An earlier version appears in the Proceedings of the Cryptographers' Track at the RSA Conference 2006—CT-RSA 2006, San Jose, California, February 2006. This paper is available online at `http://www.cs.stanford.edu/~xb/ctrsa06/`.

Chosen Ciphertext Secure Public Key Threshold Encryption Without Random Oracles

Dan Boneh[1⋆], Xavier Boyen[2], and Shai Halevi[3]

[1] Stanford University, Stanford, CA, dabo@cs.stanford.edu
[2] Voltage Security, Palo Alto, CA, xb@boyen.org
[3] IBM, T.J. Watson, NY, shaih@alum.mit.edu

Abstract. We present a non-interactive chosen ciphertext secure threshold encryption system. The proof of security is set in the standard model and does not use random oracles. Our construction uses the recent identity based encryption system of Boneh and Boyen and the chosen ciphertext secure construction of Canetti, Halevi, and Katz.

1 Introduction

A threshold public key encryption system [15, 17, 14, 20] is a public key system where the private key is distributed among n decryption servers so that at least k servers are needed for decryption. In a threshold encryption system an entity, called the *combiner*, has a ciphertext C that it wishes to decrypt. The combiner sends C to the decryption servers, and receives partial decryption shares from at least k out of the n decryption servers. It then combines these k partial decryptions into a complete decryption of C. Ideally, there is no other interaction in the system, namely the servers need not talk to each other during decryption. Such threshold systems are called non-interactive. Often one requires that threshold decryption be *robust* [23, 19], namely if threshold decryption of a valid ciphertext fails, the combiner can identify the decryption servers that supplied invalid partial decryptions.

In this paper we study threshold encryption systems secure against chosen ciphertext attacks (CCA). The first such system, using random oracles, was given by Shoup and Gennaro [35]. Without random random oracles, this problem is much harder and was left as an open problem in [35]. Further work on this problem is discussed later in the introduction.

We present a very efficient non-interactive CCA threshold encryption system without random oracles. Our construction proceeds in two steps. First, we extend the CCA construction of Canetti et al. [10] to threshold systems. Second, we give a robust threshold version of a recent Identity Based Encryption (IBE) due to Boneh and Boyen [3]. We achieve robustness by adding a number of internal checks to the system. Our main construction is obtained by composing these two results. This approach was outlined in the full versions of [3] and [10] and here we flesh out the full details. In Section 6 we briefly discuss several extensions such as proactive refresh [31, 25, 18] and distributed key generation [32, 22].

⋆ Supported by NSF and the Packard Foundation.

Related work. Recall that the Cramer-Shoup system [12] and its variants [34, 29] provide efficient chosen ciphertext secure encryption without random oracles. All these systems require that the private key be used to test ciphertext validity during decryption. In a threshold environment none of the decryption servers possess the private key needed to perform this validity test. Consequently, constructing a threshold version of the Cramer-Shoup system is non-trivial. The first such construction is due to Canetti and Goldwasser [8]; other threshold versions of Cramer-Shoup are given in [1, 26].

These systems, however, are more complicated than the system in this paper: they require either a large degree of interaction between the decryption servers, or storage of a large number of pre-shared secrets. More recent constructions [30, 16, 13] are non-interactive, but are either far less efficient than the construction in this paper, or do not inherently provide CCA security [13]. Cramer, Damgård, and Ishai [11] recently showed how to make the Canetti-Goldwasser construction non-interactive without random oracles, based on a general technique for computing low degree polynomials "in the exponent", but their technique is only efficient for a small number of servers.

We refer to [35] for a comprehensive survey of the related work as well as the many applications of threshold encryption.

Our Contribution. This paper shows that CCA-secure threshold public key systems (in the standard model) are easier to derive from semantically secure Identity Based Encryption than from the Cramer-Shoup paradigm. In the non-threshold setting, the latest variant of either approach give public key systems that have similar encryption performance, whether IBE-based [6] or CS-based [29]. On the the other hand, in the threshold setting, the IBE approach appears to offer substantial benefits in terms of efficiency. The main reason is that in the IBE-to-CCA transformation from [10], the validity test performed during decryption requires only the public key. Consequently, each decryption server can test ciphertext validity on its own and only release a partial decryption of valid ciphertexts. (The more efficient transformation of Boneh and Katz [6] does not have this property, and is thus less suitable for threshold encryption.)

We extend [10] to give a generic transformation from threshold IBE to threshold public key encryption, and present a concrete construction based on a threshold version of the Boneh-Boyen IBE [3]. We add a number of internal checks to provide robustness against misbehaving decryption servers. The basic idea of this paper was originally suggested in the expanded versions of [3] and [10], but without any detail. This work gives an explicit account of the construction including all the additional checks that one has to perform.

We note that Boyen, Mei, and Waters [7] recently gave a very simple and efficient CCA2-secure key encapsulation mechanism based on the Boneh-Boyen IBE framework. Unlike the generic CHK and BK transformations which require additional ingredients, the BMW method is self-contained. Since it supports public ciphertext verification, it is also suitable for non-interactive threshold decryption.

Indeed, they adapt our present construction to realize an efficient CCA2-secure non-interactive threshold KEM that eschews the need for signatures.

2 Definitions

As a preamble to our results, we recall the definitions of threshold PKE and IBE, and secure signatures.

As usual, we say that a function $f : \mathbb{Z} \to \mathbb{R}$ is negligible if for all $c > 0$ there exists $N \in \mathbb{Z}$ such that $|f(x)| < 1/x^c$ for all $x > N$.

2.1 Threshold Public Key Encryption

We define chosen ciphertext secure (CCA2) threshold public key encryption for a static adversary. We mostly follow the notation from Shoup and Genarro [35]. A Threshold Public Key Encryption (TPKE) system consists of five algorithms.

Setup(n, k, Λ)**:** Takes as input the number of decryption servers n, a threshold k where $1 \leq k \leq n$, and a security parameter $\Lambda \in \mathbb{Z}$. It outputs a triple $(\mathsf{PK}, \mathsf{VK}, \mathbf{SK})$ where PK is called the public key, VK is called the verification key, and $\mathbf{SK} = (\mathsf{SK}_1, \ldots, \mathsf{SK}_n)$ is a vector of n private key shares. Decryption server i is given the private key share (i, SK_i) and uses it to derive a decryption share for a given ciphertext. The verification key VK is used to check validity of responses from decryption servers.

Encrypt(PK, M)**:** Takes as input a public key PK and a message M. It outputs a ciphertext.

ShareDecrypt$(\mathsf{PK}, i, \mathsf{SK}_i, C)$**:** Takes as input the public key PK, a ciphertext C, and one of the n private key shares in $\mathbf{SK}$. It outputs a decryption share $\mu = (i, \hat{\mu})$ of the enciphered message, or a special symbol $(i, \perp)$.

ShareVerify$(\mathsf{PK}, \mathsf{VK}, C, \mu)$**:** Takes as input PK, the verification key VK, a ciphertext C, and a decryption share μ. It outputs `valid` or `invalid`. When the output is `valid` we say that μ is a valid decryption share of C.

Combine$\big(\mathsf{PK}, \mathsf{VK}, C, \{\mu_1, \ldots, \mu_k\}\big)$**:** Takes as input PK, VK, a ciphertext C, and k decryption shares $\{\mu_1, \ldots, \mu_k\}$. It outputs a cleartext M or $\perp$.

Consistency Requirements. Let $(\mathsf{PK}, \mathsf{VK}, \mathbf{SK})$ be the output of *Setup*(n, k, Λ). We require the following two consistency properties:

1. For any ciphertext C, if $\mu =$ *ShareDecrypt*$(\mathsf{PK}, i, \mathsf{SK}_i, C)$ where SK_i is the i-th private key share in $\mathbf{SK}$, then *ShareVerify*$(\mathsf{PK}, \mathsf{VK}, C, \mu) =$ `valid`.
2. If C is the output of *Encrypt*(PK, M) and $S = \{\mu_1, \ldots, \mu_k\}$ is a set of decryption shares $\mu_i =$ *ShareDecrypt*$(\mathsf{PK}, i, \mathsf{SK}_i, C)$ for k distinct private keys in $\mathbf{SK}$, then we require that *Combine*$(\mathsf{PK}, \mathsf{VK}, C, S) = M$.

Security. Security of TPKE is defined using two properties: security against chosen ciphertext attacks, and consistency of decryptions.

Chosen Ciphertext Security. Security against chosen ciphertext attacks is defined using the following game between a challenger and a static adversary $\mathcal{A}$. Both are given n, k, and a security parameter $\Lambda \in \mathbb{Z}^+$ as input.

Init. The adversary outputs a set $S \subset \{1, \ldots, n\}$ of $k-1$ decryption servers to corrupt.

Setup. The challenger runs $\mathit{Setup}(n, k, \Lambda)$ to obtain a random instance $(\mathsf{PK}, \mathsf{VK}, \mathbf{SK})$ where $\mathbf{SK} = (\mathsf{SK}_1, \ldots, \mathsf{SK}_n)$. It gives the adversary PK, VK, and all (j, SK_j) for $j \in S$.

Query phase 1. The adversary adaptively issues decryption queries (C, i) where $C \in \{0,1\}^*$ and $i \in \{1, \ldots, n\}$. The challenger responds with $\mathit{ShareDecrypt}(\mathsf{PK}, i, \mathsf{SK}_i, C)$.

Challenge. The adversary outputs two messages M_0, M_1 of equal length. The challenger picks a random $b \in \{0,1\}$ and lets $C^* = \mathit{Encrypt}(\mathsf{PK}, M_b)$. It gives C^* to the adversary.

Query phase 2. The adversary issues further decryption queries (C, i), under the constraint that $C \neq C^*$. The challenger responds as in phase 1.

Guess. Algorithm $\mathcal{A}$ outputs its guess $b' \in \{0,1\}$ for b and wins the game if $b = b'$.

We define the advantage of $\mathcal{A}$ as $\mathsf{AdvCCA}_{\mathcal{A},n,k}(\Lambda) = |\Pr[b = b'] - \frac{1}{2}|$.

Decryption Consistency. Consistency of decryption is defined using the following game. The game starts with the *Init*, *Setup*, and *Query phase 1* steps as in the game above. The adversary then outputs a ciphertext C and two sets of decryption shares $S = \{\mu_1, \ldots, \mu_k\}$ and $S' = \{\mu'_1, \ldots, \mu'_k\}$ each of size k. Let VK be the verification key generated in the Setup step. The adversary wins if:

1. The shares in S and S' are valid decryption shares for C under VK;
2. S and S' each contain decryption shares from k distinct servers; and
3. $\mathit{Combine}\big(\mathsf{PK}, \mathsf{VK}, C, S\big) \neq \mathit{Combine}\big(\mathsf{PK}, \mathsf{VK}, C, S'\big)$.

We let $\mathsf{AdvCD}_{\mathcal{A},n,k}(\Lambda)$ denote the adversary's advantage in winning this game.

Definition 1. *We say that a TPKE system is secure if for any n and k where $0 < k \leq n$, and any polynomial time algorithm $\mathcal{A}$, the functions $\mathsf{AdvCCA}_{\mathcal{A},n,k}(\Lambda)$ and $\mathsf{AdvCD}_{\mathcal{A},n,k}(\Lambda)$ are negligible.*

2.2 IBE with Threshold Key Generation

Next, we define IBE with threshold key generation. Here we are only concerned with semantic security and ignore chosen ciphertext attacks. A Threshold Identity Based Encryption (TIBE) system consists of seven algorithms.

Setup(n, k, Λ)**:** Takes as input the number of decryption servers n, a threshold k where $1 \leq k \leq n$, and a security parameter $\Lambda \in \mathbb{Z}$. It outputs a triple $(\mathsf{PK}, \mathsf{VK}, \mathbf{SK})$ where PK is called the system parameters, VK is called a verification key, and $\mathbf{SK} = (\mathsf{SK}_1, \ldots, \mathsf{SK}_n)$ is a vector of master key shares analogous to the private key shares in the definition of TPKE. Decryption server i is given the master key share (i, SK_i).

ShareKeyGen$(\mathsf{PK}, i, \mathsf{SK}_i, \mathsf{ID})$**:** Takes as input the system parameters PK, an identity ID, and a master key share (i, SK_i). It outputs a private key share $\theta = (i, \hat{\theta})$ for ID.

ShareVerify$(\mathsf{PK}, \mathsf{VK}, \mathsf{ID}, \theta)$**:** Takes as input the system parameters PK, the verification key VK, an identity ID, and a private key share θ. It outputs `valid` or `invalid`.

Combine$\big(\mathsf{PK}, \mathsf{VK}, \mathsf{ID}, \{\theta_1, \ldots, \theta_k\}\big)$**:** Takes as input PK, VK, an identity ID, and k private key shares $\{\theta_1, \ldots, \theta_k\}$. It outputs a private key d_{ID} or $\perp$.

Encrypt$(\mathsf{PK}, \mathsf{ID}, M)$**:** Takes PK, an identity ID, and a message M, and outputs a ciphertext C.

ValidateCT$(\mathsf{PK}, \mathsf{ID}, C)$**:** Takes as input PK, an identity ID, and a ciphertext C. It outputs `valid` or `invalid`. If `valid` we say that C is a valid encryption under ID.

Decrypt$(\mathsf{PK}, \mathsf{ID}, d_{\mathsf{ID}}, C)$**:** Takes as input PK, ID, a private key d_{ID}, and a ciphertext C. It outputs a message M or $\perp$.

Note that, unlike the previous section, decryption is not distributed. Only key generation is distributed.

Consistency Requirements. Let $(\mathsf{PK}, \mathsf{VK}, \mathbf{SK})$ be the output of $\mathit{Setup}(n, k, \Lambda)$. We require consistency properties as for TPKE systems:

1. For any identity ID, if $\theta = \mathit{ShareKeyGen}(\mathsf{PK}, i, \mathsf{SK}_i, C)$ where SK_i is one of the private key shares in $\mathbf{SK}$, then $\mathit{ShareVerify}(\mathsf{PK}, \mathsf{VK}, \mathsf{ID}, \theta) = \texttt{valid}$.
2. For any ID, if $S = \{\theta_1, \ldots, \theta_k\}$ where $\theta_i = \mathit{ShareKeyGen}(\mathsf{PK}, i, \mathsf{SK}_i, \mathsf{ID})$ for k distinct private keys in $\mathbf{SK}$, and d_{ID} is the output of $\mathit{Combine}(\mathsf{PK}, \mathsf{VK}, \mathsf{ID}, S)$, then we require that for any M and $C = \mathit{Encrypt}(\mathsf{PK}, \mathsf{ID}, M)$ we have $\mathit{ValidateCT}(\mathsf{PK}, \mathsf{ID}, C) = \texttt{valid}$ and $\mathit{Decrypt}(\mathsf{PK}, d_{\mathsf{ID}}, C) = M$.

Security. Security of a TIBE is defined using two properties: security against chosen identity attacks and consistency of key generation. There are two ways to define chosen identity attacks against IBE schemes, depending on whether the adversary chooses the target identity adaptively (an adaptive-ID attack [5]) or selects it in advance (a selective-ID attack [9]); we only need the latter for our purposes.

Selective-ID Security. Semantic security against a selective identity attack is defined using the following game:

Init. The adversary outputs an identity ID^* that it wishes to attack and a set of $k-1$ decryption servers $S \subset \{1, \ldots, n\}$ that it wishes to corrupt.

Setup. The challenger runs $\mathit{Setup}(n, k, \Lambda)$ to obtain a random instance $(\mathsf{PK}, \mathsf{VK}, \mathbf{SK})$ where $\mathbf{SK} = (\mathsf{SK}_1, \ldots, \mathsf{SK}_n)$. It gives the adversary PK, VK, and all (j, SK_j) for $j \in S$.

Query phase 1. The adversary adaptively issues chosen identity queries (ID, i) where $\mathsf{ID} \in \{0,1\}^*$ and $i \in \{1, \ldots, n\}$. The only constraint is that $\mathsf{ID} \neq \mathsf{ID}^*$. The challenger responds with $\mathit{ShareKeyGen}(\mathsf{PK}, i, \mathsf{SK}_i, \mathsf{ID})$.

Challenge. The adversary outputs two messages M_0, M_1 of equal length. The challenger picks a random $b \in \{0,1\}$ and sets the challenge ciphertext to $C^* = \mathit{Encrypt}(\mathsf{PK}, \mathsf{ID}^*, M)$. It gives C^* to the adversary.

Query phase 2. The adversary and the challenger interact as in phase 1.

Guess. Algorithm $\mathcal{A}$ outputs its guess $b' \in \{0,1\}$ for b and wins the game if $b = b'$.

We define the advantage of $\mathcal{A}$ as $\mathsf{AdvIND\text{-}ID}_{\mathcal{A},n,k}(\Lambda) = |\Pr[b = b'] - \frac{1}{2}|$.

Key Generation Consistency. Consistency of key generation (and decryption) is defined using the following game. The game starts with the *Init*, *Setup*, and *Query phase 1* steps as in the game above. The adversary then outputs an identity ID, a ciphertext C, and two sets of private key shares $S = \{\theta_1, \ldots, \theta_k\}$ and $S' = \{\theta'_1, \ldots, \theta'_k\}$ each of size k. Let PK and VK be the system parameters and the verification key generated in the Setup step. The adversary wins if:

1. the shares in S and S' are valid private key shares for ID under VK;
2. S and S' each contain private key shares from k distinct servers;
3. C is valid for the given ID, i.e., $\mathit{ValidateCT}(\mathsf{PK}, \mathsf{ID}, C) = \mathtt{valid}$;
4. the keys $d_{\mathsf{ID}} = \mathit{Combine}(\mathsf{PK}, \mathsf{VK}, \mathsf{ID}, S)$ and $d'_{\mathsf{ID}} = \mathit{Combine}(\mathsf{PK}, \mathsf{VK}, \mathsf{ID}, S')$ are such that $\perp \neq d_{\mathsf{ID}} \neq d'_{\mathsf{ID}} \neq \perp$;
5. $\mathit{Decrypt}(\mathsf{PK}, \mathsf{ID}, d_{\mathsf{ID}}, C) \neq \mathit{Decrypt}(\mathsf{PK}, \mathsf{ID}, d'_{\mathsf{ID}}, C)$.

Let $\mathsf{AdvCD\text{-}ID}_{\mathcal{A},n,k}(\Lambda)$ be the adversary's advantage in winning the game.

Definition 2. *We say that a TIBE system is selective-ID secure if for any n, k (where $0 < k \leq n$), and any polynomial time $\mathcal{A}$, the functions* $\mathsf{AdvIND\text{-}ID}_{\mathcal{A},n,k}(\Lambda)$ *and* $\mathsf{AdvCD\text{-}ID}_{\mathcal{A},n,k}(\Lambda)$ *are negligible.*

2.3 Strong Existentially Unforgeable Signatures

A signature scheme is made up of three algorithms, *SigKeyGen*, *Sign*, and *SigVerify*, for generating a key pair, signing a message, and verifying a signature, respectively.

The standard notion of security for a signature scheme is called existential unforgeability under a chosen message attack [24]. We need a slightly stronger notion of security, called strong existential unforgeability [2]. We define strong existential unforgeability under a "one chosen message" attack using the following game between a challenger and an adversary $\mathcal{A}$:

Setup. The challenger runs algorithm $\mathit{SigKeyGen}(\Lambda)$ to obtain a public key *VerK* and a private key *SigK*. The adversary $\mathcal{A}$ is given *VerK*.

Query. The adversary $\mathcal{A}$ requests a signature on a single messages of its choice, $M \in \{0,1\}^*$, under *VerK*. The challenger responds with a signature $\sigma = \mathit{Sign}(\mathit{SigK}, M)$.

Output. The adversary $\mathcal{A}$ outputs a pair (M', σ') and wins the game if $(M', \sigma') \neq (M, \sigma)$ and $\mathit{SigVerify}(\mathit{VerK}, M', \sigma') = \mathtt{valid}$.

We define $\mathsf{AdvSig}_{\mathcal{A}}(\Lambda)$ to be the probability that $\mathcal{A}$ wins in the above game.

Definition 3. *A signature scheme is existentially unforgeable under a one chosen message attack if for any probabilistic polynomial time algorithm $\mathcal{A}$ the function $\mathsf{AdvSig}_{\mathcal{A}}(\Lambda)$ is negligible.*

Efficient constructions for such signatures schemes, without random oracles, are known using the Strong-RSA assumption [21] and the 2-Strong-Diffie-Hellman (2-SDH) assumption [4].

2.4 Bilinear Maps

We briefly review the necessary facts about bilinear groups and bilinear maps, also called pairings, using the following notation:

1. $\mathbb{G}$ and $\mathbb{G}_1$ are two (multiplicative) cyclic groups of prime order p;
2. g is a generator of $\mathbb{G}$;
3. e is a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$;
4. $\mathcal{GG}(\Lambda)$ is a bilinear Group Generator as described below.

A pairing is a map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ with the following properties [27, 28, 5]:

1. Bilinearity: for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}$, we have $e(u^a, v^b) = e(u, v)^{ab}$.
2. Non-degeneracy: $e(g, g) \neq 1$.

Algorithm $\mathcal{GG}(\Lambda)$ is a bilinear Group Generator that takes a security parameter $\Lambda \in \mathbb{Z}$ as input and outputs the description of groups $\mathbb{G}$ and $\mathbb{G}_1$ and a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ where the group operation in $\mathbb{G}$ and $\mathbb{G}_1$ as well as the map e can be computed in polynomial time in Λ. To simplify the notation we use $\mathbb{G} \xleftarrow{\mathrm{R}} \mathcal{GG}(\Lambda)$ to denote the output of a random execution of $\mathcal{GG}$ on input Λ, and posit that the output $\mathcal{GG}(\Lambda)$ contains a description of p, $\mathbb{G}$, $\mathbb{G}_1$, and e.

2.5 Bilinear Diffie-Hellman Assumption

We say that an algorithm $\mathcal{B}$ that outputs $b \in \{0, 1\}$ has advantage $\epsilon(\Lambda)$ in solving the *decision* BDH problem [27, 33, 5] for the bilinear group generator $\mathcal{GG}$ if

$$\left|\Pr\left[\mathcal{B}(\mathbb{G}, g, g^a, g^b, g^c, e(g,g)^{abc}) = 0\right] - \Pr\left[\mathcal{B}(\mathbb{G}, g, g^a, g^b, g^c, T) = 0\right]\right| \geq \epsilon(\Lambda)$$

where the probability is over the random choice of group $\mathbb{G} \xleftarrow{\mathrm{R}} \mathcal{GG}(\Lambda)$, the random choice of generator g in $\mathbb{G}$, the random choice of a, b, c in $\mathbb{Z}_p$, the random choice of $T \in \mathbb{G}_1$, and the random bits consumed by $\mathcal{B}$. We refer to the distribution on the left as $\mathcal{P}_{BDH}$, and on the right as $\mathcal{R}_{BDH}$.

Definition 4. *We say that the Decision-BDH assumption holds for $\mathcal{GG}$ if any polynomial time algorithm has negligible advantage in solving the Decision BDH problem for $\mathcal{GG}$.*

3 A Threshold Identity Based Encryption System

We start with a description of a concrete Threshold IBE (TIBE) system and prove its semantic security against selective identity attacks without random oracles. For robustness against misbehaving servers we need to add several internal checks to the scheme from [3]. In a later section we show how this construction leads to a non-interactive threshold PKE with chosen ciphertext security. The TIBE system works as follows:

Setup(n, k, Λ)**.** Run the group generator $\mathcal{GG}(\Lambda)$ to obtain a bilinear group $\mathbb{G}$ of prime order $p > n$. Select random generators g, g_2, h_1 in $\mathbb{G}$, and a random degree $k-1$ polynomial $f \in \mathbb{Z}_p[X]$. Set $\alpha = f(0) \in \mathbb{Z}_p$ and $g_1 = g^\alpha$.
The system parameters PK consist of $\mathsf{PK} = (\mathbb{G}, g, g_1, g_2, h_1)$. For $i = 1, \ldots, n$ the master key share (i, SK_i) of server i is defined as $\mathsf{SK}_i = g_2^{f(i)}$. The public verification key VK consists of the n-tuple $(g^{f(1)}, \ldots, g^{f(n)})$.

ShareKeyGen$(\mathsf{PK}, i, \mathsf{SK}_i, \mathsf{ID})$**.** Let $\mathsf{PK} = (\mathbb{G}, g, g_1, g_2, h_1)$ and pick a random $r \in \mathbb{Z}_p$. Output the private key share $\theta_i = (i, (w_{i,0}, w_{i,1}))$ calculated as

$$w_{i,0} = \mathsf{SK}_i \cdot (g_1^{\mathsf{ID}} h_1)^r \ , \qquad w_{i,1} = g^r \ .$$

ShareVerify$(\mathsf{PK}, \mathsf{VK}, \mathsf{ID}, \theta_i)$**.** To verify that θ_i is a valid private key share for identity ID, let $\mathsf{VK} = (u_1, \ldots, u_n)$ where $u_i = g^{f(i)}$, and $\theta_i = (i, (w_{i,0}, w_{i,1}))$. Output `valid` or `invalid` according to the truth of the following condition:

$$e(u_i, g_2) \cdot e(g_1^{\mathsf{ID}} h_1, w_{i,1}) = e(g, w_{i,0})$$

Combine$(\mathsf{PK}, \mathsf{VK}, \mathsf{ID}, (\theta_1, \ldots, \theta_k))$**.** If one of $\theta_1, \ldots, \theta_k$ is invalid, or if two shares θ_i and θ_j bear the same server index, then output $\perp$ and exit. Otherwise, let $\theta_i = (i, (w_{i,0}, w_{i,1})$. Without loss of generality we assume that decryption servers $i = 1, \ldots, k$ were used to generate $\theta_1, \ldots, \theta_k$. To derive the private key for ID let $\lambda_1, \ldots, \lambda_k \in \mathbb{Z}_p$ be the Lagrange coefficients so that $\alpha = f(0) = \sum_{i=1}^{k} \lambda_i f(i)$. Output the reconstituted private key $d_{\mathsf{ID}} = (w_0, w_1)$ given by

$$w_0 = \prod_{i=1}^{k} w_{i,0}^{\lambda_i} \quad , \quad w_1 = \prod_{i=1}^{k} w_{i,1}^{\lambda_i}$$

Encrypt$(\mathsf{PK}, \mathsf{ID}, M)$**.** To encrypt $M \in \mathbb{G}_1$ for identity ID, pick a random $s \in \mathbb{Z}_p$ and output

$$C = \left(e(g_1, g_2)^s \cdot M, \quad g^s, \quad g_1^{s \cdot \mathsf{ID}} h_1^s \right)$$

ValidateCT$(\mathsf{PK}, \mathsf{ID}, C)$**.** To validate a ciphertext $C = (A, B, C_1)$ with respect to an identity ID, output `valid` or `invalid` depending on whether

$$e(B, g_1^{\mathsf{ID}} h_1) = e(C_1, g)$$

Decrypt$(\mathsf{PK}, \mathsf{ID}, d_{\mathsf{ID}}, C)$**.** To decrypt $C = (A, B, C_1)$ using a private key $d_{\mathsf{ID}} = (w_0, w_1)$, first check that $ValidateCT(\mathsf{PK}, \mathsf{ID}, C) = \texttt{valid}$ and that $e(g_1, g_2) \cdot e(g_1^{\mathsf{ID}} h_1, w_1) = e(g, w_0)$. If either check fails, output $\perp$ and exit. Otherwise, output the plaintext

$$A \cdot e(C_1, w_1)/e(B, w_0)$$

The two checks during decryption ensure that C is a valid ciphertext under ID and that d_{ID} is a valid private key for ID. These checks are needed to ensure consistency of key generation in case some server misbehave. If these conditions are fullfilled, then the decryption is correct, because $(w_0, w_1) = (g_2^{\alpha}(g_1^{\mathsf{ID}} h_1)^{\bar{r}},\ g^{\bar{r}})$ for some $\bar{r} \in \mathbb{Z}_p$, and

$$A \cdot \frac{e(C_1, w_1)}{e(B, w_0)} = M \cdot e(g_1, g_2)^s \cdot \frac{e(g_1^{\mathsf{ID}}, g)^{s\bar{r}} \cdot e(h_1, g)^{s\bar{r}}}{e(g, g_2^{\alpha})^s \cdot e(g, g_1^{\mathsf{ID}})^{s\bar{r}} \cdot e(g, h_1)^{s\bar{r}}} = M$$

3.1 Security

We now prove the semantic security of this threshold IBE against selective identity attacks. The proof is based on the proof in [3] and gives a tight reduction.

As in [3], the key to the simulation is the construction of a public key $(..., h_1, ...)$ that allows the simulator to calculate private key shares for any identity except ID^*. A difference from the proof in [3] is that the simulator must be able to extract private key shares (i.e., elements of the vector SK). In addition, the simulator must produce a valid verification key VK, which only exists in the threshold setting. In order to construct the components of VK that correspond to the corrupted servers, the simulator does interpolation in the exponent. The Lagrange interpolation coefficients are blinded and yet carry over unaffected through the bilinear map in the verification equation. The details follow.

Theorem 1. *Suppose the Decision BDH assumption holds for $\mathcal{GG}$. Then the TIBE system above is semantically secure against selective identity, chosen plaintext attacks.*

Proof. First, suppose $\mathcal{A}$ has advantage $\mathsf{AdvIND\text{-}ID}_{\mathcal{A},n,k} > \epsilon$ in attacking the threshold IBE system for a given value of the security parameter Λ. We build an algorithm $\mathcal{B}$ that solves the Decision BDH problem in a random instance $\mathbb{G} \xleftarrow{\mathrm{R}} \mathcal{GG}(\Lambda)$ with advantage ϵ.

Let there thus be a random bilinear group $\mathbb{G} \xleftarrow{\mathrm{R}} \mathcal{GG}(\Lambda)$ and a random generator $g \in \mathbb{G}^*$ of $\mathbb{G}$. Algorithm $\mathcal{B}$ is given as input a random tuple $(\mathbb{G}, g, g^a, g^b, g^c, T)$ that is either sampled from $\mathcal{P}_{BDH}$ (where $T = e(g, g)^{abc}$) or from $\mathcal{R}_{BDH}$ (where T is uniform and independent in $\mathbb{G}_1$). Algorithm $\mathcal{B}$'s goal is to output 1 ("`true`") if $T = e(g, g)^{abc}$ and 0 ("`false`") otherwise. Set $g_1 = g^a,\ g_2 = g^b,\ g_3 = g^c$. Algorithm $\mathcal{B}$ works by interacting with $\mathcal{A}$ in a threshold selective-ID game as follows:

Initialization. The adversary $\mathcal{A}$ chooses a set S of $k-1$ decryption servers that it wants to corrupt. Let $S = \{s_1, \ldots, s_{k-1}\} \subset \{1, \ldots, n\}$. The adversary $\mathcal{A}$ also announces the identity ID^* it wants to attack.

Setup. $\mathcal{B}$ does the following:

1. First, $\mathcal{B}$ picks a random integer $\gamma \in \mathbb{Z}_p$ and defines $h_1 = g_1^{-\mathsf{ID}^*} g^\gamma \in \mathbb{G}$. Algorithm $\mathcal{B}$ gives $\mathcal{A}$ the public key $\mathsf{PK} = (\mathbb{G}, g, g_1, g_2, h_1)$. Note that the corresponding master key, which is unknown to $\mathcal{B}$, is $g_2^a = g^{ab} \in \mathbb{G}$.
2. Next, $\mathcal{B}$ generates the master key shares for the $k-1$ corrupt servers in S. To do so, $\mathcal{B}$ first picks $k-1$ random integers $\alpha_1, \ldots, \alpha_{k-1} \in \mathbb{Z}_p$. Let $f \in \mathbb{Z}_p[X]$ be the degree $k-1$ polynomial implicitly defined to satisfy $f(0) = a$ and $f(s_i) = \alpha_i$ for $i = 1, \ldots, k-1$; note that $\mathcal{B}$ does not know f since it does not know a. Algorithm $\mathcal{B}$ gives $\mathcal{A}$ the $k-1$ master key shares $\mathsf{SK}_{s_i} = g_2^{\alpha_i}$. These keys are consistent with this polynomial f since $\mathsf{SK}_{s_i} = g_2^{f(s_i)}$ for $i = 1, \ldots, k-1$.
3. Finally, $\mathcal{B}$ constructs the verification key, which is a n-vector $(u_1, \ldots, u_n)$ such that $u_i = g^{f(i)}$ for the polynomial f defined above, as follows.
   - For $i \in S$, computing u_i is easy since $f(i)$ is equal to one of the $\alpha_1, \ldots, \alpha_{k-1}$, which are known to $\mathcal{B}$. Thus, $u_{s_1}, \ldots, u_{s_k} \in \mathbb{G}$ are easy for $\mathcal{B}$ to compute.
   - For $i \notin S$, algorithm $\mathcal{B}$ needs to compute the Lagrange coefficients $\lambda_0, \lambda_1, \ldots, \lambda_{k-1} \in \mathbb{Z}_p$ such that $f(i) = \lambda_0 f(0) + \sum_{j=1}^{k-1} \lambda_j f(s_j)$; these Lagrange coefficients are easily calculated since they do not depend on f. Algorithm $\mathcal{B}$ then sets $u_i = g_1^{\lambda_0} u_{s_1}^{\lambda_1} \cdots u_{s_{k-1}}^{\lambda_{k-1}}$, which entails that $u_i = g^{f(i)}$ as required.

   Once it has computed all the u_i's, $\mathcal{B}$ gives to $\mathcal{A}$ the verification key $\mathsf{VK} = (u_1, \ldots, u_n)$.

Phase 1. $\mathcal{A}$ issues up to q_S private key share generation queries to the uncorrupt servers. Consider a key generation query to server $i \notin S$ for the identity $\mathsf{ID} \neq \mathsf{ID}^*$.

Algorithm $\mathcal{B}$ needs to return $(i, (w_{i,0}, w_{i,1}))$ where $w_{i,0} = \mathsf{SK}_i (g_1^{\mathsf{ID}} h_1)^r$ and $w_{i,1} = g^r$ for some random $r \in \mathbb{Z}_p$. To do so, $\mathcal{B}$ first computes the Lagrange coefficients $\lambda_0, \lambda_1, \ldots, \lambda_{k-1} \in \mathbb{Z}_p$ such that $f(i) = \lambda_0 f(0) + \sum_{j=1}^{k-1} \lambda_j f(s_j)$. Next, $\mathcal{B}$ picks a random $r \in \mathbb{Z}_p$ and sets

$$w_{i,0} = g_2^{\frac{-\gamma\lambda_0}{\mathsf{ID}-\mathsf{ID}^*}} (g_1^{\mathsf{ID}} h_1)^r \cdot \prod_{j=1}^{k-1} g_2^{\lambda_j \alpha_j}, \quad w_{i,1} = g_2^{\frac{-\lambda_0}{\mathsf{ID}-\mathsf{ID}^*}} g^r$$

We claim that $(w_{i,0}, w_{i,1})$ are a valid response to this decryption query. To see this, let $\tilde{r} = r - \frac{b\lambda_0}{\mathsf{ID}-\mathsf{ID}^*}$. Then we have that

$$\begin{aligned} g_2^{\frac{-\gamma\lambda_0}{(\mathsf{ID}-\mathsf{ID}^*)}} (g_1^{\mathsf{ID}} h_1)^r &= g_2^{\frac{-\gamma\lambda_0}{(\mathsf{ID}-\mathsf{ID}^*)}} (g_1^{\mathsf{ID}-\mathsf{ID}^*} g^\gamma)^r \\ &= g_2^{\lambda_0 a} (g_1^{\mathsf{ID}-\mathsf{ID}^*} g^\gamma)^{r - \frac{b\lambda_0}{\mathsf{ID}-\mathsf{ID}^*}} = g_2^{\lambda_0 a} (g_1^{\mathsf{ID}} h_1)^{\tilde{r}} \end{aligned}$$

It follows that the private key share $(i, (w_{i,0}, w_{i,1}))$ defined above satisfies

$$w_{i,0} = g_2^{f(i)} \cdot (g_1^{\mathsf{ID}} h_1)^{\tilde{r}}, \qquad w_{i,1} = g^{\tilde{r}}$$

and $\tilde{r}$ is uniform in $\mathbb{Z}_p$ as required. Hence, $(i, (w_{i,0}, w_{i,1}))$ is a valid response to $\mathcal{A}$.

Challenge. $\mathcal{A}$ outputs two same-length messages M_0 and M_1 on which it wishes to be challenged. $\mathcal{B}$ flips a fair coin $b \in \{0,1\}$, and responds with the challenge ciphertext

$$C = (T \cdot M_b,\ g_3,\ g_3^{\gamma})$$

Since $C = (T \cdot M_b,\ g^c,\ g_1^{c \cdot \mathsf{ID}^*} h_1^c)$, the challenge ciphertext is a valid encryption of M_b with the correct distribution whenever $T = e(g,g)^{abc} = e(g_1, g_2)^c$ (as is the case when the input 5-tuple is sampled from $\mathcal{P}_{BDH}$). On the other hand, when T is uniform and independent in $\mathbb{G}_1$ (which occurs when the input 5-tuple is sampled from $\mathcal{R}_{BDH}$) the challenge ciphertext C is independent of b in the adversary's view.

Phase 2. $\mathcal{A}$ issues additional queries as in Phase 1, to which algorithm $\mathcal{B}$ responds as before.

Guess. Finally, $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$. Algorithm $\mathcal{B}$ concludes its own game by outputting a guess as follows. If $b = b'$ then $\mathcal{B}$ outputs 1 meaning $T = e(g,g)^{abc}$. Otherwise, it outputs 0 meaning $T \neq e(g,g)^{abc}$.

When the input 5-tuple is sampled from $\mathcal{P}_{BDH}$ (where $T = e(g,g)^{abc}$) then $\mathcal{A}$'s view is identical to its view in a real attack game and therefore $\mathcal{A}$ must satisfy $|\Pr[b = b'] - 1/2| > \epsilon$. On the other hand, when the input 5-tuple is sampled from $\mathcal{R}_{BDH}$ (where T is uniform in $\mathbb{G}_1$) then $\Pr[b = b'] = 1/2$. Therefore, with uniformly chosen g in $\mathbb{G}^*$, uniformly chosen a, b, c in $\mathbb{Z}_p$, and uniformly chosen T in $\mathbb{G}_1$, we have, as required, that

$$\left| \begin{array}{l} \Pr\left[\mathcal{B}(\mathbb{G}, g, g^a, g^b, g^c, e(g,g)^{abc}) = 0\right] \\ - \Pr\left[\mathcal{B}(\mathbb{G}, g, g^a, g^b, g^c, T) = 0\right] \end{array} \right| \ \geq\ \left| \left(\frac{1}{2} \pm \epsilon\right) - \frac{1}{2} \right| \ =\ \epsilon$$

To complete the proof of Theorem 1 it remains to prove consistency of key generation. We argue that for any algorithm $\mathcal{A}$ we have $\mathsf{AdvCD\text{-}ID}_{\mathcal{A},n,k}(\Lambda) = 0$. To see this, observe that the two tests performed during decryption ensure that *Decrypt*$(\mathsf{PK}, \mathsf{ID}, d_{\mathsf{ID}}, C)$ outputs the same value for all reconstituted keys d_{ID} that pass the tests. Furthermore, conditions (1)–(4) needed for the adversary to win the consistency of key generation game ensure that both tests succeed. Hence, *Decrypt* will output the same value no matter which key d_{ID} is given as input, and thus $\mathsf{AdvCD\text{-}ID}_{\mathcal{A},n,k}(\Lambda) = 0$.

4 Threshold Public Key Encryption from Threshold IBE

In this section, we use the techniques of Canetti et al. [10] to show that any semantically secure TIBE gives a chosen ciphertext secure TPKE. Later, we apply this transformation on our TIBE to obtain an efficient chosen ciphertext secure TPKE in the standard model.

Let $\mathcal{E}_{TIBE} = (Setup_{TIBE},\ ShareKeyGen_{TIBE},\ ShareVerify_{TIBE},\ Combine_{TIBE}, \ldots)$ be a TIBE system. Let $\mathcal{S} = (SigKeyGen,\ Sign,\ SigVerify)$ be a signature system. We construct a TPKE as follows:

$Setup_{TPKE}(n, k, \Lambda)$. To generate a TPKE key set, execute $Setup_{TIBE}(n, k, \Lambda)$ from the TIBE system and output the resulting tuple $(\mathsf{PK}, \mathsf{VK}, \mathbf{SK})$.

$Encrypt_{TPKE}(\mathsf{PK}, M)$. To encrypt a message M under the public key PK, first run $SigKeyGen(\Lambda)$ to obtain a signing/verification key pair $(SigK, VerK)$. Next, run $Encrypt_{TIBE}(\mathsf{PK}, VerK, M)$ to obtain a ciphertext C_0, i.e., using $VerK$ as the identity to encrypt to. Then, run $Sign(SigK, C_0)$ to obtain a signature σ. Output the triple $C = (C_0, VerK, \sigma)$ as the complete ciphertext.

$ShareDecrypt_{TPKE}(\mathsf{PK}, i, \mathsf{SK}_i, C)$. To obtain a partial decryption of a ciphertext $C = (C_0, VerK, \sigma)$ under private key share SK_i, do the following:

1. Run $SigVerify(VerK, C_0, \sigma)$. If the verification fails, output $(i, \perp)$ and exit.
2. Run $ValidateCT_{TIBE}(\mathsf{PK}, VerK, C_0)$. If the validation fails, output $(i, \perp)$ and exit.
3. Run $ShareKeyGen_{TIBE}(\mathsf{PK}, i, \mathsf{SK}_i, VerK)$ to obtain a TIBE private key share μ for the identity $VerK$. Output μ as the decryption share.

$ShareVerify_{TPKE}(\mathsf{PK}, \mathsf{VK}, C, \mu)$. To verify a decryption share μ with respect to a ciphertext $C = (C_0, VerK, \sigma)$ under verification key VK, do the following:

1. Run $SigVerify(VerK, C_0, \sigma)$ and $ValidateCT_{TIBE}(\mathsf{PK}, VerK, C_0)$. If either test fails do: if $\mu = (i, \perp)$ then output `valid` and exit and if not then output `invalid` and exit.
2. Otherwise, both tests succeeded. Run $ShareVerify_{TIBE}(\mathsf{PK}, \mathsf{VK}, VerK, \mu)$ and output the result.

$Combine_{TPKE}\big(\mathsf{PK}, \mathsf{VK}, C, \{\mu_1, \ldots, \mu_k\}\big)$. To obtain a full decryption of a ciphertext $C = (C_0, VerK, \sigma)$ given k partial decryption shares $\mu_1, \ldots, \mu_k$, first check that all shares are valid and none are of the form $(i, \perp)$. Output $\perp$ and exit if not. Next, run $Combine_{TIBE}(\mathsf{PK}, \mathsf{VK}, VerK, \{\mu_1, \ldots, \mu_k\})$ to obtain a private key d for identity $VerK$. If $d = \perp$, output $\perp$ and exit. Otherwise, run $Decrypt_{TIBE}(\mathsf{PK}, VerK, d, C_0)$ and output the result.

4.1 Security

The following theorem proves security of this system. The proof is based on the proof in [10].

Theorem 2. *Suppose $\mathcal{E}_{TIBE}$ is a selective-ID secure TIBE and $\mathcal{S}$ is existentially unforgeable under a one chosen message attack. Then the TPKE system above is chosen ciphertext secure.*

Proof. Suppose $\mathcal{A}$ has non-negligible advantage in attacking the TPKE above. First, suppose $\mathsf{AdvCCA}_{\mathcal{A},n,k}(\Lambda) > 1/\Lambda^c$ for some $c > 0$, and sufficiently large Λ. We build an algorithm that either breaks the TIBE or breaks the signature scheme. We start with an algorithm $\mathcal{B}$ that breaks the TIBE. Algorithm $\mathcal{B}$ uses $\mathcal{A}$ to interact with a TIBE challenger as follows:

Initialization. Algorithm $\mathcal{B}$ runs $\mathcal{A}$ to obtain a list $S \subset \{1, \ldots, n\}$ of the $k-1$ servers that $\mathcal{A}$ wishes to corrupt. Next, $\mathcal{B}$ runs $SigKeyGen(\Lambda)$ to obtain a signing key $SigK^*$ and a verification key $VerK^*$. It outputs the set S and the identity $\mathsf{ID}^* = VerK^*$ to the TIBE challenger.

Setup. The TIBE challenger runs $Setup(n, k, \Lambda)$ to obtain $(\mathsf{PK}, \mathsf{VK}, \mathbf{SK})$. It gives $\mathcal{B}$ the values PK, VK, and all (j, SK_j) for $j \in S$. Algorithm $\mathcal{B}$ forwards these values to $\mathcal{A}$.

Query phase 1. $\mathcal{A}$ adaptively issues decryption queries of the form (C, i) where $C = (C_0, \mathit{VerK}, \sigma)$ and $i \in \{1, \ldots, n\}$. For each such query:

1. $\mathcal{B}$ runs $\mathit{SigVerify}(\mathit{VerK}, C_0, \sigma)$ and $\mathit{ValidateCT}_{\mathit{TIBE}}(\mathsf{PK}, \mathit{VerK}, C_0)$. If either output is `invalid`, algorithm $\mathcal{B}$ responds to $\mathcal{A}$'s query with $\mu = (i, \perp)$.
2. Otherwise, in the unlikely event that $\mathit{VerK} = \mathit{VerK}^*$, algorithm $\mathcal{B}$ moves to the challenge phase, picks a random $b' \in \{0, 1\}$ as its guess for b, outputs b', and aborts the simulation.
3. Otherwise, $\mathcal{B}$ issues an identity query $(\mathsf{ID} = \mathit{VerK}, i)$ to the TIBE challenger and obtains a private key share θ in return. It gives the decryption share $\mu = \theta$ to $\mathcal{A}$.

Challenge. $\mathcal{A}$ outputs two equal length messages M_0 and M_1. Algorithm $\mathcal{B}$ forwards M_0 and M_1 to the TIBE challenger. Recall that the challenge identity ID^* was set during initialization to $\mathsf{ID}^* = \mathit{VerK}^*$. The TIBE challenger responds with the encryption C_0^* of M_b under ID^* for some $b \in \{0, 1\}$. Algorithm $\mathcal{B}$ then runs $\mathit{Sign}(\mathit{SigK}^*, C_0^*)$ to obtain a signature σ^*. It gives $\mathcal{A}$ the challenge ciphertext $C^* = (C_0^*, \mathit{VerK}^*, \sigma^*)$.

Query phase 2. Algorithm $\mathcal{A}$ continues to issue decryption queries (C, i) where $C = (C_0, \mathit{VerK}, \sigma)$ and $C \neq C^*$. Algorithm $\mathcal{B}$ responds as in the query phase 1; in particular if $\mathit{VerK} = \mathit{VerK}^*$ then $\mathcal{B}$ picks a random $b' \in \{0, 1\}$ as its guess for b, outputs b' and aborts the simulation.

Guess. Eventually, $\mathcal{A}$ outputs its guess $b' \in \{0, 1\}$ for b. Algorithm $\mathcal{B}$ forwards b' to the TIBE challenger and wins the game if $b = b'$.

This completes the description of algorithm $\mathcal{B}$. Let $\mathsf{AdvIND\text{-}ID}_{\mathcal{B},n,k}(\Lambda)$ be $\mathcal{B}$'s advantage in winning the TIBE game above. Let $\mathsf{AdvCCA}_{\mathcal{A},n,k}(\Lambda)$ be $\mathcal{A}$'s advantage in winning the TPKE game. Let `abort` be the event that $\mathcal{B}$ aborted during the simulation in query phase 1 or 2.

As long as event `abort` does not happen, $\mathcal{B}$'s simulation of a TPKE challenger is perfect. Therefore,

$$|\mathsf{AdvIND\text{-}ID}_{\mathcal{B},n,k}(\Lambda) - \mathsf{AdvCCA}_{\mathcal{A},n,k}(\Lambda)| < \Pr[\mathtt{abort}] \tag{1}$$

Now, observe that when event `abort` happens, then $\mathcal{B}$ obtains an existential forgery for the signature public key VerK^*. If `abort` happens in query phase 1 then the forgery is obtained with no chosen message queries. If `abort` happens in query phase 2 then the forgery is obtained after one chosen message query. Either way, we obtain an algorithm, $\mathcal{C}$, that produces an existential forgery on the signature scheme $\mathcal{S}$ with probability $\Pr[\mathtt{abort}]$ using at most one chosen message query. Hence, $\mathsf{AdvSig}_{\mathcal{C}} = \Pr[\mathtt{abort}]$. It now follows from (1) that

$$\mathsf{AdvIND\text{-}ID}_{\mathcal{B},n,k}(\Lambda) + \mathsf{AdvSig}_{\mathcal{C}}(\Lambda) > \mathsf{AdvCCA}_{\mathcal{A},n,k}(\Lambda)$$

Therefore, if $\mathsf{AdvCCA}_{\mathcal{A},n,k}(\Lambda)$ is a non-negligible function then at least one of $\mathsf{AdvIND\text{-}ID}_{\mathcal{B},n,k}(\Lambda)$ or $\mathsf{AdvSig}_{\mathcal{C}}(\Lambda)$ must also be non-negligible, as required.

To complete the proof of Theorem 2 we need to argue that $\mathsf{AdvCD}_{\mathcal{A},n,k}(\Lambda)$ is a negligible function. Suppose, $\mathsf{AdvCD}_{\mathcal{A},n,k}(\Lambda)$ is non-negligible. Then we immediately obtain an algorithm $\mathcal{B}$ for which $\mathsf{AdvCD\text{-}ID}_{\mathcal{B},n,k}(\Lambda)$ is non-negligible contradicting the fact that $\mathcal{E}_{\scriptscriptstyle TIBE}$ is a secure TIBE. To see this, suppose $\mathcal{A}$ outputs (C, S, S') that lets $\mathcal{A}$ win the TPKE decryption consistency game. Let $C = (C_0, \mathit{VerK}, \sigma)$. Then $\mathit{ValidateCT}_{\scriptscriptstyle TIBE}(\mathsf{PK}, \mathit{VerK}, C_0) = \mathtt{valid}$, since otherwise all shares in S and S' must be of the form $(j, \perp)$. Furthermore, $\mathit{ShareVerify}_{\scriptscriptstyle TIBE}(\mathsf{PK}, \mathsf{VK}, \mathit{VerK}, \mu) = \mathtt{valid}$ for all shares $\mu \in S, S'$. Therefore, the decryption shares in S and S' are valid private key shares for $\mathsf{ID} = \mathit{VerK}$. It now follows that $(\mathit{VerK}, C_0, S, S')$ is a tuple that wins the TIBE key generation consistency game as required. This completes the proof of Theorem 2.

5 A Concrete Threshold Public Key System

Our full non-interactive, CCA2-secure, threshold PKE system is immediately obtained by applying the generic transformation of Sections 4 to the threshold IBE system of Section 3. The construction in described in Appendix A. We outline the properties of the system.

Recall that the TIBE of Section 3 worked for identities in $\mathbb{Z}_p^*$ where p was the order of the bilinear groups $\mathbb{G}$. To apply the conversion method we need identities that are public keys of a signature system. Such identities may not be elements of $\mathbb{Z}_p^*$. Therefore, the threshold system described in the appendix uses a collision resistant hash H to hash arbitrary identities into $\mathbb{Z}_p^*$. Security, of course, depends on the BDH assumption, security of the signature system, and the collision resistance of H.

The security of the TPKE scheme follows immediately from that of the underlying TIBE of Section 3 and the generic conversion from TIBE to TPKE from Section 4. We thus have the following corollary.

Corollary 3 *The system in Appendix A is chosen ciphertext secure assuming the BDH assumption holds for $\mathcal{G}\mathcal{G}$, the signature scheme is existentially unforgeable under a one chosen message attack, and the hash function H is collision resistant.*

Thus, we are able to construct a CCA2-secure threshold public key system, without random oracles, in which there is no interaction needed between the decryption parties. The reason we are able to avoid interaction is that using the method of [10] anyone can check that a ciphertext is valid. In the Cramer-Shoup framework only parties possessing the private key can check ciphertext validity, which makes threshold decryption non-trivial.

The system includes additional tests during *ShareDecrypt* and *Decrypt* to provide robustness against misbehaving servers. These tests are possible with no additional information due to the fact that the DDH problem is easy in bilinear groups. In particular, we are able to test that a given IBE private key is valid for a given identity and that a given IBE ciphertext is a valid encryption under a given identity.

We note that a more efficient transformation from IBE to CCA2-secure public-key encryption was presented by Boneh and Katz [6]. Because that transformation uses MACs and commitments instead of signatures, only parties possessing the private key can check ciphertext validity. As a result, the method of [6] does not lend itself to the construction of non-interactive CCA2-secure threshold systems. This is the primary reason why, in the above construction, we had to use the original transformation of [10] based on signatures (or one-time signatures). An elegant alternative was recently proposed in [7].

6 Extensions

Distributed key generation. In the TPKE system of Section 5 one need not rely on a trusted dealer to issue shares to the decryption servers. One can generate a public key and shares of a private key using standard distributed key generation techniques used for ElGamal encryption [32, 22].

Proactive refresh. Proactive refresh enables the decryption servers to refresh their shares of the secret decryption key, without changing the key. Periodic proactive refresh make it harder for an adversary to recover k shares of the secret key, since he must recover all k shares within one time period. The standard proactive refresh techniques of [31, 25, 18] used for ElGamal encryption also apply to our Threshold PKE.

7 Conclusions

We presented a simple non-interactive threshold encryption system that is chosen ciphertext secure without random oracles. The construction illustrates the benefits of building chosen ciphertext security from identity based encryption.

References

1. M. Abe. Robust distributed multiplication without interaction. In *Proceedings of Crypto 1999*, pages 130–47, 1999.
2. J. H. An, Y. Dodis, and T. Rabin. On the security of joint signature and encryption. In *Proceedings of Eurocrypt 2002*, volume 2332 of *LNCS*. Springer-Verlag, 2002.
3. D. Boneh and X. Boyen. Efficient selective-ID identity based encryption without random oracles. In *Proceedings of Eurocrypt 2004*, volume 3027 of *LNCS*, pages 223–38. Springer-Verlag, 2004.
4. D. Boneh and X. Boyen. Short signatures without random oracles. In *Proceedings of Eurocrypt 2004*, volume 3027 of *LNCS*, pages 56–73. Springer-Verlag, 2004.
5. D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. In J. Kilian, editor, *Proceedings of Crypto 2001*, volume 2139 of *LNCS*, pages 213–29. Springer-Verlag, 2001.
6. D. Boneh and J. Katz. Improved efficiency for CCA-secure cryptosystems built using identity based encryption. In *Proceedings of RSA 2005*, LNCS. Springer-Verlag, 2005.

7. X. Boyen, Q. Mei, and B. Waters. Direct chosen ciphertext security from identity-based techniques. In *ACM Conference on Computer and Communications Security—CCS 2005*. ACM Press, 2005. Full version available at `http://eprint.iacr.org/2005/288`.
8. R. Canetti and S. Goldwasser. An efficient threshold public key cryptosystem secure against adaptive chosen ciphertext attack. In *Proceedings of Eurocrypt 1999*, pages 90–106, 1999.
9. R. Canetti, S. Halevi, and J. Katz. A forward-secure public-key encryption scheme. In *Proceedings of Eurocrypt 2003*, volume 2656 of *LNCS*. Springer-Verlag, 2003.
10. R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. In *Proceedings of Eurocrypt 2004*, volume 3027 of *LNCS*, pages 207–22. Springer-Verlag, 2004.
11. R. Cramer, I. Damgård, and Y. Ishai. Share conversion, pseudorandom secret-sharing and applications to secure computation. In *Proceedings of TCC 2005*, volume 3378 of *LNCS*, pages 342–62, 2005.
12. R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal of Computing*, 33:167–226, 2003. Extended abstract in Crypto 1998.
13. I. Damgård, N. Fazio, and A. Nicolosi. Secret-key zero-knowledge protocols for NP and applications to threshold cryptography. manuscript, 2004.
14. A. DeSantis, Y. Desmedt, Y. Frankel, and M. Yung. How to share a function securely. In *Proceedings of STOC 1994*, pages 522–33, 1994.
15. Y. Desmedt and Y. Frankel. Threshold cryptosystems. In *Proceedings of Crypto 1989*, pages 307–15, 1989.
16. Y. Dodis and J. Katz. Chosen-ciphertext security of multiple encryption. In *Proceedings of TCC 2005*, LNCS. Springer-Verlag, 2005.
17. Y. Frankel. A practical protocol for large group oriented networks. In *Proceedings of Eurocrypt 1989*, pages 56–61, 1989.
18. Y. Frankel, P. Gemmell, P. MacKenzie, and M. Yung. Optimal resilience proactive public key cryptosystems. In *Proceedings of FOCS 1997*, pages 384–93, 1997.
19. Y. Frankel, P. Gemmell, and M. Yung. Witness-based cryptographic program checking. In *Proceedings of STOC 1996*, pages 499–08, 1996.
20. P. Gemmel. An introduction to threshold cryptography. *RSA CryptoBytes*, 2(3):7–12, 1997.
21. R. Gennaro, S. Halevi, and T. Rabin. Secure hash-and-sign signatures without the random oracle. In *Proceedings of Eurocrypt 1999*, LNCS, pages 123–39. Springer-Verlag, 1999.
22. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure distributed key generation for discrete-log based cryptosystems. In J. Stern, editor, *Proceedings of Eurocrypt 1999*, volume 1592 of *LNCS*, pages 295–310. Springer-Verlag, 1999.
23. R. Gennaro, T. Rabin, S. Jarecki, and H. Krawczyk. Robust and efficient sharing of RSA functions. *J. Cryptology*, 13(2):273–300, 2000.
24. S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Computing*, 17(2):281–308, 1988.
25. A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung. Proactive secret sharing, or how to cope with perpetual leakage. In *Proceedings of Crypto 1995*, 1995.
26. S. Jarecki and A. Lysyanskaya. Adaptively secure threshold cryptography: introducing concurrency, removing erasures. In *Proceedings of Eurocrypt 2000*, pages 221–42, 2000.

27. A. Joux. A one round protocol for tripartite Diffie-Hellman. In W. Bosma, editor, *Proceedings of ANTS IV*, volume 1838 of *LNCS*, pages 385–94. Springer-Verlag, 2000.
28. A. Joux and K. Nguyen. Separating decision Diffie-Hellman from Diffie-Hellman in cryptographic groups. *Journal of Cryptology*, 16(4):239–47, 2003.
29. K. Kurosawa and Y. Desmedt. A new paradigm of hybrid encryption scheme. In *Proceedings of Crypto 2004*, volume 3152 of *LNCS*, pages 426–42. Springer-Verlag, 2004.
30. P. MacKenzie. An efficient two-party public key cryptosystem secure against adaptive chosen ciphertext attack. In *Proceedings of PKC 2003*, 2003.
31. R. Ostrovsky and M. Yung. How to withstand mobile virus attacks. In *Proceedings of PODC 1991*, pages 51–61, 1991.
32. T. Pederson. A threshold cryptosystem without a trusted party. In *Proceedings of Eurocrypt 1991*, volume 547 of *LNCS*, pages 522–26, 1991.
33. R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairings. In *Proceedings of the Symposium on Cryptography and Information Security—SCIS 2000*, Japan, 2000.
34. V. Shoup and R. Cramer. Universal hash proofs and a paradigm for chosen ciphertext secure public key encryption. In *Proceedings of Eurocrypt 2002*, 2002.
35. V. Shoup and R. Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. *Journal of Cryptology*, 15(2):75–96, 2002. extended abstract in Eurocrypt 1998.

A Description of the Full TPKE System

We give an explicit description of the full non-interactive CCA2-secure threshold PKE system from Section 5. It is obtained by directly composing the constructions given in Sections 3 and 4. The system works as follows:

Setup(n, k, Λ)**.** Run the group generator $\mathcal{GG}(\Lambda)$ to obtain a bilinear group $\mathbb{G}$ of prime order $p > n$. Select random generators g, g_2, h_1 in $\mathbb{G}$, and a random degree $k-1$ polynomial $f \in \mathbb{Z}_p[X]$. Set $\alpha = f(0) \in \mathbb{Z}_p$ and $g_1 = g^\alpha$.
The public key PK consist of $\mathsf{PK} = (\mathbb{G}, g, g_1, g_2, h_1)$. For $i = 1, \ldots, n$ the secret key SK_i of server i is defined as $\mathsf{SK}_i = g_2^{f(i)}$. The public verification key VK consists of PK along with the n-tuple $(g^{f(1)}, \ldots, g^{f(n)})$.
We will also need a collision resistant hash function H that outputs digests in $\mathbb{Z}_p$, and a signature scheme (*SigKeyGen*, *Sign*, *SigVerify*) that is strongly existentially unforgeable against one chosen message attacks. Both H and the signature scheme are part of PK, but we leave them as implicit members to simplify the presentation.

Encrypt(PK, M)**.** To encrypt a message $M \in \mathbb{G}_1$ under the public key $\mathsf{PK} = (g, g_1, g_2, h_1)$, first run *SigKeyGen* to obtain a signing key *SigK* and a signature verification key *VerK*. Let $\mathsf{ID} = H(\mathit{VerK})$. Next, pick a random $s \in \mathbb{Z}_p$ and compute

$$C_0 = \left(e(g_1, g_2)^s \cdot M, \quad g^s, \quad g_1^{s \cdot \mathsf{ID}} h_1^s \right)$$

Let $\sigma = \mathit{Sign}(\mathit{SigK}, C_0)$ be a signature on C_0 using the signing key *SigK*. Output the ciphertext $C = (C_0, \mathit{VerK}, \sigma)$.

ShareDecrypt($\mathsf{PK}, \mathsf{SK}_i, C$)**.** Decryption server i uses its private key share SK_i to partially decrypt a ciphertext $C = (C_0, \mathit{VerK}, \sigma)$ as follows. First, run algorithm $\mathit{SigVerify}(\mathit{VerK}, C_0, \sigma)$ to check that σ is a valid signature of C under VerK. Also let $\mathsf{ID} = H(\mathit{VerK})$ and test whether $e(B, g_1^{\mathsf{ID}} h_1) = e(C_1, g)$. If either condition fails, output $\mu = (i, \perp)$ and exit.

Otherwise, C is well-formed, and the decryption server i needs to output a share of the private key needed to decrypt C_0. To do so, it picks a random r in $\mathbb{Z}_p$, and outputs the decryption share $\mu_i = (i, (w_0, w_1))$, where

$$w_0 = \mathsf{SK}_i \cdot (g_1^{\mathsf{ID}} h_1)^r \qquad \text{and} \qquad w_1 = g^r$$

Notice that (w_0, w_1) is an IBE private key share corresponding to the identity $\mathsf{ID} = H(\mathit{VerK})$.

ShareVerify($\mathsf{PK}, \mathsf{VK}, C, \mu_i$)**.** To verify that μ_i is a correct partial decryption of the ciphertext $C = (C_0, \mathit{VerK}, \sigma) = ((A, B, C_1), \mathit{VerK}, \sigma)$, first run algorithm $\mathit{SigVerify}(\mathit{VerK}, C_0, \sigma)$ to check that σ is a valid signature of C_0 under VerK. Also let $\mathsf{ID} = H(\mathit{VerK})$ and test whether $e(B, g_1^{\mathsf{ID}} h_1) = e(C_1, g)$. We say that C is well-formed if both tests succeed.

1. If C is not well-formed: if μ_i is of the form $(i, \perp)$ then output `valid` and exit, otherwise output `invalid` and exit.
2. If C is well-formed *and* μ_i is of the form $(i, \perp)$, then output `invalid` and exit.
3. Otherwise, C is well-formed and $\mu_i = (i, (w_0, w_1))$. In this case, let $\mathsf{VK} = (u_1, \ldots, u_n)$ where $u_i = g^{f(i)}$, and output `valid` or `invalid` according to whether the following equation holds or not:

$$e(u_i, g_2) \cdot e(g_1^{\mathsf{ID}} h_1, w_1) = e(g, w_0)$$

Combine($\mathsf{PK}, \mathsf{VK}, C, \{\mu_1, \ldots, \mu_k\}$)**.** To decrypt a ciphertext $C = (C_0, \mathit{VerK}, \sigma)$ using the partial decryptions $\mu_1, \ldots, \mu_k$, first check that all shares $\mu_i = (i, \hat{\mu}_i)$ bear distinct server indices i, and that they are all *valid*, i.e., that all $\mathit{ShareVerify}(\mathsf{PK}, \mathsf{VK}, C_0, \mu_i) =$ `valid`; otherwise output $\perp$ and exit.

Without loss of generality, assume that the shares $\mu_1, \ldots, \mu_k$ were generated by the decryption servers $i = 1, \ldots, k$, respectively. The combiner proceeds as follows:

1. If any partial decryption μ_i is of the form $(i, \perp)$, then output $\perp$ and exit.
2. Otherwise, all shares $\mu_1, \ldots, \mu_k$ are of the form $\mu_i = (i, (w_{i,0}, w_{i,1}))$ with distinct i, and $\mathit{SigVerify}(\mathit{VerK}, C_0, \sigma)$ and $\mathit{ValidateCT}(\mathsf{PK}, H(\mathit{VerK}), C_0)$ must both succeed. Determine the Lagrange coefficients $\lambda_1, \ldots, \lambda_k \in \mathbb{Z}_p$ so that $\alpha = f(0) = \sum_{i=1}^{k} \lambda_i f(i)$, and set

$$w_0 = \prod_{i=1}^{k} w_{i,0}^{\lambda_i} \qquad \text{and} \qquad w_1 = \prod_{i=1}^{k} w_{i,1}^{\lambda_i}$$

3. Use (w_0, w_1) to decrypt $C_0 = (A, B, C_1)$, as

$$M = A \cdot e(B, w_0)/e(C_1, w_1)$$

Observe that the above decryption goes through since as we observed earlier $w_0 = g_2^{\alpha} \cdot (g_1^{\mathsf{ID}} h_1)^{\bar{r}}$ and $w_1 = g^{\bar{r}}$ for some $\bar{r} \in \mathbb{Z}_p$, hence (w_0, w_1) is an IBE private key corresponding to $\mathsf{ID} = H(\mathit{VerK})$.