z)$ for $a, b, c \xleftarrow{\$} \mathbb{Z}_p$ and $Z \xleftarrow{\$} \mathbb{G}_T$. An algorithm $\mathcal{A}$ has advantage*

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

The DBDH assumption holds if every PPT attacker has negligible advantage.

Theorem 2. *Suppose that there exists a PPT attacker $\mathcal{A}$ against the sID-IND-CPA security of the Boneh-Boyen HIBE that makes at most q_K `Corrupt` oracle queries and has advantage $Adv_{\mathcal{A}}^{\mathit{HIBE}}(k)$. Then there exists a PPT algorithm $\mathcal{B}$ that solves the DBDH problem with advantage $Adv_{\mathcal{B}}^{\mathit{BDDH}}(k) \geq Adv_{\mathcal{A}}^{\mathit{HIBE}}(k)/2 - q_K/2p$.*

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 $ID^* = (ID_0^*, \ldots, ID_{\ell^*}^*)$ where $ID_0^* = TA_1^*$ (wlog).
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

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

5. $\mathcal{B}$ runs $\mathcal{A}_1$ on the input $(g_1, g_2, u_{0,0}, u_{0,1}, \ldots, u_{L,0}, u_{L,1}, PK)$ where $PK = (pk_1, \ldots, pk_{n^*})$. If $\mathcal{A}_1$ makes an oracle query, then $\mathcal{B}$ answers queries as follows:
   - `CreateTA`: $\mathcal{B}$ generates $\alpha_{TA} \xleftarrow{\$} \mathbb{Z}_p$ and returns the public key $g_1^{\alpha_{TA}}$, while storing α_{TA} for future use.

   Note that $\mathcal{B}$ knows the private key for all TAs except TA_1^*. Hence, we only have to show how to simulate the remaining oracles for TA_1.
   - `SubmitTA`: $\mathcal{B}$ ignores any queried made to this oracle (as the Boneh-Boyen scheme does not make use of public key values in the `CoalitionBroadcast` algorithm and the challenge coalition $\mathcal{C}^*$ must be entirely honest).
   - `Corrupt`: Suppose $\mathcal{A}$ requests the decryption key for $(TA_1^*, ID_1 \ldots, ID_\ell)$. If ID is not ancestor of $(TA_1^*, ID_1^*, \ldots, ID_L^*)$, then there exists an index $1 \le \mu \le \ell$ such that $ID_\mu \ne ID_\mu^*$. $\mathcal{B}$ generates $r_1, \ldots, r_\mu \xleftarrow{\$} \mathbb{Z}_p$ and computes the decryption key $(h, a_0, \ldots, a_\mu)$ for $(ID_0, \ldots, ID_\mu)$ as

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

   $\mathcal{B}$ computes the decryption key for ID using the key derivation algorithm and returns the result. If no such μ exists (i.e. if ID is an ancestor of $(TA_1^*, ID_1^*, \ldots, ID_L^*)$ then $\mathcal{B}$ aborts .
   - `CoalitionBroadcast`: Suppose that $\mathcal{A}$ requests that TA_1^* sends messages to the coalition $\mathcal{C}$. $\mathcal{B}$ generates $r_1 \xleftarrow{\$} \mathbb{Z}_p$ and computes for each $TA_i \in \mathcal{C} \setminus \{TA_1^*\}$

$$\begin{aligned} w_{i,0} &\leftarrow g_2^{-\frac{k_{1,0}}{k_{1,1}(TA_i - TA_1^*)}} \cdot g_2^{-\sum_{j=2}^{n^*} \alpha_j} \cdot \left(u_{1,0} \cdot u_{1,1}^{TA_i} \right)^{r_1} \\ w_{i,1} &\leftarrow g_2^{-\frac{1}{k_{1,1}(TA_i - TA_1^*)}} g_1^{r_1} \end{aligned}$$

   and sets $w_j \leftarrow (w_{j,0}, w_{j,1})$. $\mathcal{B}$ returns the list $\{w_j : TA_j \in \mathcal{C} \setminus \{TA_1^*\}\}$.
   - `CoalitionUpdate`: The output of this oracle can be returned directly as it is independent of any private key values.

- `Encrypt`: Suppose $\mathcal{A}_1$ makes the oracle query on two equal-length messages (m_0, m_1). $\mathcal{B}$ chooses a bit $b \xleftarrow{\$} \{0,1\}$ and computes the ciphertext

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

$\mathcal{A}_1$ terminates with the output of a bit b'.

6. If $b = b'$ then $\mathcal{B}$ outputs 1. Otherwise, outputs 0.

The `Corrupt` oracle works perfectly provided that $\mathcal{A}_1$ does not abort, which can only occur if $\mathcal{A}_1$ makes query on an identity ID which is not an ancestor of ID^* but is an ancestor of $(ID_0^*, \ldots, ID_L^*)$. This can only occur if $\ell > \ell^*$ and $ID_{\ell^*+1} = ID^*_{\ell^*+1}$, 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. We note that the corrupt oracle gives correct responses for queries since

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

A similar calculation shows that the `CoalitionBroadcast` algorithm gives correct broadcast messages for TA_1^*. All other oracles that $\mathcal{B}$ provides correctly simulate the security model for $\mathcal{A}$.

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

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

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. This completes the proof. □

4 Strengthened Security Results

4.1 IND-CPA Security

We may prove the security of the Boneh-Boyen scheme in the (non-selective-identity) IND-CPA model by hashing all identities before use. The proof of this fact mirrors the proof in Abdalla *et al.* [1] (in the random oracle model).

$\mathtt{Setup}'(1^k)$:
  $param \xleftarrow{\$} \mathtt{Setup}(1^k)$
  $\sigma \xleftarrow{\$} \mathcal{G}(1^k)$
  $param' \leftarrow (param, \sigma)$
  Return $param'$

$\mathtt{Encode}(P, \alpha)$:
  Parse P as $(P_0, \ldots, P_\ell)$
  For $i = 0, \ldots, \ell$, $P'_i \leftarrow P_i$
  For $i = \ell + 1, \ldots, L$, $P'_i \leftarrow$ "–"
  $P'_{L+1} \leftarrow \alpha$
  Return P'

$\mathtt{Encode}'(P, ID, \alpha)$:
  For $i = 1, \ldots, |P| - |ID|$
    If $P_{|ID|+i} \neq *$ then $ID'_i \leftarrow P_{|ID|+i}$
    If $P_{|ID|+i} = *$ then $ID'_i \leftarrow 1^k$
  For $i = 1, \ldots, L - |P|$
    $ID_{|P|-|ID|+i} \leftarrow$ "–"
  $ID_{L-|ID|+1} \leftarrow \alpha$
  Return ID'

$\mathtt{CoalitionExtract}'(d_{ID}, v)$:
  $c_{ID} \leftarrow (d_{ID}, v)$
  Return c_{ID}

$\mathtt{Encrypt}'(\mathcal{C}, PK, P, m)$:
  $(K, com, dec) \xleftarrow{\$} \mathcal{S}(1^k, \sigma)$
  $P' \leftarrow \mathtt{Encode}(P, com)$
  $m' \leftarrow (m, dec)$
  $C' \xleftarrow{\$} \mathtt{Encrypt}(\mathcal{C}, PK, P', m')$
  $\tau \leftarrow \textsc{Mac}_K(\mathcal{C} \| P \| C')$
  Return $(com, \mathcal{C}, P, C', \tau)$

$\mathtt{Decrypt}'(c_{ID}, C)$:
  Parse C as $(com, \mathcal{C}, P, C', \tau)$
  Parse c_{ID} as (d_{ID}, v)
  $ID' \leftarrow \mathtt{Encode}'(P, ID, com)$
  $d' \xleftarrow{\$} \mathtt{Extract}(ID, d_{ID}, ID')$
  $c' \leftarrow \mathtt{CoalitionUpdate}(d', v)$
  $(m, dec) \xleftarrow{\$} \mathtt{Decrypt}(c, C')$
  $K \leftarrow \mathcal{R}(\sigma, com, dec)$
  If $\textsc{Mac}_K(\mathcal{C} \| P \| C') \neq \tau$ then return $\perp$
  Else return m

Fig. 2. The Boneh-Katz transform for a MTA-WIBE. Any algorithm of Π' not explicitly defined in this figure is identical to the corresponding algorithm in Π. The $\mathtt{Encode}$ algorithm turns an ℓ level pattern into an $L + 1$ level pattern. The $\mathtt{Encode}'$ algorithm computes the extension to ID required to turn an identity which matches P into an identity which matches $\mathtt{Encode}(P, \alpha)$.

4.2 IND-CCA2 Security

We may transform an IND-CPA MTA-WIBE scheme into an IND-CCA2 MTA-WIBE scheme using the CHK transform [6] in a manner similar to that described in Abdalla *et al.* [1]. However, we will describe an alternative transformation based on the Boneh-Katz (BK) transform [5]. This gives a new and more efficient method to transform IND-CPA secure WIBEs into IND-CCA2 secure WIBEs. For the specific MTA-WIBE scheme presented in Section 3, there may be more efficient IND-CCA2 constructions; however, we believe that the BK transformation in this section is of independent interest and provides the most efficient generic currently-known method to construct IND-CCA2 WIBEs and MTA-WIBEs from IND-CPA schemes.

Boneh-Katz transforms an MTA-WIBE Π into a new MTA-WIBE Π' using a MAC algorithm $\textsc{Mac}$ and an "encapsulation" scheme $(\mathcal{G}, \mathcal{S}, \mathcal{R})$. The encapsulation scheme has a key generation algorithm $\sigma \xleftarrow{\$} \mathcal{G}(1^k)$, commitment algorithm $(K, com, dec) \xleftarrow{\$} \mathcal{S}(1^k, \sigma)$, and a decommitment algorithm $\mathcal{D}(\sigma, com, dec)$ which outputs either a bitstring K or the error symbol $\perp$. We assume that $K \in \{0, 1\}^k$.

We require that if $(K, com, dec) \xleftarrow{\$} \mathcal{S}(1^k, \sigma)$ then $\mathcal{D}(\sigma, com, dec) = K$. We also require that the scheme is hiding in the sense that all PPT attackers $\mathcal{A}$ have negligible advantage:

$$\left| \Pr\left[\mathcal{A}(1^k, \sigma, com, K_b) = b \;:\; \begin{array}{c} \sigma \xleftarrow{\$} \mathcal{G}(1^k);\ K_0 \xleftarrow{\$} \{0,1\}^k \\ (K_1, com, dec) \xleftarrow{\$} \mathcal{S}(1^k, \sigma);\ b \xleftarrow{\$} \{0,1\} \end{array} \right] - \frac{1}{2} \right|$$

We further require that the encapsulation scheme is binding in the sense that all PPT attackers $\mathcal{A}$ have negligible advantage:

$$\Pr\left[\mathcal{R}(\sigma, com, dec') \notin \{\perp, K\} \;:\; \begin{array}{c} \sigma \xleftarrow{\$} \mathcal{G}(1^k);\ (K, com, dec) \xleftarrow{\$} \mathcal{S}(1^k, \sigma) \\ dec' \xleftarrow{\$} \mathcal{A}(1^k, \sigma, K, com, dec) \end{array} \right]$$

Lastly, we assume that the decommitments dec^* are always of some fixed size (which may depend on k). The security notions for a MAC scheme are given in Appendix A.2. The transform of Π into Π' is given in Figure 2. We assume that "−" represents some fixed, publicly-known allowable identity for the CPA scheme; we will deliberately exclude "−" from the space of allowable identities in the CCA scheme.

Theorem 3. *Suppose that the Π is an IND-CPA secure MTA-WIBE,* Mac *is an unforgeable MAC scheme, and $(\mathcal{G}, \mathcal{S}, \mathcal{R})$ is a hiding and binding encapsulation algorithm. Then the MTA-WIBE Π' produced by the BK transform is IND-CCA2 secure.*

The proof strategy is similar to that of Boneh and Katz [5] but has to deal with technical details introduced by the trust authorities and the WIBE scheme. A full proof is given in Appendix C

Acknowledgements

This research was sponsored by the US Army Research Laboratory and the UK Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the US Army Research Laboratory, the US Government, the UK Ministry of Defence, or the UK Government. The US and UK Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon. The research was conducted while the second author was visiting The Graduate Center of the City University of New York. The authors would like to thank the reviewers for their comments on the paper.

References

1. Abdalla, M., Catalano, D., Dent, A.W., Malone-Lee, J., Neven, G., Smart, N.P.: Identity-based encryption gone wild. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 300–311. Springer, Heidelberg (2006)

2. Abdalla, M., Kiltz, E., Neven, G.: Generalized key delegation for hierarchical identity-based encryption. In: Biskup, J., López, J. (eds.) ESORICS 2007. LNCS, vol. 4734, pp. 139–154. Springer, Heidelberg (2007)
3. Boklan, K.D., Klagsbrun, Z., Paterson, K.G., Srinivasan, S.: Flexible and secure communications in an identity-based coalition environment. In: Proc. IEEE Military Communications Conference - MILCOM 2008 (2008)
4. Boneh, D., Boyen, X.: Efficient selective-ID secure identity based encryption without random oracles. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)
5. Boneh, D., Katz, J.: Improved efficiency for CCA-secure cryptosystems built using identity-based encryption. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 87–103. Springer, Heidelberg (2005)
6. Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext security from identity-based encryption. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 207–222. Springer, Heidelberg (2004)
7. Chatterjee, S., Sarkar, P.: Multi-receiver identity-based key encapsulation with shortened ciphertext. In: Barua, R., Lange, T. (eds.) INDOCRYPT 2006. LNCS, vol. 4329, pp. 394–408. Springer, Heidelberg (2006)
8. Dodis, Y., Fazio, N.: Public key broadcast encryption for stateless receivers. In: Feigenbaum, J. (ed.) DRM 2002. LNCS, vol. 2696, pp. 61–80. Springer, Heidelberg (2003)
9. Horwitz, J., Lynn, B.: Towards hierarchical identity-based encryption. In: Knudsen, L. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 466–481. Springer, Heidelberg (2002)
10. Park, J.H., Kim, K.T., Lee, D.H.: Cryptanalysis and improvement of a multi-receiver identity-based key encapsulation at INDOCRYPT'06. In: ASIAN ACM Symposium on Information, Computer and Communications Security – ASIA CCS 2008, pp. 373–380. ACM Press, New York (2008)
11. Paterson, K.G., Srinivasan, S.: Security and anonymity of identity-based encryption with multiple trusted authorities. In: Galbraith, S.D., Paterson, K.G. (eds.) Pairing 2008. LNCS, vol. 5209, pp. 354–375. Springer, Heidelberg (2008)
12. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)

A Standard Definitions

A.1 Algorithms and Assignment

Throughout this article, $y \leftarrow x$ denotes the assignment of the value x to the variable y and $y \xleftarrow{\$} S$ denotes the assignment of a uniform random element of the finite set S to the variable y. If $\mathcal{A}$ is a probabilistic algorithm, then $y \xleftarrow{\$} \mathcal{A}(x)$ denotes the assignment of the output of the algorithm $\mathcal{A}$ run on the input x to the variable y when $\mathcal{A}$ is computed using a fresh set of random coins. We write $y \leftarrow \mathcal{A}(x)$ if $\mathcal{A}$ is deterministic.

A.2 MAC Algorithms

A MAC algorithm is a deterministic polynomial-time algorithm $\textsc{Mac}$. It takes as input a message $m \in \{0,1\}^*$ and a symmetric key $K \in \{0,1\}^k$, and outputs a tag $\tau \leftarrow \textsc{Mac}_K(m)$. It should be infeasible for a PPT attacker $\mathcal{A}$ to win the unforgeability game: (1) the challenger generates a key $K \xleftarrow{\$} \{0,1\}^k$; (2) the attacker outputs a forgery $(m^*, \tau^*) \xleftarrow{\$} \mathcal{A}(1^k)$. During its execution the attacker can query a MAC oracle with a message m and will receive $\textsc{Mac}_K(m)$. The attacker wins if $\textsc{Mac}_K(m^*) = \tau^*$ and m^* was never queried to the MAC oracle. The attackers probability of winning is written $Adv_{\mathcal{A}}^{\mathtt{MAC}}(k)$.

B Security Proof for BB-WIBE to BB-HIBE Reduction

Proof of Theorem 1

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

$$W(P) = \{1 \le i \le k : P_i = *\} \quad \text{and} \quad W(P_{\le j}) = \{1 \le i \le \min\{j,k\} : P_i = *\}\,.$$

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. $\pi(i) = i - |W(P_{\le i}^*)|$. $\mathcal{B}$ 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:

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

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.
   - `SubmitTA`: $\mathcal{B}$ may ignore these queries as the `CoalitionBroadcast` algorithm does not depend upon individual TA's public keys.
   - `CoalitionBroadcast`: $\mathcal{B}$ forwards this request to its own oracle and returns the response.
   - `CoalitionUpdate`: $\mathcal{B}$ forwards this request to its own oracle and returns the response.

– `Corrupt`: To 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, $\hat{ID}$ 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, $\hat{ID}$ 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 $(h, a_0, \ldots, a_\ell)$.

– `Encrypt`: $\mathcal{A}$ outputs two equal-length messages (m_0, m_1). $\mathcal{B}$ queries its own encryption oracle on 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

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

$\mathcal{A}_1$ terminates with the output of a bit b'.

4. $\mathcal{B}$ 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, the theorem is proven. □

C Security Proof for the BK Transform

Proof of Theorem 3
Our proof proceeds through a series of games. Let W_i be the event that the attacker $\mathcal{A}$ outputs $b' = b$ in Game i and let starred values denote values computed during the computation of the challenge ciphertext. Let Game 1 be the normal IND-CCA2 game for $\mathcal{A}$. Hence,

$$Adv_{\mathcal{A}}^{\mathtt{CCA}}(k) = 2 \cdot |\Pr[W_1] - 1/2| .$$

Let Game 2 be the same as Game 1 except that $\mathcal{A}$ is deemed to lose if it submits a ciphertext (com^*, P, C, τ) such that the decommitment value dec' recovered during the decryption process satisfies $\mathcal{R}(\sigma, com^*, dec') \notin \{\perp, K^*\}$. It is easy to show that there exists a PPT algorithm $\mathcal{B}$ such that $|\Pr[W_1] - \Pr[W_2]| \leq Adv_{\mathcal{B}}^{\mathtt{bind}}(k)$.

Let Game 3 be identical to Game 2 except that the encryption oracle computes $m'^* \leftarrow (m_b, 0^{|dec^*|})$ rather than $m'^* \leftarrow (m_b, dec^*)$. Let E_2 be the event that $\mathcal{A}$ submits a ciphertext $(com^*, \mathcal{C}, P, C', \tau)$ to the decryption oracle with $\textsc{Mac}_{K^*}(\mathcal{C} \| P \| C') = \tau$ in Game 2. Let E_3 be the event that $\mathcal{A}$ submits a ciphertext $(com^*, \mathcal{C}, P, C', \tau)$ to the decryption oracle with $\textsc{Mac}_{K^*}(\mathcal{C} \| P \| C') = \tau$ in Game 3. There exists an algorithm $\mathcal{B}^*$ against the IND-CPA security of Π such that $|\Pr[E_3] - \Pr[E_2]| \leq Adv_{\mathcal{B}^*}^{\mathtt{CPA}}(k)$. The attacker $\mathcal{B}^*(param)$ is defined as follows:

1. $\sigma \xleftarrow{\$} \mathcal{G}(1^k)$ and $(K^*, com^*, dec^*) \xleftarrow{\$} \mathcal{S}(1^k, \sigma)$.
2. $param' \leftarrow (param, \sigma)$.
3. Run $b' \xleftarrow{\$} \mathcal{A}(param')$. Suppose $\mathcal{A}$ makes an oracle query.
   - If $\mathcal{A}$ makes a `CreateTA`, `SubmitTA`, `CoalitionBroadcast`, `Coalition Update`, `CoalitionExtract` or `Corrupt` query, then $\mathcal{B}^*$ passes the query to its own oracle and returns the result.
   - If $\mathcal{A}$ makes an encryption oracle query on the equal-length messages (m_0, m_1), the pattern P^*, and the coalition $\mathcal{C}^*$ then $\mathcal{B}^*$ computes $P'^* \leftarrow$ `Encode`(P^*, com^*), $d \xleftarrow{\$} \{0,1\}$, $m'_0 \leftarrow (m_d, dec^*)$, $m'_1 \leftarrow (m_d, 0^{|dec^*|})$, and queries its encryption oracle on (m'_0, m'_1), the pattern P'^*, and the coalition $\mathcal{C}^*$. It receives C'^* from its oracle, and computes $\tau^* \leftarrow \textsc{Mac}_{K^*}(\mathcal{C}^* \| P^* \| C'^*)$. It returns $(com^*, \mathcal{C}^*, P^*, C'^*, \tau^*)$.
   - If $\mathcal{A}$ makes a decryption query on $(com, \mathcal{C}, P, C', \tau)$ with $com \neq com^*$ for the identity ID and the coalition $\mathcal{C}$, then $\mathcal{B}^*$ computes $P' \leftarrow$ `Encode` (P, com), replaces the wildcards in P' with 1^k to form the identity ID' and requests the decryption key $d_{ID'}$ for ID'. Since $\mathcal{A}$ can only make decryption queries for coalitions for which the adjustment parameter v is known, $\mathcal{B}^*$ forms the decryption key $c' \xleftarrow{\$}$ `CoalitionExtract`$(d_{ID'}, v)$. $\mathcal{B}^*$ can use this key to decrypt C', and decrypt the rest of the ciphertext as normal.
   - If $\mathcal{A}$ makes a decryption query on $(com^*, \mathcal{C}, P, C', \tau)$, then $\mathcal{B}^*$ checks whether $\textsc{Mac}_{K^*}(\mathcal{C} \| P \| C') = \tau$. If so, $\mathcal{B}^*$ outputs 1 and terminates. Otherwise, $\mathcal{B}$ returns $\perp$ to $\mathcal{A}$.
4. $\mathcal{B}^*$ outputs 0

This attacker is legal since it only queries the decryption oracle on identities with $com \neq com^*$. If $b = 0$ then $\mathcal{B}^*$ outputs 1 whenever E_2 occurs. If $b = 1$ then $\mathcal{B}^*$ outputs 1 whenever E_3 occurs. Hence, $|Pr[E_3] - \Pr[E_2]| \leq Adv^{\mathtt{CPA}}_{\mathcal{B}^*}(k)$.

There exists an attacker $\mathcal{B}'$ such that $|\Pr[W_3|\neg E_3] - \Pr[W_2|\neg E_2]| \leq Adv^{\mathtt{CPA}}_{\mathcal{B}'}(k)$. The attacker $\mathcal{B}'(param)$ is defined as follows:

1. $\sigma \xleftarrow{\$} \mathcal{G}(1^k)$ and $(K^*, com^*, dec^*) \xleftarrow{\$} \mathcal{S}(1^k, \sigma)$.
2. $param' \leftarrow (param, \sigma)$.
3. Run $d' \xleftarrow{\$} \mathcal{A}(param')$. Suppose $\mathcal{A}$ makes an oracle query.
   - If $\mathcal{A}$ makes a `CreateTA`, `SubmitTA`, `CoalitionBroadcast`, `Coalition Update`, `CoalitionExtract` or `Corrupt` query, then $\mathcal{B}$ passes the query to its own oracle and returns the result.
   - If $\mathcal{A}$ makes an encryption oracle query on the equal-length messages (m_0, m_1) and the pattern P^*, then $\mathcal{B}$ computes $P'^* \leftarrow \mathtt{Encode}(P^*, com^*)$, $d \xleftarrow{\$} \{0,1\}$, $m_0' \leftarrow (m_d, dec^*)$, $m_1' \leftarrow (m_d, 0^{|dec^*|})$, and queries its encryption oracle on (m_0', m_1') and the pattern P'. It receives C'^* from its oracle, and computes $\tau^* \leftarrow \textsc{Mac}_{K^*}(\mathcal{C}^* \| P'^* \| C'^*)$. It returns $(com^*, \mathcal{C}^*, P^*, C'^*, \tau^*)$.
   - If $\mathcal{A}$ makes a decryption query on $(com, \mathcal{C}, P, C', \tau)$ with $com \neq com^*$ for the identity ID and the coalition $\mathcal{C}$, then $\mathcal{B}$ computes $P' \leftarrow \mathtt{Encode}$ (P, com), replaces the wildcards in P' with 1^k to form the identity ID' and requests the decryption key $d_{ID'}$ for ID'. Since $\mathcal{A}$ can only make decryption queries for coalitions for which the adjustment parameter v is known, $\mathcal{B}$ forms the decryption key $c' \xleftarrow{\$} \mathtt{CoalitionExtract}(d_{ID'}, v)$. $\mathcal{B}$ can use this key to decrypt C', and decrypt the rest of the ciphertext as normal.
   - If $\mathcal{A}$ makes a decryption query on $(com^*, \mathcal{C}, P, C', \tau)$, then $\mathcal{B}$ returns $\perp$.
4. If $d = d'$ then $\mathcal{B}'$ returns 1, else it returns 0.

This is a legal attacker and $|\Pr[W_3|\neg E_3] - \Pr[W_2|\neg E_2]| \leq Adv^{\mathtt{CPA}}_{\mathcal{B}'}(k)$. A simple probability argument can be used to show that:

$$|\Pr[W_3] - \Pr[W_2]| \leq 2 \cdot Adv^{\mathtt{CPA}}_{\mathcal{B}^*}(k) + Adv^{\mathtt{CPA}}_{\mathcal{B}'}(k) + \Pr[E_3]\,.$$

Next, let Game 4 be identical to Game 3 except that the key K^* used in the encryption algorithm (and to determine if ciphertexts should be rejected) is randomly chosen from $\{0,1\}^k$. There exists an attacker $\mathcal{B}^\dagger$ against the hiding property of the encapsulation algorithm such that $|\Pr[W_4] - \Pr[W_3]| \leq 2 \cdot Adv^{\mathtt{hide}}_{\mathcal{B}^\dagger}(k)$. Let E_4 be the event that $\mathcal{A}$ submits a ciphertext $(com^*, \mathcal{C}, P, C', \tau)$ to the decryption oracle with $\textsc{Mac}_{K^*}(\mathcal{C}\|P\|C') = \tau$ in Game 4. Again, we have $|\Pr[E_4] - \Pr[E_3]| \leq 2 \cdot Adv^{\mathtt{hide}}_{\mathcal{B}^\dagger}(k)$.

Finally, let Game 5 be identical to Game 4 except that (a) the attacker loses if it queries the decryption oracle on a ciphertext $(com^*, \mathcal{C}, P, C', \tau)$ before it queries the encryption oracle, and (b) the attacker returns $\perp$ whenever the attacker queries the decryption oracle on a ciphertext $(com^*, \mathcal{C}, P, C', \tau)$ after it queries the encryption oracle. There exists an algorithm $\mathcal{B}''$ against the

MAC algorithm such that $|\Pr[W_5] - \Pr[W_4]| \leq q_D Adv^{\mathtt{MAC}}_{\mathcal{B}''}(k) + \gamma(k)$ where $\gamma(k)$ is the maximum probability that a randomly generated com^* is any fixed binary value. As a byproduct, we also obtain $\Pr[E_4] \leq q_D Adv^{\mathtt{MAC}}_{\mathcal{B}''}(k)$.

We can show a direct reduction from Game 5 to the underlying IND-CPA security of Π. There exists an algorithm $\mathcal{B}^\sharp$ such that $2 \cdot |\Pr[W_5] - 1/2| = Adv^{\mathtt{CPA}}_{\mathcal{B}^\sharp}(k)$. This algorithm simply translates decryption oracle queries made by $\mathcal{A}$ against the MTA-WIBE scheme into translates decryption oracles made by $\mathcal{B}^\sharp$ against the tag-based encryption scheme (for ciphertexts with $com \neq com^*$) or returns $\perp$ (for ciphertexts with $com = com^*$). All decryption oracle queries made by $\mathcal{B}^\sharp$ are legal as the weak selective-tag IND-CPA security model allows for decryption oracle queries for tags $com \neq com^*$. This concludes the proof. $\square$