Public Key Encryption with Distributed Keyword Search

Veronika Kuchta(✉) and Mark Manulis

Department of Computing, University of Surrey, Guildford, UK
v.kuchta@surrey.ac.uk, mark@manulis.eu

Abstract. In this paper we introduce Threshold Public Key Encryption with Keyword Search (TPEKS), a variant of PEKS where the search procedure for encrypted keywords is distributed across multiple servers in a threshold manner. TPEKS schemes offer stronger privacy protection for keywords in comparison to traditional PEKS schemes. In particularly, they prevent *keyword guessing attacks* by malicious servers. This protection is not achievable in a single-server PEKS setting.

We show how TPEKS can be built generically from any anonymous Identity-Based Threshold Decryption (IBTD), assuming the latter is indistinguishable, anonymous and robust. In order to instantiate our TPEKS construction we describe an efficient IBTD variant of the Boneh-Franklin IBE scheme. We provide an appropriate security model for such IBTD schemes and give an efficient construction in the random oracle model.

TPEKS constructions are particularly useful in distributed cloud storage systems where none of the servers alone is sufficiently trusted to perform the search procedure and where there is a need to split this functionality across multiple servers to enhance security and reliability.

1 Introduction

Cloud computing provides convenient, on-demand network access to shared services and applications over the Internet. The main advantages of cloud computing are the virtually unlimited data storage capabilities, universal data access, and savings on hardware and software expenses. Despite the many technical and economical advantages, availability and data privacy are amongst those issues that prevent potential users from trusting the cloud services. The reason of these concerns is that upon outsourcing their data to the cloud, users lose control over their data.

While for better availability it is advisable to distribute copies of data across multiple cloud servers, for data privacy and its protection from unauthorized access the use of complete encryption prior to outsourcing is indispensable. If the data is encrypted and the user wishes to access certain files at a later stage, the cloud needs to perform the search and retrieve the corresponding ciphertexts. In order to facilitate the search process each outsourced file is typically associated with a set of keywords. Since adding plaintext keywords [11] to each file prior to

M. Yung et al. (Eds.): INTRUST 2015, LNCS 9565, pp. 62–83, 2016.
DOI: 10.1007/978-3-319-31550-8_5

outsourcing would leak information about the file contents to the cloud services, a better solution is to encrypt the associated keywords and provide cloud services with the ability to search for encrypted keywords. This permission comes in form of trapdoors allowing cloud services to test whether an encrypted file contains keywords for which the trapdoors were derived by the user. Such searchable encryption techniques are particularly helpful to protect outsourcing of sensitive files, e.g., those containing medical and health records [6,20,21].

A range of encryption schemes supporting keyword search have been proposed, based on symmetric encryption (e.g., [12,14,16,27]) and public-key encryption (e.g., [7,9,17,18]) techniques. While some schemes can cope only with single keywords (e.g., [26,29]), which is too restrictive in practice, more advanced schemes (e.g., [11,22,25,28]) can process multiple keywords. The majority of searchable encryption schemes issue trapdoors for keywords and any party in possession of those trapdoors can perform the search procedure on its own. In a cloud-based storage setting this imposes a single point of trust with regard to the search functionality. When it comes to the use of multiple cloud services for better availability, a distributed search approach would therefore help to reduce this trust requirement. Encryption schemes supporting distributed keyword search procedures in a cloud environment exist so far only in the symmetric setting, namely in [30], those constructions however were not formally modeled and analyzed.

Our Threshold Public Key Encryption with Keyword Search (TPEKS). We model security and propose first constructions of Threshold Public Key Encryption with Keyword Search (TPEKS), where the ability to search over encrypted keywords requires participation of at least t parties, each equipped with its own trapdoor share. Main benefits of TPEKS over traditional PEKS constructions include the distribution of trust across multiple servers involved in a search procedure and more importantly stronger privacy protection of keywords against keyword guessing attacks [10,24], based on which information about keywords can be revealed from associated trapdoors; notably, all single-server-based PEKS constructions are inherently vulnerable to keyword guessing attacks.

Our security model for TPEKS is motivated by the security goals and known attacks on single-server-based PEKS constructions (e.g. [1,3,4,10,13,19]). The concept of PEKS was introduced by Boneh et al. [7], along with the formalization of two security goals: indistinguishability and consistency. While indistinguishability aims at privacy of encrypted keywords, consistency aims to prevent false positives, where a PEKS ciphertext created for one keyword can successfully be tested with a trapdoor produced for another keyword. Initially, PEKS constructions were able to search for individual keywords, whereas later schemes were designed to handle conjunctive queries on encrypted keywords [17,18,23] and thus process keyword sets within a single search operation. The majority of PEKS schemes offer security against chosen-plaintext attacks [1,7,10,13] and only few constructions remain secure against chosen-ciphertext attacks, e.g. [15]. The vulnerability of PEKS constructions, e.g. [7], against (offline) keyword

guessing attacks was discovered by Byun et al. [10]. In short, a keyword guessing attack can be mounted by creating a PEKS ciphertext for some candidate keyword and then testing this ciphertext with the given trapdoor. Obviously, this attack works if keywords have low entropy, which is what typically happens in practice. As shown by Jeong et al. [19], keyword guessing attacks are inherent to all single-server based PEKS constructions with consistency, a necessary security property of PEKS. Through secret sharing of the trapdoor information across multiple servers, TPEKS significantly reduces the risk of keyword guessing attacks, which are modeled as part of the indistinguishability property.

In the design of our TPEKS construction we extend the ideas underlying the transformation by Abdalla et al. [1] for building indistinguishable and computationally consistent (single-server) PEKS from anonymous Identity-Based Encryption (IBE). Although our transformation also treats identities as keywords, it assumes a different building block, namely anonymous Identity-Based Threshold Decryption (IBTD), which extends IBE by the distributed decryption process for which a threshold number t-out-of-n servers contribute with their own decryption shares. We show that while IBTD anonymity is essential for the indistinguishability (with resistance to keyword guessing attacks) of the constructed TPEKS scheme, IBTD indistinguishability informs computational consistency property of the TPEKS scheme. Aiming to instantiate our TPEKS construction, we propose an anonymous IBTD scheme, as a modification of the well-known anonymous IBE scheme by Boneh and Franklin (BF) [8]. This modification is performed by distributing the decryption process of the original BF-scheme.

2 Anonymous Identity-Based Threshold Decryption

We start with the definitions of Identity-Based Threshold Decryption (IBTD) along with its security properties: indistinguishability, anonymity and robustness. Our IBTD model extends the model from [5], where this primitive along with the indistinguishability property was introduced, by additional anonymity requirement, which also requires some small modifications to the assumed syntax of the IBTD decryption process in comparison to [5].

2.1 IBTD Syntax and Security Goals

We formalize the IBTD syntax in Definition 1. In contrast to [5], we treat the validity checking process for decryption shares implicitly as part of the decryption algorithm `Dec`, whereas in [5] this property was outsourced into a separate verification algorithm, aiming at public verifiability of individual decryption shares. In our case, where we additionally require the IBTD scheme to be anonymous such syntax change is necessary, as discussed in Remark 1.

Definition 1 (Identity-Based Threshold Decryption (IBTD)). *An IBTD scheme consists of the following algorithms* (`Setup`, `KeyDer`, `Enc`, `ShareDec`, `Dec`)*:*

$\texttt{Setup}(n, t, 1^k)$: *On input the number of decryption servers* n*, a threshold parameter* t*,* $(1 \leq t \leq n)$ *and a security parameter* 1^k*, it outputs a master public key* mpk *and a master secret key* msk*.*

$\texttt{KeyDer}(mpk, msk, id, t, n)$: *On input a master public key* mpk*, master secret key* msk*, identity* id*, and threshold parameters* t, n*, it computes the secret key* sk_{id} *for identity* id *and outputs the private tuple* $(i, sk_{id,i})$ *for server* S_i $1 \leq i \leq n$*.*

$\texttt{Enc}(mpk, id, m)$: *On input* mpk, id, m *it outputs a ciphertext* C*.*

$\texttt{ShareDec}(mpk, (i, sk_{id,i}), C)$: *On input a master public key* mpk*, secret shares* (i, sk_{id_i}) *for servers* $1 \leq i \leq n$ *and ciphertext* C*. It outputs decryption shares* δ_i *for* $1 \leq i \leq n$*.*

$\texttt{Dec}(mpk, \{\delta_i\}_{i \in \Omega}, C)$: *On input a master public key* mpk*, a set of decryption shares* $\{\delta_i\}_{i \in \Omega}$*, where* $|\Omega| \geq t$ *and a ciphertext* C*. It outputs* m *(which can also be* $\perp$ *to indicate a failure).*

The following Definition 2 formalizes IBTD indistinguishability against chosen-ciphertext attacks (IBTD-IND-CCA) and bears similarities with the corresponding definition for IBE [8]; namely, our experiment takes into account the threshold nature of the decryption algorithm allowing the adversary to reveal up to $t - 1$ secret key shares.

Definition 2 (IBTD Indistinguishability). *Let* $\mathcal{A}_{ind}$ *be a PPT adversary against the* IBTD-IND-CCA *security of the IBTD scheme, associated with the following experiment* $\mathbf{Exp}^{\text{IBTD-IND-CCA}-b}_{\mathcal{A}_{ind}}(1^\lambda)$*:*

1. $(mpk, msk) \xleftarrow{r} \texttt{Setup}(1^\lambda, t, n)$
2. *Let* **List** *be a list comprising* (id, S_{id})*, where* $S_{id} := \{(1, sk_{id,1}), \ldots, (n, sk_{id,n})\}$ *and* $(i, sk_{id,i})$ *are the outputs of* $\texttt{KeyDer}(mpk, msk, id, t, n)$ *algorithm. Note: at the beginning of the experiment the list is empty.*
3. $(id^*, m_0, m_1, state) \xleftarrow{r} A_{ind}^{\mathcal{O}\texttt{KeyDer}(\cdot), \mathcal{O}\texttt{Dec}(\cdot)}(find, mpk)$
4. *if* $(id^*, S_{id^*}) \notin$ **List**, *run* $(sk_{id^*}, (1, sk_{id^*,1}), \ldots, (n, sk_{id^*,n})) \xleftarrow{r} \texttt{KeyDer}(mpk, msk, id^*, t, n)$*, set* $S_{id^*} := \{(1, sk_{id^*,1}), \ldots, (n, sk_{id^*,n})\}$*, add* (id^*, S_{id^*}) *to* **List**
5. *pick* $b \in \{0, 1\}$*, compute* $C^* \xleftarrow{r} \texttt{Enc}(mpk, id^*, m_b)$
6. $b' \xleftarrow{r} \mathcal{A}_{ind}^{\mathcal{O}\texttt{KeyDer}(\cdot), \mathcal{O}\texttt{Dec}(\cdot)}(guess, C^*, mpk)$

The experiment outputs 1 *if all of the following holds:*

- $b' = b$
- $\mathcal{A}$ *issued at most* $t - 1$ *queries* $\mathcal{O}\texttt{KeyDer}(id^*, i)$
- $\mathcal{A}_{ind}$ *did not query* $\mathcal{O}\texttt{Dec}(id^*, C^*)$

where the two oracles are defined as follows:

$\mathcal{O}\texttt{KeyDer}(id, i)$*: On input* (id, i) *check whether* $(id, S_{id}) \in$ **List**. *If so, parse* S_{id} *as* $\{(1, sk_{id,1}), \ldots, (n, sk_{id,n})\}$ *and output* (i, sk_{id_i})*. If* $(id, S_{id}) \notin$ **List** *run* $S \xleftarrow{r} \texttt{KeyDer}(mpk, msk, id, t, n)$*. Add* (id, S_{id}) *to* **List**, *output* $(i, sk_{id,i})$*.*

$\mathcal{O}\texttt{Dec}(id, C)$*: On input* (id, C) *check whether* $(id, S_{id}) \in$ **List**. *If so, parse* S_{id} *as* $\{(1, sk_{id,1}), \ldots, (n, sk_{id,n})\}$, *run* $\delta_i \xleftarrow{r} \texttt{ShareDec}(mpk, (i, sk_{id,i}), C)$ *for* $i \in [n]$. *Take at least* t*-out-of-*n *decryption shares* δ_i, *run* $\texttt{Dec}(\{\delta_i\}_{i\in\Omega}, C)$, *where* $|\Omega| \geq t$ *and output* m *or* 0. *If* $(id, S_{id}) \notin$ **List**, *compute* $S_{id} \xleftarrow{r} \texttt{KeyDer}(mpk, msk, id, t, n)$ *add* (id, S_{id}) *to the* **List**. *Compute* $\delta_i \xleftarrow{r} \texttt{ShareDec}(mpk, (i, sk_{id,i}), C)$, *where* $i \in [n]$. *Take at least* t*-out-of-*n *decryption shares* δ_i, *run* $\texttt{Dec}(\{\delta_i\}_{i\in\Omega}, C)$, *output* m *or* 0.

$\mathcal{A}_{ind}$*'s success is given as*

$$\mathbf{Adv}^{\text{IBTD-IND-CCA}-b}_{\mathcal{A}_{ind}}(1^\lambda) = \Big| Pr\left[\mathbf{Exp}^{\text{IBTD-IND-CCA}-1}_{\mathcal{A}_{ind}}(1^\lambda) = 1\right] - Pr\left[\mathbf{Exp}^{\text{IBTD-IND-CCA}-0}_{\mathcal{A}_{ind}}(1^\lambda) = 1\right]\Big|.$$

The scheme is IBTD-IND-CCA *secure if* $\mathbf{Adv}^{\text{IBTD-IND-CCA}-b}_{\mathcal{A}_{ind}}$ *is negligible.*

In Definition 3 we model anonymity as a new property for IBTD schemes. Our definition bears some similarity with the anonymity property for IBE schemes as defined, e.g. in [1], except that we consider chosen-ciphertext attacks through the inclusion of the decryption oracle and account for the threshold setting by allowing the anonymity adversary to reveal up to $t-1$ secret key shares. This latter ability is particularly important for achieving protection against keyword guessing attacks for our transformation from IBTD to TPEKS.

Definition 3 (IBTD Anonymity). *Let* $\mathcal{A}_{ano}$ *be a probabilistic polynomial-time adversary against the* IBTD-ANO-CCA *security of the IBTD scheme, associated with the following experiment* $\mathbf{Exp}^{\text{IBTD-ANO-CCA}-b}_{\mathcal{A}_{ano}}(1^\lambda)$

1. $(mpk, msk) \xleftarrow{r} \texttt{Setup}(1^\lambda, t, n)$
2. *Let* **List** *be a list storing* (id, S_{id}), *where* $S_{id} := \{(1, sk_{id,1}), \ldots, (n, sk_{id,n})\}$ *and* $(i, sk_{id,i})$ *are the outputs of* $\texttt{KeyDer}(mpk, msk, id, t, n)$ *algorithm. Note: at the beginning of the experiment the list is empty.*
3. $(id_0, id_1, m^*, state) \xleftarrow{r} A_{ano}^{\mathcal{O}\texttt{KeyDer}(\cdot),\mathcal{O}\texttt{Dec}(\cdot)}(find, mpk)$
4. *if* $(id_0, S_{id_0}) \notin$ **List**: $(sk_{id_0}, (1, sk_{id_0,1}), \ldots, (n, sk_{id_0,n})) \xleftarrow{r} \texttt{KeyDer}(mpk, msk, id_0, t, n)$, *set* $S_{id_0} := \{(1, sk_{id_0,1}), \ldots, (n, sk_{id_0,n})\}$, *add* (id_0, S_{id_0}) *to* **List**
5. *if* $(id_1, S_{id_1}) \notin$ **List**: $(sk_{id_1}, (1, sk_{id_1,1}), \ldots, (n, sk_{id_1,n})) \xleftarrow{r} \texttt{KeyDer}(mpk, msk, id_1, t, n)$, *set* $S_{id_1} := \{(1, sk_{id_1,1}), \ldots, (n, sk_{id_1,n})\}$, *add* (id_1, S_{id_1}) *to* **List**
6. $b \in \{0, 1\}$; $C^* \xleftarrow{r} \texttt{Enc}(mpk, id_b, m^*)$
7. $b' \xleftarrow{r} \mathcal{A}_{ano}^{\mathcal{O}\texttt{KeyDer}(\cdot),\mathcal{O}\texttt{Dec}(\cdot)}(guess, C^*, mpk)$.

The experiment outputs 1 *if all of the following holds*

- $b' = b$
- $\mathcal{A}_{ano}$ *issued at most* $t-1$ *queries* $\mathcal{O}\texttt{KeyDer}(id_0, i)$ *and at most* $t-1$ *queries* $\mathcal{O}\texttt{KeyDer}(id_1, i)$
- $\mathcal{A}_{ano}$ *did not query* $\mathcal{O}\texttt{Dec}(id_0, C^*)$ *or* $\mathcal{O}\texttt{Dec}(id_1, C^*)$

where the two oracles are defined as follows:

$\mathcal{O}\texttt{KeyDer}(id, i)$: *On input* (id, i) *check whether* $(id, S_{id}) \in$ **List**. *If so, parse* S_{id} *as* $\{(1, sk_{id,1}), \ldots, (n, sk_{id,n})\}$ *and output* (i, sk_{id_i}). *If* $(id, S_{id}) \notin$ **List** *run* $S_{id} \xleftarrow{r} \texttt{KeyDer}(mpk, msk, id, t, n)$. *Add* (id, S_{id}) *to* **List**, *output* $(i, sk_{id,i})$.

$\mathcal{O}\texttt{Dec}(id, C)$*: On input* (id, C) *check whether* $(id, S_{id} \in$ **List**. *If so, parse* S_{id} *as* $\{(1, sk_{id,1}), \ldots, (n, sk_{id,n})\}$, *compute* $\delta_i \xleftarrow{r} \texttt{ShareDec}(mpk, (i, sk_{id,i}), C)$ *for* $i \in [n]$. *Take at least* t*-out-of-*n *decryption shares* δ_i, *run* $\texttt{Dec}(\{\delta_i\}_{i\in\Omega}, C)$, *where* $|\Omega| \geq t$ *and output* m *or* 0. *If* $(id, S_{id}) \notin$ **List**, *compute* $S_{id} \xleftarrow{r} \texttt{KeyDer}(mpk, msk, id, t, n)$ *add* (id, S_{id}) *to the* **List**. *Compute* $\delta_i \xleftarrow{r} \texttt{ShareDec}(mpk, (i, sk_{id,i}), C)$, *where* $i \in [n]$. *Take at least* t*-out-of-*n *decryption shares* δ_i, *run* $\texttt{Dec}(\{\delta_i\}_{i\in\Omega}, C)$, *output* m *or* 0.

The advantage of $\mathcal{A}_{ano}$ *is defined as*

$$\mathbf{Adv}^{\text{IBTD-ANO-CCA}-b}_{\mathcal{A}_{ano}}(1^\lambda) = \Big| Pr\left[\mathbf{Exp}^{\text{IBTD-ANO-CCA}-1}_{\mathcal{A}_{ano}}(1^\lambda) = 1\right] - Pr\left[\mathbf{Exp}^{\text{IBTD-ANO-CCA}-0}_{\mathcal{A}_{ano}}(1^\lambda) = 1\right]\Big| .$$

The scheme is IBTD-ANO-CCA *secure if* $\mathbf{Adv}^{\text{IBTD-ANO-CCA}}_{\mathcal{A}_{ano}}$ *is negligible.*

Remark 1. Without the aforementioned change to the IBTD syntax of the decryption process, in comparison to [5], we would not be able to allow adversarial access to up to $t-1$ decryption shares for the challenge ciphertext in the above anonymity experiment. The ability to publicly verify individual decryption shares using the challenge ciphertext and a candidate identity (as in [5]) would rule out any meaningful definition of anonymity; in particular, a single decryption share for the challenge ciphertext would suffice to break its anonymity property. In fact, it can be easily verified that the IBTD construction in [5] is not anonymous according to our definition.

Robustness of IBTD. In the following definition we formalize IBTD robustness, meaning that the decryption algorithm will output $\perp$ with overwhelming probability if an IBTD ciphertext computed for some id is decrypted using $sk_{id'}$ for $id' \neq id$. Our definition of strong robustness extends the one for IBE schemes in [2] to the threshold decryption setting.

Definition 4 (IBTD-SROB-CCA)**.** *Let* $\mathcal{A}_{rob}$ *be a probabilistic polynomial-time adversary against the* IBTD-SROB-CCA *security of the IBTD scheme, associated with the following experiment* $\mathbf{Exp}^{\text{IBTD-SROB-CCA}}_{\mathcal{A}_{rob}}(1^\lambda)$*:*

1. $(mpk, msk) \xleftarrow{r} \texttt{Setup}(1^\lambda, t, n), b \xleftarrow{r} \{0,1\}$, **List** $:= \emptyset$, $I = \emptyset$
2. *Let* **List** *be a list storing* (id, S_{id_b}, I_b), *with* $S_{id_b} := \{(1, sk_{id_b,1}), \ldots, (n, sk_{id_b,n})\}$ *and* $b \xleftarrow{r} \{0,1\}$, *where* $(i, sk_{id_b,i}) \leftarrow \texttt{KeyDer}(mpk, msk, id_b, t, n)$
3. $(id_0^*, id_1^*, C^*, state) \xleftarrow{r} A_{rob}^{\mathcal{O}\texttt{KeyDer}(\cdot), \mathcal{O}\texttt{Dec}(\cdot)}(find, mpk)$
4. *(i) If* $id_0 = id_1$ *then return* 0.
   (ii) If $(id_0, S_{id_0}, I_0) \notin$ **List** *or* $(id_1, S_{id_1}, I_1) \notin$ **List**, *return* 0.

(iii) If $|I_0| \geq t$ *or* $|I_1| \geq t$*, then return* 0*. Else compute decryption shares* $\delta_{0,i} \xleftarrow{r} \texttt{ShareDec}(mpk, (i, sk_{id_0,i}), C)$*,* $m_0 \xleftarrow{r} \texttt{Dec}(mpk, \{\delta_{0,i}\}_{i\in\Omega}, C)$ *and* $\delta_{1,i} \xleftarrow{r} \texttt{ShareDec}(mpk, (i, sk_{id_1,i}), C)$*,* $m_1 \xleftarrow{r} \texttt{Dec}(mpk, \{\delta_{1,i}\}_{i\in\Omega}, C)$*. If* $m_0 \neq \perp$ *and* $m_1 \neq \perp$ *return* 1*.*

$\mathcal{O}\texttt{KeyDer}(id, i)$*: On input* (id, i) *check whether* $(id, S_{id}) \notin$ **List***. If so, compute* $S \xleftarrow{r} \texttt{KeyDer}(mpk, msk, id, t, n)$*, where* $S_{id} := \{(1, sk_{id,1}), \ldots, (n, sk_{id,n})\}$ *and* $I \subset [1, n]$*, add* (id, S_{id}, I) *to* **List***. Then add* i *to* I *and return* $(i, sk_{id,i})$*.*

$\mathcal{O}\texttt{Dec}(id, C)$*: On input* (id, C) *check whether* $(id, S_{id}) \notin$ **List***. If so, compute* $S_{id} \xleftarrow{r} \texttt{KeyDer}(mpk, msk, id, t, n)$ *add* (id, S_{id}, I) *to* **List***. Finally compute* $\delta_i \xleftarrow{r} \texttt{ShareDec}(mpk, (i, sk_{id,i}), C)$*,* $i \in [n]$*,* $m \leftarrow \texttt{Dec}(mpk, \{\delta_i\}_{i\in\Omega}, C)$*, where* $|\Omega| \geq t$*. Output* m*.*

We have: $\mathbf{Adv}^{\text{IBTD-SROB-CCA}}_{\mathcal{A}_{rob}}(1^\lambda) = \left| Pr\left[\mathbf{Exp}^{\text{IBTD-SROB-CCA}}_{\mathcal{A}_{rob}}(1^\lambda) = 1\right]\right|$.

2.2 An Anonymous IBTD Scheme Based on Boneh-Franklin IBE

We propose a concrete IBTD construction, based on Boneh-Franklin IBE [8] where we apply secret sharing to individual private keys and to the decryption procedure. In particular, upon receiving at least t decryption shares, the decryption algorithm outputs either the message m or 0 (to indicate a failure). In contrast to the so-far only IBTD scheme in [5], which also builds on the Boneh-Franklin IBE, our construction is anonymous. Abdalla et al. [2] proved that Boneh-Franklin IBE is robust in the random oracle model.

Definition 5 (Anonymous IBTD Scheme). $\texttt{Setup}(n, t, 1^k)$*: On input a security parameter* 1^k*, it specifies* $\mathbb{G}, \mathbb{G}_T$ *of order* $q \geq 2^k$*, chooses a generator* $g \in \mathbb{G}$*, specifies a bilinear map* $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$*, random oracles* H_1, H_2, H_3, H_4 *s.t.* $H_1 : \{0,1\}^* \to \mathbb{G}$*;* $H_2 : \mathbb{G}_T \to \{0,1\}^\ell$*;* $H_3 : \{0,1\}^\ell \times \{0,1\}^\ell \to \mathbb{Z}_q^*$*;* $H_4 : \{0,1\}^\ell \to \{0,1\}^\ell$*. The message space is* $\mathcal{M} = \{0,1\}^\ell$*. The ciphertext space is* $\mathcal{C} = \mathbb{G}^* \times \{0,1\}^\ell$*. It picks* $x \xleftarrow{r} \mathbb{Z}_q^*$ *and computes* $Y = g^x$*. It returns* $mpk = (\mathbb{G}, \mathbb{G}_T, q, g, e, H_1, H_2, H_3, H_4, Y)$ *and* $msk = x$*.*

$\texttt{KeyDer}(mpk, msk, id, t, n)$*: On input an identity* id*, computes* $Q_{id} = H_1(id) \in \mathbb{G}^*$ *where* $1 \leq t \leq n < q$ *and using* $msk = x$ *it computes* $sk_{id} = Q_{id}^x = H_1(id)^x$*. It picks* $a_1, \ldots, a_{t-1} \xleftarrow{r} \mathbb{G}$*, computes a polynomial* $f(u) = f(0) + \sum_{i=1}^{t-1} a_i u^i$*, where* $f(u) \in \mathbb{Z}_q(u)$*,* $u \in \mathbb{N} \cup \{0\}$*, s.t.* $f(0) = x$*. It outputs* n *master key shares* $(i, sk_{id,i})$*, where* $sk_{id,i} = Q_{id}^{f(i)}$ *for* $i \in \{1, \ldots, n\}$*. To derive the private key let* $\lambda_1, \ldots, \lambda_t \in \mathbb{Z}_q$ *be the Lagrange coefficients, s.t.* $x = \sum_{i=0}^{t-1} \lambda_i f(i)$*.*

$\texttt{Enc}(mpk, id, m)$*: On input the public key* mpk*, a plaintext* $m \in \{0,1\}^\ell$ *and an identity* $id \in \{0,1\}^*$ *computes* $Q_{id} = H_1(id) \in \mathbb{G}^*$*, chooses* $\sigma \xleftarrow{r} \{0,1\}^\ell$*, sets* $r = H_3(\sigma, m)$*. It computes* $U = g^r, V = \sigma \oplus H_2(\kappa_{id}), W = m \oplus H_4(\sigma)$*, where* $\kappa_{id} = e(Q_{id}, Y)^r$ *and returns* $C = \langle U, V, W \rangle$*.*

$\texttt{ShareDec}(mpk, (i, sk_{id,i}), C)$*: On input a ciphertext* $C = \langle U, V, W \rangle$ *and a secret key share* $(i, sk_{id,i})$*, the algorithm returns* $\delta_i = e(sk_{id,i}, U) = e(Q_{id}^{f(i)}, U)$*.*

$\mathtt{Dec}(mpk, \{\delta_i\}_{i\in\Omega}, C)$: *Given a set of decryption shares* $\{\delta_i\}_{i\in\Omega}$, $|\Omega| \geq t$ *and a ciphertext* $C = \langle U, V, W\rangle$, *it computes Lagrange coefficients* $\lambda_i = \prod_{j\in\Omega, j\neq i}^{t} \frac{-j}{i-j}$ *and reconstructs* $\kappa_{id} := \prod_{i=1}^{t} \delta_i^{\lambda_i}$. *It computes* $\sigma = V \oplus H_2(e(\kappa_{id}))$, $m = W \oplus H_4(\sigma)$, *and* $r = H_3(\sigma, m)$. *Then, if* $U = g^r$ *it outputs* m; *otherwise it outputs 0. (Note that the equality check* $U = g^r$ *is essential for detecting inconsistent ciphertexts and by this preventing chosen ciphertext attacks.)*

Correctness. The proposed IBTD scheme is correct since κ_{id} computed by the decryption algorithm is the same as was used by the encryption algorithm, i.e. $\kappa_{id} = \prod_{i=1}^{t} e(sk_{id,i}, U)^{\lambda_i} = \prod_{i=1}^{t} e(sk_{id,i}^{\lambda_i}, g^r) = e\left(\prod_{i=1}^{t} Q_{id}^{f(i)\lambda_i}, g\right)^r = e\left(Q_{id}^{\sum_{i=0}^{t-1} f(i)\lambda_i}, g^r\right) = e(Q_{id}^x, g^r) = e(Q_{id}, Y)^r$.

2.3 Security Analysis

The overall security of our IBTD scheme relies on the well-known Decisional Bilinear Diffie-Hellman (DBDH) assumption and the random oracle model.

Definition 6 (DBDH Assumption). *The Decisional Bilinear Diffie-Hellman (DBDH) assumption in the bilinear map setting* $(q, \mathbb{G}, \mathbb{G}_T, e, g)$, *where* $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ *and* g *is the generator of* $\mathbb{G}$ *states that for any PPT distinguisher* $\mathcal{A}$ *the following advantage is negligible:* $\mathbf{Adv}_{\mathcal{A}}^{DBDH} = \left|Pr\left[\mathcal{A}(g, g^a, g^b, g^c, e(g,g)^{abc}) = 1\right] - Pr\left[\mathcal{A}(g, g^a, g^b, g^c, e(g,g)^z) = 1\right]\right|$, *where the probability is taken over the random choices of* $a, b, c, z \in \mathbb{Z}_q$ *and the random bits of* $\mathcal{A}$.

Theorem 1 (IBTD-IND-CCA)**.** *Our IBTD scheme from* Definition 5 *is IBTD-IND-CCA secure under the DBDH assumption in the random oracle model.*

Proof. For the proof of this theorem we refer to Appendix A.

Theorem 2 (IBTD-ANO-CCA)**.** *Our IBTD scheme from* Definition 5 *is IBTD- ANO-CCA secure under the DBDH assumption in the random oracle model.*

Proof. Since the proof of anonymity is similar to the proof of indistinguishability we only give a sketch. The simulator sets $Q_{id_\beta} = \left(g^b\right)^{\nu_\beta}$, for a random $\beta \in \{0,1\}$ and $Y = g^c$. It responds to the key derivation and decryption queries like in the indistinguishability proof. Upon finishing phase 1, $\mathcal{A}_{ano}$ outputs a message m and two identities id_0, id_1 it wants to be challenged on. The challenge ciphertext is computed as follows. $\mathcal{A}_{ano}$ sets $Q_{id_\beta} = (g^b)^{\nu_\beta}$ and $Y = g^c$ such that $e(Q_{id_\beta}, Y) = e(g^b, g^c)$ and $\kappa_{id} = Z$, where Z is the value from BDH instance. It chooses $s \in \{0,1\}^\ell$ uniformly at random. $\mathcal{A}_{BDH}$ gives $C^* = (g^a, s \oplus H_2(Z), m \oplus H_4(s)), \beta \in \{0,1\}$ to $\mathcal{A}_{ano}$. U is chosen uniformly at random by the encryption algorithm. V depends on the randomly chosen $s \in \{0,1\}^\ell$ and $H_2(Z)$. Since Z is randomly chosen, it does not depend on id_β. Also W is independent of id_β and therefore the ciphertext has the same distribution for both $\beta \in \{0,1\}$ and $\mathcal{A}_{ano}$ will have 0 advantage to distinguish between id_0 and id_1.

Theorem 3 (IBTD-SROB-CCA**).** *Our IBTD scheme from* Definition 5 *is unconditionally IBTD-SROB-CCA secure in the random oracle model.*

Proof. (Sketch) We show that IBTD-SROB-CCA property holds in the random oracle model. Assume $\mathcal{A}_{rob}$ be a IBTD-SROB-CCA adversary that is given the master key x. He can receive at most $t-1$ secret shares from $\mathcal{O}\texttt{KeyDer}$. We note, that H_1 is a map to $\mathbb{G}^*$, where all the outputs of this map are elements of order q. That means that the probability of finding two different identities $id_1 \neq id_2$ such that $H_1(id_1) = H_1(id_2)$ is negligible. Since $Y = g^x \in \mathbb{G}^*$ we have that κ_{id_1} and κ_{id_2} are not equal and of order q. Since H_3 maps into $\mathbb{Z}_q^*$, then $\kappa_{id_1}^r$ and $\kappa_{id_2}^r$ are different. Assuming H_2 as a random oracle, means that $H_2(\kappa_{id_1}^r) \neq H_2(\kappa_{id_2}^r)$. Decryption under different identities yields therefore two different values $\sigma_1 \neq \sigma_2$. In order for the ciphertext to be valid for both id's it should hold that $r = H_3(\sigma_1, m_1) = H_3(\sigma_2, m_2)$, which happens with negligible probability. It follows that our IBTD scheme is IBTD-SROB-CCA secure.

3 Threshold Public Key Encryption with Keyword Search

We start by defining the TPEKS syntax and its security goals. Towards the end of this section we propose a general transformation for building a secure TPEKS from anonymous and robust IBTD.

3.1 TPEKS Definitions and Security Model

Our model for TPEKS assumes a sender who encrypts keywords and at least t out of n servers, each equipped with its own trapdoor share, who participate in the search procedure. The latter represents the main difference to single-server based PEKS construction. We stress that the parameters t and n need not be fixed during the setup phase but can be chosen upon the generation of the trapdoors, which allows for greater flexibility.

Definition 7 (TPEKS). *A TPEKS scheme consists of the following five algorithms* $(\texttt{Setup}, \texttt{PEKS}, \texttt{Trpd}, \texttt{ShareTrpd}, \texttt{Test})$*:*

$\texttt{Setup}(1^k)$*: On input* 1^k*, it generates a private/public key pair* (sk, pk)*.*

$\texttt{PEKS}(pk, w)$*: On input* pk *and a keyword* w*, it outputs a PEKS ciphertext* Φ*.*

$\texttt{ShareTrpd}(pk, sk, w, t, n)$*: On input* (pk, sk) *and a keyword* w*, it generates a list of trapdoor shares* $T_w := \{(1, T_{w,1}), \ldots, (n, T_{w,n})\}$*.*

$\texttt{ShareTest}(pk, (i, T_{w,i}), \Phi)$*: On input* pk*, a trapdoor share* $T_{w,i}$*, and a PEKS ciphertext* Φ*, it outputs a test share* τ_i*.*

$\texttt{Test}(pk, \{\tau_i\}_{i\in\Omega}, \Phi)$*: On input* pk*, a set of test shares* $\{\tau_i\}_{i\in\Omega}$*,* $|\Omega| \geq t$ *and a PEKS ciphertext* $\Phi(w')$*, it outputs* 1 *if* Φ *encrypts* w*; otherwise it outputs* 0*.*

In Definition 8 we define TPEKS indistinguishability against chosen-ciphertext attacks, denoted by TPEKS-IND-CCA, aiming to protect privacy of the encrypted keywords in presence of an attacker who may learn up to $t-1$ trapdoor shares. Apart from the access to the trapdoor share oracle our scheme allows the adversary to issue up to $t-1$ queries to the test oracle.

Definition 8 (TPEKS Indistinguishability). *Let $\mathcal{B}_{ind}$ be a PPT adversary against the* TPEKS-IND-CCA *security of the TPEKS scheme, associated with the following experiment* $\mathbf{Exp}^{\text{TPEKS-IND-CCA}-b}_{\mathcal{B}_{ind}}(1^\lambda)$*:*

1. $(pk, sk) \xleftarrow{r} \texttt{Setup}(1^\lambda, t, n)$.
2. *Let* **List** *be a list storing a keyword w and a set $T_w = \{(1, T_{w,1}), \ldots, (n, T_{w,n})\}$, where $(i, T_{w,i})$ are the outputs of* $\texttt{ShareTrpd}(pk, sk, w, t, n)$ *algorithm.*
   At the beginning of the experiment the list is empty.
3. $(w_0, w_1, state) \xleftarrow{r} \mathcal{B}_{ind}^{\mathcal{O}\texttt{ShareTrpd}(\cdot), \mathcal{O}\texttt{Test}(\cdot)}(find, pk)$
4. *If $(w_0, T_0) \notin$* **List**, *run* $T_0 := \{(1, T_{w_0,1}), \ldots, (n, T_{w_0,n})\} \xleftarrow{r} \texttt{ShareTrpd}(pk, sk, w_0, t, n)$, *add (w_0, T_0) to* **List**
5. *If $(w_1, T_1) \notin$* **List**, *run* $T_1 := \{(1, T_{w_1,1}), \ldots, (n, T_{w_1,n})\} \xleftarrow{r} \texttt{ShareTrpd}(pk, sk, w_1, t, n)$, *add (w_1, T_1) to* **List**
6. $b \xleftarrow{r} \{0, 1\}$; $\Phi \xleftarrow{r} \texttt{PEKS}(pk, w_b)$
7. $b' \xleftarrow{r} \mathcal{B}_{ind}^{\mathcal{O}\texttt{ShareTrpd}(\cdot), \mathcal{O}\texttt{Test}(\cdot)}(guess, \Phi, state)$

   The experiment outputs 1 *if all of the following holds:*
   - $b' = b$
   - $\mathcal{B}_{ind}$ *asked at most $t-1$ queries to* $\mathcal{O}\texttt{ShareTrpd}(w_0, i)$ *and at most $t-1$ queries to* $\mathcal{O}\texttt{ShareTrpd}(w_1, i)$
   - $\mathcal{B}_{ind}$ *didn't query* $\mathcal{O}\texttt{Test}(w_0, \Phi)$ *or* $\mathcal{O}\texttt{Test}(w_1, \Phi)$

where the two oracles are defined as follows:

$\mathcal{O}\texttt{ShareTrpd}(w, i)$*: On input (w, i) check whether $(w, T_w) \in$* **List**. *If so, parse T_w as $\{(1, T_{w,1}), \ldots, (n, T_{w,n})\}$ and output $(i, T_{w,i})$. If $(w, T_w) \notin$* **List**, *run* $T_w \xleftarrow{r} \texttt{ShareTrpd}(pk, sk, w, t, n)$. *Add (w, T_w) to* **List**, *output $(i, T_{w,i})$.*

$\mathcal{O}\texttt{Test}(w, \Phi)$*: On input (w, Φ) check whether $(w, T_w) \in$* **List**. *If so, parse T_w as $\{(1, T_{w,i}), \ldots, (n, T_{w,n})\}$, compute* $\tau_i \xleftarrow{r} \texttt{ShareTest}(pk, (i, T_{w,i}), \Phi)$. *Take at least t-out-of-n test shares τ_i, run* $\texttt{Test}(pk, \{\tau_i\}_{i \in \Omega}, \Phi)$, *where $|\Omega| \geq t$, output* 1 *or* 0. *If $(w, T_w) \notin$* **List**, *compute* $T_w \xleftarrow{r} \texttt{ShareTrpd}(pk, sk, w, t, n)$, *add (w, T) to the* **List**. *Compute* $\tau_i \xleftarrow{r} \texttt{ShareTest}(pk, (i, T_{w,i}), \Phi)$, *where $i \in [n]$. Take at least t-out-of-n test shares τ_i, run* $\texttt{Test}(pk, \{\tau_i\}_{i \in \Omega}, \Phi)$, *output* 1 *or* 0.

The advantage of $\mathcal{B}_{ind}$ is defined as

$$\mathbf{Adv}^{\text{TPEKS-IND-CCA}-b}_{\mathcal{B}_{ind}}(1^k) = \Big| \Pr\left[\mathbf{Exp}^{\text{TPEKS-IND-CCA}-1}_{\mathcal{B}_{ind}}(1^\lambda) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{TPEKS-IND-CCA}-0}_{\mathcal{B}_{ind}}(1^\lambda) = 1\right] \Big|.$$

The scheme is TPEKS-IND-CCA *secure if* $\mathbf{Adv}^{\text{TPEKS-IND-CCA}}_{\mathcal{B}_{ind}}$ *is negligible.*

In Definition 9 we model computational consistency of TPEKS schemes, by extending the corresponding property for PEKS schemes from [1, Sect. 3]. Our definition allows the adversary to test polynomially-many keyword-ciphertext pairs, thus modeling chosen-ciphertext attacks, and accounts for the threshold setting by allowing the adversary to learn up to $t-1$ trapdoor shares for keywords that will be used to mount a successful attack.

Definition 9 (TPEKS Consistency). *Let $\mathcal{B}_c$ be a PPT adversary against the* TPEKS-CONS *security of the TPEKS scheme, associated with the following experiment* $\mathbf{Exp}_{\mathcal{B}_c}^{\text{TPEKS-CONS}}(1^\lambda)$*:*

1. $(pk, sk) \xleftarrow{r} \texttt{Setup}(1^\lambda, t, n)$
   Let **List** *be list storing a keyword w and a set $T_w = \{(1, T_{w,1}), \ldots, (n, T_{w,n})\}$, where $(i, T_{w,i})$ are the outputs of* $\texttt{ShareTrpd}(pk, sk, w, t, n)$ *algorithm.*
   At the beginning of the experiment the list is empty.
2. $(w, w') \xleftarrow{r} \mathcal{B}_c^{\mathcal{O}\texttt{ShareTrpd}(\cdot), \mathcal{O}\texttt{Test}(\cdot)}(pk)$
3. $\Phi \xleftarrow{r} \texttt{PEKS}(pk, w)$
4. *If* $(w', (i, T_{w',i})) \notin$ **List** *run* $T_{w'} := \{(1, T_{w',1}), \ldots, (n, T_{w',n})\} \xleftarrow{r} \texttt{ShareTrpd}(pk, sk, w', t, n)$ *and add* (w', T') *to* **List**
   The experiment outputs 1 *if all of the following holds:*
   - $w \neq w'$
   - $\mathcal{B}_c$ *asked at most* $t-1$ *queries to* $\mathcal{O}\texttt{ShareTrpd}(w', i)$ *and at most* $t-1$ *queries to* $\mathcal{O}\texttt{ShareTrpd}(w, i)$.

where the two oracles are defined as follows:

$\mathcal{O}\texttt{ShareTrpd}(w, i)$*: On input* (w, i) *check whether* $(w, T_w) \in$ **List**. *If so, parse* T *as* $\{(1, T_{w,1}), \ldots, (n, T_{w,n})\}$ *and output* $(i, T_{w,i})$. *If* $(w, T_w) \notin$ **List**, *run* $T \xleftarrow{r} \texttt{ShareTrpd}(pk, sk, w, t, n)$. *Add* (w, T_w) *to* **List**, *output* $(i, T_{w,i})$.

$\mathcal{O}\texttt{Test}(w, \Phi)$*: On input* (w, Φ) *check whether* $(w, T_w) \in$ **List**. *If so, parse* T_w *as* $\{(1, T_{w,i}), \ldots, (n, T_{w,n})\}$, *compute* $\tau_i \xleftarrow{r} \texttt{ShareTest}(pk, (i, T_{w,i}), \Phi)$. *Take at least t-out-of-n test shares* τ_i, *run* $\texttt{Test}(pk, \{\tau_i\}_{i \in \Omega}, \Phi)$, *where* $|\Omega| \geq t$, *output* 1 *or* 0. *If* $(w, T_w) \notin$ **List**, *compute* $T_w \xleftarrow{r} \texttt{ShareTrpd}(pk, sk, w, t, n)$, *add* (w, T_w) *to the* **List**. *Compute* $\tau_i \xleftarrow{r} \texttt{ShareTest}(pk, (i, T_{w,i}), \Phi)$, *where* $i \in [n]$. *Take at least t-out-of-n test shares* τ_i, *run* $\texttt{Test}(pk, \{\tau_i\}_{i \in \Omega}, \Phi)$, *output* 1 *or* 0.

The advantage of $\mathcal{B}_c$ is defined as

$$\mathbf{Adv}_{\mathcal{B}_c}^{\text{TPEKS-CONS}}(1^k) = Pr\left[\mathbf{Exp}_{\mathcal{B}_c}^{\text{TPEKS-CONS}}(1^\lambda) = 1\right].$$

The scheme is TPEKS-CONS *secure if* $\mathbf{Adv}_{\mathcal{B}_c}^{\text{TPEKS-CONS}}$ *is negligible.*

Note: It is obvious that the correctness property of our TPEKS is satisfied. Correctness ensures that the test algorithm always outputs the correct answer.

3.2 A General TPEKS Construction from an Anonymous and Robust IBTD Scheme

The design rationale of our TPEKS construction from IBTD follows the transformation from [1] for single-server PEKS from any anonymous and robust IBE, where the PEKS private/public key pair (sk, pk) corresponds to the IBE master private/public key pair (msk, mpk) and a PEKS ciphertext $\Phi = (C, R)$ for some keyword w is computed by encrypting some random message R using keyword w as an identity. The search procedure for Φ decrypts C using the trapdoor T_w and compares the decrypted message with R.

Our IBTD-to-TPEKS transformation, detailed in Definition 10, treats TPEKS keywords as IBTD identities and performs distributed search on TPEKS ciphertexts using the IBTD threshold decryption procedure. While we describe only general IBTD-to-TPEKS transform, we remark that a concrete TPEKS instantiation can be easily obtained using our concrete IBTD construction from Definition 5.

Definition 10 (IBTD-to-TPEKS Transform).

$\texttt{Setup}(1^k)$*: On input a security parameter* 1^k*, it runs the parameter generation algorithm of the IBTD scheme* $(msk, mpk) \xleftarrow{r} \texttt{Setup}(n, t, 1^k)$ *and outputs* $sk = msk$ *and* $pk = mpk$*.*

$\texttt{PEKS}(pk, w)$*: On input a public key* pk *and a keyword* w*, it runs the encryption algorithm of the IBTD scheme* $C \xleftarrow{r} \texttt{Enc}(pk, w, R)$*, where* $R \xleftarrow{r} \{0,1\}^k$ *is picked randomly. It returns the TPEKS ciphertext* $\Phi = (C, R)$*.*

$\texttt{ShareTrpd}(pk, sk, w, t, n)$*: On input* (pk, sk, w, t, n)*, it runs the key derivation procedure* $(T_w, \{i, T_{w,i}\}) \xleftarrow{r} \texttt{KeyDer}(pk, sk, w, t, n)$ *of the IBTD scheme where* sk *is the master key* msk *and keyword* w *is used as id. The trapdoor* T_w *associated to* w *corresponds to* sk_{id} *generated by the IBTD. It outputs* $\{(1, T_{w,1}), \ldots, (n, T_{w_n})\}$ *which correspond to* $\{(i, sk_{id,i}), \ldots, (n, sk_{id,n})\}$ *of the IBTD scheme.*

$\texttt{ShareTest}(pk, (i, T_{w,i}), \Phi)$*: On input* pk*, a trapdoor share* $(i, T_{w,i})$*, and a TPEKS ciphertext* $\Phi = (C, R)$*, it outputs* $\tau_i \xleftarrow{r} \texttt{ShareDec}(pk, (i, T_{w,i}), \Phi)$ *using the distributed decryption algorithm of the IBTD scheme.*

$\texttt{Test}(pk, \{\tau_{w,i}\}_{i \in \Omega}, (C, R))$*: On input of at least* t *test shares* $\{\tau_{w,i}\}_{i \in \Omega}$*,* $|\Omega| \geq t$ *and* (C, R) *is a TPEKS ciphertext, it computes* $R' \xleftarrow{r} \texttt{Dec}(pk, \{\delta_i\}_{i \in \Omega}, C)$ *and outputs 1 if* $R' = R$*, and 0 otherwise.*

3.3 Security Analysis

With regard to security, in Theorem 4, we establish similar implications for TPEKS indistinguishability as in [1], namely we rely on the anonymity and robustness of the IBTD scheme. In Theorem 5 we show that for TPEKS consistency the underlying IBTD scheme must not only be indistinguishable but also robust. This contrasts to [1,2] where IBE robustness was not required for

PEKS consistency and is mainly due to the fact that our definition of consistency allows adaptive queries to the distributed test procedure, which was not an issue in [1,2].

Theorem 4 (TPEKS-IND-CCA). *If IBTD scheme is* IBTD-ANO-CCA *secure then the obtained TPEKS scheme in* Definition 10 *is* TPEKS-IND-CCA *secure*

Proof. We use a TPEKS-IND-CCA adversary $\mathcal{B}_{ind}$ against the TPEKS scheme to construct a simulator $\mathcal{S}$ that breaks the assumed IBTD-ANO-CCA and IBTD-SROB-CCA properties of the IBTD scheme. That is, $\mathcal{S}$ acts as $\mathcal{A}_{ano}$ attacking the IBTD-ANO-CCA security and as $\mathcal{A}_{rob}$ against IBTD-SROB-CCA. The simulation of the view of $\mathcal{B}_{ind}$ happens in several games. The initial game is the Game0 which describes the real attack. First the challenger runs the `Setup` of the TPEKS scheme on input a security parameter λ, threshold parameter t and number of servers n. The challenger gives $\mathcal{B}_{ind}$ the public key pk. If $\mathcal{B}_{ind}$ submits a pair of keywords w_0, w_1, the challenger computes a target ciphertext Φ^*. $\mathcal{B}_{ind}$ issues trapdoor share and test queries on the PEKS ciphertext Φ. The first game (Game1) differs from the previous one by simulation of trapdoor share queries. If these shares are involved in computing the PEKS ciphertext, the simulator modifies the challenge ciphertext. The rest of Game 0 remains unmodified. The simulation is distributed in two subcases. In Case 1 holds $C \neq C^*, w^* \neq w_b$, which invokes $\mathcal{A}_{ano}$ to simulate the queries on identities $id \neq id_b$. In Case 2 holds $C = C^*$, $w^* = w_b$ for $b \in \{0,1\}$ where $\mathcal{A}_{rob}$ is invoked for the simulation of queries on id^*. In Case 1, $\mathcal{A}_{ano}$ aborts the game if $\mathcal{B}_{ind}$ issues more than $t-1$ queries on $w^* \neq w_b$. In Case 2, where holds $C = C^*$, $\mathcal{A}_{rob}$ aborts the game if $\mathcal{B}_{ind}$ issues more than $t-1$ queries on $w^* = w_b = w_{1-b}$. The second game differs from Game 1 by simulation of test queries. In Case 1, $\mathcal{A}_{ano}$ aborts the game if $\mathcal{B}_{ind}$ issues queries on w_b. In Case 2, where holds $C = C^*$, $\mathcal{A}_{rob}$ aborts the game if $\mathcal{B}_{ind}$ issues queries on $w^* = w_b = w_{1-b}$. Each of the simulation steps looks as follows:

Setup: $\mathcal{S}$ is given as input mpk of IBTD. It sets TPEKS public key pk equal to mpk. We assume that a set of $t-1$ servers have been corrupted. When $\mathcal{B}_{ind}$ issues trapdoor share and decryption queries on input (w,i) and (w,Φ) respectively, where $w \neq w_b$, and $\Phi = (C,R), \Phi^* = (C^*,R^*)$, $\mathcal{S}$ distinguishes between two cases - Case 1: $C \neq C^*, w \neq w_b$ where $\mathcal{A}_{ano}$ is invoked and Case 2: $C = C^*$, $w = w_b$ for $b \in \{0,1\}$ where $\mathcal{A}_{rob}$ is invoked.

Queries to $\mathcal{O}$`ShareTrpd`: Let (w,i) be a trapdoor share query issued by $\mathcal{B}_{ind}$. $\mathcal{S}$ sets $w \leftarrow id$ and queries its oracle $\mathcal{O}\texttt{KeyDer}(w,i)$. The oracle outputs $(i, sk_{w,i})$, which $\mathcal{S}$ sets equal to $(i, T_{w,i})$ and returns it to $\mathcal{B}_{ind}$. In Case 1, the simulator represented by $\mathcal{A}_{ano}$ aborts if $\mathcal{B}_{ind}$ issued more than $t-1$ queries to $\mathcal{O}\texttt{ShareTrpd}(w_0,i)$ and to $\mathcal{O}\texttt{ShareTrpd}(w_1,i)$. In Case 2, simulator represented by $\mathcal{A}_{rob}$ aborts if $w_0 = w_1$ or $(w_0, T_0, I_0) \notin$ **List**, or $(w_1, T_1, I_1) \notin$ **List**, or $|I_0| \geq t, |I_1| \geq t$, where $T_b = \{(1, T_{w_b,1}), \ldots, (n, T_{w_b,n})\}, I_b \subset [1,n], b \in \{0,1\}$.

Queries to $\mathcal{O}$`Test`: $\mathcal{B}_{ind}$ issues test queries on (w,Φ). $\mathcal{S}$ sets $w \leftarrow id, \Phi = (C,R)$ and queries its oracle $\mathcal{O}\texttt{Dec}(w,C)$. The oracle outputs m. $\mathcal{S}$ sets $R \leftarrow m$ and

returns R to $\mathcal{B}_{ind}$. In Case 1, $\mathcal{S}$ aborts simulation if $\mathcal{B}_{ind}$ queried $\mathcal{O}\texttt{Test}(w_0, \Phi^*)$ or $\mathcal{O}\texttt{Test}(w_1, \Phi^*)$, where $\Phi^* = (C^*, R^*)$. In Case 2, $\mathcal{S}$ aborts if $w_0 = w_1$ or $(w_0, T_0, I_0) \notin \textbf{List}$, or $(w_1, T_1, I_1) \notin \textbf{List}$, or $|I_0| \geq t, |I_1| \geq t$, where $T_b = \{(1, T_{w_b,1}), \ldots, (n, T_{w_b,n})\}$ and $I_b \subset [1, n]$, for $b \in \{0, 1\}$.

Challenge: $\mathcal{B}_{ind}$ outputs two identities w_0, w_1 and $R^* \xleftarrow{r} \{0,1\}^\ell$. $\mathcal{S}$ responds with IBTD ciphertext $\Phi^* = (C^*, R^*)$, where $C^* \leftarrow \texttt{Enc}(pk, w_b, R^*)$, for $b \in \{0, 1\}$ and $R^* \xleftarrow{r} \{0, 1\}$. $\mathcal{S}$ returns its ciphertext $C^* \leftarrow Enc(mpk, w_b, R^*)$.

Analysis: In Case 1: Simulator $\mathcal{S}$ is represented by $\mathcal{A}_{ano}$. Let q_{ts}, q_t be the number of issued trapdoor share queries on different id's. We assume that $\mathcal{B}_{ind}$ corrupts $t-1$-out-of-n servers with index $i_1, \ldots, i_{t-1}$. The probability that $\mathcal{S}$ corrupts a server j with $j \in \{i_1, \ldots, i_{t-1}\}$ is $1/\binom{n}{t-1}$. Let E denote the event that $\mathcal{B}_{ind}$ wins the indistinguishability experiment from Definition 8. Let $E1$ denote the event that $\mathcal{B}_{ind}$ wins Game 1. It holds $\frac{1}{2}\mathbf{Adv}^{\text{TPEKS-IND-CCA}}_{\mathcal{B}_{ind}}(1^\lambda) = Pr[E] - 1/2 \geq 1/\binom{n}{t-1}\frac{1}{q_t s}(Pr[E1] - 1/2)$. Let $E2$ denote the event that $\mathcal{B}_{ind}$ wins Game 2, then holds: $Pr[E1] - 1/2 \geq \frac{1}{q_t}(Pr[E2] - 1/2)$

$\Leftrightarrow \frac{1}{2}\mathbf{Adv}^{\text{TPEKS-IND-CCA}}_{\mathcal{B}_{ind}}(1^\lambda) = Pr[E] - 1/2 \geq 1/\binom{n}{t-1}\frac{1}{q_{ts}}\frac{1}{q_t}(Pr[E2] - 1/2)$ In Case 2: Simulator $\mathcal{S}$ is represented by $\mathcal{A}_{rob}$. Let q'_{ts}, q'_t be the number of issued trapdoor share queries on different id's. We assume that $\mathcal{B}_{ind}$ corrupts $t-1$-out-of-n servers with index $i_1, \ldots, i_{t-1}$. The probability that $\mathcal{S}$ corrupts a server j with $j \in \{i_1, \ldots, i_{t-1}\}$ is $1/\binom{n}{t-1}$. Let E denote the event that $\mathcal{B}_{ind}$ wins the indistinguishability experiment from Definition 8. Let $\widetilde{E1}$ denote the event that $\mathcal{B}_{ind}$ wins Game 1. It holds $\frac{1}{2}\mathbf{Adv}^{\text{TPEKS-IND-CCA}}_{\mathcal{B}_{ind}}(1^\lambda) = Pr[E] - 1/2 \geq 1/\binom{n}{t-1}\frac{1}{q_{ts}}(Pr[\widetilde{E1}] - 1/2)$. Let $\widetilde{E2}$ denote the event that $\mathcal{B}_{ind}$ wins Game 2, then holds: $Pr[E1] - 1/2 \geq \frac{1}{q_t}(Pr[\widetilde{E2}] - 1/2)$

$\Leftrightarrow \frac{1}{2}\mathbf{Adv}^{\text{TPEKS-IND-CCA}}_{\mathcal{B}_{ind}}(1^\lambda) = Pr[E] - 1/2 \geq 1/\binom{n}{t-1}\frac{1}{q_{ts}}\frac{1}{q_t}(Pr[\widetilde{E2}] - 1/2)$ The total advantage of $\mathcal{B}_{ind}$ is given by

$$\begin{aligned}
&\frac{1}{2}\mathbf{Adv}^{\text{TPEKS-IND-CCA}}_{\mathcal{B}_{ind}}(1^\lambda) = Pr[E|Case1] + Pr[E|Case2] = 2Pr[E] - 1 \\
&\geq 1/\binom{n}{t-1}\frac{1}{q_{ts}}\frac{1}{q_t}(Pr[E2] - 1/2) + 1/\binom{n}{t-1}\frac{1}{q_{ts}}\frac{1}{q_t}(Pr[\widetilde{E2}] - 1/2) \\
&= 1/\binom{n}{t-1}\frac{1}{q_{ts}}\frac{1}{q_t}(Pr[E2] + Pr[\widetilde{E2}] - 1)
\end{aligned}$$

Theorem 5 (TPEKS Consistency). *If IBTD scheme is* IBTD-IND-CCA *secure then the obtained TPEKS scheme in Definition 10 is* TPEKS-CONS *secure.*

Proof. For the proof of this theorem we refer to Appendix B.

3.4 Application to Cloud Setting

In the introduction we mentioned the applicability of TPEKS to distributed cloud storage, as a solution to mitigate the single point of trust with regard to

the search procedure and the insecurity of single-server PEKS schemes against keyword guessing attacks. Together with its properties, TPEKS seems to be particularly attractive for this application, as detailed in the following two scenarios.

In the first use case we assume an user who wishes to upload his data files on the cloud servers to have access to these file at a later point in time. Assume an user who uploads to n cloud servers m encrypted data files with m PEKS ciphertexts where each of them is encrypted on l different keywords $w_1, \ldots, w_l$, i.e. PEKS ciphertext for the j-th file is given by $\{\Phi(pk, j, w_{i_1}, \ldots, w_{i_d})\}_{i_d \in [l], j \in [m]}$. When the user wants to download files which contain a keyword w_i, he computes trapdoor shares for each server on that keyword, i.e. he sends trapdoors $T_{w_i,1}, \ldots, T_{w_i,n}$ on w_i to the n servers, where $i \in \{1, \ldots, l\}$ denotes the index one of the l keywords. Each cloud server computes test shares taking as input the different PEKS ciphertext for each data file and the trapdoor for the k-th server $\tau_{k,j} \leftarrow \texttt{ShareTest}(\Phi(pk, j, w_{i_1}, \ldots, w_{i_d}), T_{w_i,k})$, where j denotes the index of PEKS ciphtertext for file j and $i_1, \ldots, i_d \in [l]$ is a set of l keywords. Each server outputs m test shares $\{\tau_{k,j}\}_{j \in [m]}$, such that the user obtains in total $m \times n$ different test shares. For the ease of analysis we observe 2 cloud servers and therefore $2m$ different test shares. Since the user does not know which test shares belong to which files, he has to run m^2 test algorithms, namely $\texttt{Test}(\tau_{1,j}, \tau_{2,j'}, pk)$ where $j, j' \in [m]$. This scenario has a total complexity of $\mathcal{O}(m^2)$. This scenario guarantees privacy of the user, because the trapdoors do not reveal anything about the keywords and the servers do not learn anything about the keywords since they take the ciphertexts keywords and the trapdoors without getting any information about the content of the inputs.

To reduce the complexity the user could use random indices for each uploaded PEKS ciphertexts on keywords. That means that he would need to upload $(r_\chi, \{\Phi(pk, j, w_{i_1}, \ldots, w_{i_d})\}_{i_d \in [l], j \in [m]})$ to each server, where r_χ denotes a random index for the ciphertext $\Phi(\cdot)$ on a set of keywords $w_{i_1}, \ldots, w_{i_d}$, with $i_1, \ldots, i_d \in [l]$. The user has to remember $(r_\chi, w_{i_1}, \ldots, w_{i_d})$ for later use. Finally if he wants to download data files with keyword $w_i, i \in [l]$ he computes n trapdoor shares for all n servers and sends them together with the index r_χ to the server. Each server compares whether the received randomness belongs to one of the stored PEKS ciphertexts. If so each server computes trapdoor shares $(r_\chi, \tau_{k,j}) \leftarrow \texttt{ShareTest}((r_\chi, \Phi(pk, j, w_{i_1}, \ldots, w_{i_d}), T_{w_i,k})$ and sends them to the user. Upon receiving the test shares together with the randomness, the user can recognize which test shares to which file and can be used to run test algorithm. If the output of the algorithm is 1, the user sends the randomness to one of the servers to get access for the download of a file. The complexity in this scenario can be reduced to the linear size $\mathcal{O}(m)$. This example still guarantees privacy of the user because a randomness is prepared for a set of keywords, such that the files are unlinkable to the keywords.

As a second use case we consider a sender who sends a set of m encrypted messages with PEKS ciphertexts $\Phi(pk, j, w_1, \ldots, w_l)$ on l different keywords to the n servers, where $j \in [m]$, for a recipient who owns the public key pk. The receipient computes trapdoor shares for each server on a set of required

keywords and sends them to the servers. Each of the servers computes m different test shares and return them to the user. The user needs to find the sets of the test shares for the same encrypted messages. To do so he needs to try m^n combinations which gives us the complexity of this scenario, $\mathcal{O}(m^n)$. To reduce the complexity the user could compute $l \times n$ trapdoor shares for the l shares and send them together with a randomness $\{r_\chi\}_{\chi \in [l]}$ of each keyword to the servers. Each of the servers runs m test share algorithms and returns $m \times l$ test shares together with randomness such that the user can combine the test shares with the corresponding randomness. The complexity of this scenario is reduced to $\mathcal{O}(l \times m)$. The privacy of user regarding the keywords remains guaranteed, because no one can learn the queried keywords.

A Proof of Theorem 1

Proof. Let $\mathcal{A}_{ind}$ be an adversary that defeats the IBTD-IND-CCA security of the IBTD scheme, and $\mathcal{A}_{BDH}$ let be an adversary for the BDH problem. $\mathcal{A}_{BDH}$ are given BDH parameters $(\mathbb{G}, \mathbb{G}_T, e, q)$ and a random instance $(g, g^a, g^b, g^c, e(g,g)^{abc})$ of the BDH problem for these parameters. That means $g \xleftarrow{r} \mathbb{G}^*$ and $a, b, c \xleftarrow{r} \mathbb{Z}_q^*$ are random. Let $Z = e(g,g)^z \in \mathbb{G}_T$, where $\mathcal{A}_{ind}$'s aim is to decide whether $z = abc$, or z is a randomly chosen value from $\mathbb{Z}_q^*$. The definition of CCA security allows the adversary to obtain the secret share associated with any identity id_i, where $i \in [q_1, \ldots, q_m]$ of her choice and $\mathcal{A}_{ind}$ is challenged on a public key id^* of her choice. $\mathcal{A}_{ind}$ issues queries $q_1, \ldots, q_m$, where q_i, for $i \in [m]$ is one the key derivation or decryption queries. $\mathcal{A}_{BDH}$ uses $\mathcal{A}_{ind}$ to find d as follows:

Setup: The simulator $\mathcal{A}_{BDH}$ generates IBTD master public key $mpk = (\mathbb{G}, \mathbb{G}_T, q, g, e, H_1, H_2, H_3, H_4, Y)$ by setting $Y = g^c$ and $Q_{id} = g^b$. $H_1, \ldots, H_4$ are random oracles controlled by $\mathcal{A}_{BDH}$. The queries to the oracles are described as follows. $\mathcal{A}_{BDH}$ gives mpk to $\mathcal{A}_{ind}$. $\mathcal{A}_{ind}$ issues q_{H_1}, q_{ks} queries on an id to H_1 oracle, the key derivation oracle and a query on id^* to the both oracles in the challenge stage, respectively.

H_1 **Oracle Queries:** Let H_1 List be a list used for storing the results of queries to the H_1 oracle, (id, Q_{id}). Whenever H_1 is queried at $id \in \{0,1\}^\ell$, $\mathcal{A}_{BDH}$ does the following: If $(id, Q_{id}) \in H_1$List, it returns Q_{id} to $\mathcal{A}_{ind}$. For $id \neq id^*$, $\mathcal{A}_{BDH}$ sets $Q_{id} = H_1(id) = \left(g^b\right)^\gamma$ for a random $\gamma \xleftarrow{r} \mathbb{Z}_q$, where g^b is given from the BDH instance. If $id = id^*$, $\mathcal{A}_{BDH}$ computes $Q_{id} = g^\gamma$, where $\gamma \xleftarrow{r} \mathbb{Z}_q$, adds it to the H_1List and returns Q_{id} to $\mathcal{A}_{ind}$.

H_2 **Oracle Queries:** Let H_2 List be a list consisting of all pairs $(\kappa_{id}, H_2(\kappa_{id}))$. When $\mathcal{A}_{ind}$ queries on input $\kappa_{id} = e(Q_{id}, Y)^r$, $\mathcal{A}_{BDH}$ checks whether the queried value is in the H_2List, if so it returns $H_2(\kappa_{id})$. Otherwise it chooses a random $H_2' \in \{0,1\}^\ell$ and gives $H_2(\kappa_{id}) = H_2'$ to $\mathcal{A}_{ind}$.

H_3 **Oracle Queries:** Let H_3 List consisting of elements $(\sigma, m, H_3(\sigma, m))$, where $\sigma \in_r \{0,1\}^l$. When $\mathcal{A}_{ind}$ issues queries on input (σ, m) it invokes $\mathcal{A}_{BDH}$ that checks whether $(\sigma, m) \in H_3$List. If so it returns the corresponding $H_3(\sigma, m)$.

Otherwise $\mathcal{A}_{BDH}$ chooses a random $H_3' \in \mathbb{Z}_q^*$ and sets $H_3' = H_3(\sigma, m)$ that it gives to $\mathcal{A}_{ind}$

H_4 **Oracle Queries:** Let H_4 List consist of all pairs $(\sigma, H_4(\sigma))$. When $\mathcal{A}_{ind}$ issues a query on $(\sigma, \cdot)$, $\mathcal{A}_{BDH}$ checks, whether $\sigma \in H_4$List. If so, it returns the corresponding $H_4(\sigma)$, otherwise it chooses $H_4' \in \{0,1\}^\ell$ and sets $H_4' = H_4(\sigma)$ and gives H_4' to $\mathcal{A}_{ind}$.

Phase 1: $\mathcal{A}_{ind}$ issues up to q_m queries to the key derivation and decryption oracles.

Queries to $\mathcal{O}\texttt{KeyDer}(id, i)$: *For* $id \neq id^*$: When $\mathcal{A}$ submits a key derivation query on input (id, i), $\mathcal{A}_{BDH}$ checks whether $(id, S) \in$ **List**. If so, $\mathcal{A}_{BDH}$ returns the corresponding secret share $sk_{id,i}$ for index i to $\mathcal{A}_{ind}$. If $(id, S) \notin$ **List**, $\mathcal{A}_{BDH}$ simulates the key shares as follows: For the $t-1$ corrupted servers with indices $i_1, \ldots, i_{t-1}$, $\mathcal{A}_{BDH}$ chooses $t-1$ random values χ_i, such that $Q_{id}^{f(i)} = \chi_i$. If $id = id^*$, $\mathcal{A}_{BDH}$ picks a random $\chi_i \in \mathbb{G}$ and returns it to $\mathcal{A}_{ind}$. If $\mathcal{A}_{ind}$ issues more than $t-1$ queries on id^*, $\mathcal{A}_{BDH}$ aborts the simulation. If $id \neq id^*$, $\mathcal{A}_{BDH}$ chooses $\gamma \xleftarrow{r} \mathbb{Z}_q^*$, adds $Q_{id} = (g^b)^\gamma$ to the H_1List and outputs to $\mathcal{A}_{ind}$

Queries to $\mathcal{O}\texttt{Dec}(id, C)$: If $id = id^*$, $\mathcal{A}_{BDH}$ aborts the simulaton. For $id \neq id^*$, $\mathcal{A}_{ind}$ issues a decryption query on input (id, C) to its decryption oracle, where $C \neq C^*$. $\mathcal{A}_{BDH}$ simulates the decryption oracle without knowing the decryption shares. It does the following:

1. $\mathcal{A}_{BDH}$ checks whether $(id, H_1(id)) \in H_1$list. If so, it fixes the corresponding $Q_{id} = H_1(id)$.
2. It computes $\kappa_{id} = e(Q_{id}, Y)^r$, using the fixed Q_{id} from H_1List and $Y = g^c$.
3. To determine the corresponding r, the simulator searches the H_3List. Choosing each triple (σ, m, r), the simulator compares, whether $g^r = U$, where U is given from the received ciphertext. After fixing the matching r, $\mathcal{A}_{BDH}$ receives (σ, m).
4. Using the fixed r and σ from the previous step the simulator computes $\kappa_{id} = e(Q_{id}, Y)^r$ and searches the H_2List for the corresponding value $H_2(k_{id})$. It checks, whether $V = \sigma \oplus H_2(\kappa_{id})$.
5. Taking σ, m from step 3. the simulator searches H_4List for the corresponding $H_4(\sigma)$ entry. Upon finding the matching value it checks whether $W = m \oplus H_4(\sigma)$. If one of the computations in the above 5 steps fails, the simulator aborts the game. Otherwise if all 5 steps finished successful, $\mathcal{A}_{BDH}$ returns m to $\mathcal{A}_{ind}$.

Challenge Ciphertext: At some point $\mathcal{A}_{ind}$ outputs two messages m_0, m_1 and an identity id on which it wishes to be challenged. $\mathcal{A}_{BDH}$ simulates the ciphertext as follows. He replaces U by g^a from its BDH instance. It sets $Q_{id} = g^b$ and $Y = g^c$ such that $e(Q_{id}, Y) = e(g^b, g^c)$ and $\kappa_{id} = Z$, where Z is the value from BDH instance. It chooses $s \in \{0,1\}^\ell$ uniformly at random. $\mathcal{A}_{BDH}$ gives $C^* = (g^a, s \oplus H_2(Z), m_\beta \oplus H_4(s)), \beta \in \{0,1\}$ as challenge to $\mathcal{A}$.

Phase 2: $\mathcal{A}_{ind}$ issues additional queries as in Phase 1, to which $\mathcal{A}_{BDH}$ responds as before

Guess: Eventually $\mathcal{A}_{ind}$ outputs β' as its guess for β. Algorithm $\mathcal{A}_{BDH}$ outputs β' as its guess for β

Analysis: Let q_{H_1}, q_{ks}, q_d be the number of issued H_1 oracle queries, key share queries, decryption queries on an identity id, respectively and $\mathcal{A}_{ind}$ issues one query on challenge id to the three oracles. The probability that $\mathcal{A}_{BDH}$ guesses the correct challenge id is $\delta_1 := \frac{1}{q_{H_1}+q_{ks}+1}$. It aborts the simulation with probability δ if $id = id^*$, which has already been queried either to H_1 oracle or to $\mathcal{O}\texttt{Dec}$. We assume that $\mathcal{A}_{ind}$ corrupts $t-1$-out-of-n servers with index $i_1, \ldots, i_{t-1}$. The probability that $\mathcal{A}_{BDH}$ matches a server j with $j \in \{i_1, \ldots, i_{t-1}\}$ is $\delta_2 := \frac{1}{\binom{n}{t-1}}$. If $\mathcal{A}_{ind}$ issues more than $t-1$ secret share queries on the same identity id, $\mathcal{A}_{BDH}$ aborts the simulation. The simulator aborts the decryption simulation in the non challenge phase with negligible probability δ_3, where $\delta_3 \in [0,1]$. The probability that it does not abort in the first phase is $1-(\delta_1+\delta_3)$. The simulator aborts in the challenge phase if $\mathcal{A}_{ind}$ issues more than $t-1$ queries to $\mathcal{O}\texttt{Dec}$ and $\mathcal{O}\texttt{KeyDer}$ on the challenge identity id^*. It also stops the simulation if $Z = e(g,g)^{abc}$, where δ_4 is the probability, that the equation holds. The probability for abortion during the key derivation or decryption queries on id^* is $\delta_2 + \delta_2\delta_3$. The probability that it does not abort in the challenge step is $1-(\delta_2+\delta_2\delta_3+\delta_4)$. Therefore the probability that $\mathcal{A}_{BDH}$ does not abort during the simulation is $(1-(\delta_1+\delta_3))\,(1-(\delta_2+\delta_2\delta_3+\delta_4)) = 1-\tilde{\delta}$, where $\tilde{\delta}$ is negligible. Advantage of $\mathcal{A}_{ind}$ is given by

$$\mathbf{Adv}_{\mathbf{A_{BDH}}} \geq \mathbf{Adv}^{\text{IBTD-IND-CCA}}_{\mathcal{A}_{\mathbf{ind}}} = \mathbf{1} - \tilde{\delta}$$

It follows that $\mathbf{Adv}_{\mathbf{A_{BDH}}} > \mathbf{1} - \tilde{\delta}$ is non-negligible which is a contradiction to the assumption. Therefore we follow, that the advantage of $\mathcal{A}_{ind}$ is negligible.

B Proof of Theorem 5

Proof. We use a TPEKS-CONS adversary $\mathcal{B}_c$ to construct a simulator $\mathcal{S}$ that breaks the IBTD-IND-CCA and IBTD-SROB-CCA properties of IBTD. That is, $\mathcal{S}$ acts as $\mathcal{A}_{ind}$ against IBTD-IND-CCA and as $\mathcal{A}_{rob}$ against IBTD-SROB-CCA. The initial game is the Game 0 which describes the real attack. First the challenger runs the `Setup` of the TPEKS scheme on input a security parameter λ, threshold parameter t and number of servers n. The challenger gives $\mathcal{B}_c$ the public key pk. If $\mathcal{B}_c$ submits a pair of keywords w, w', the challenger computes a target ciphertext Φ^*. $\mathcal{B}_c$ issues trapdoor share and test queries on the PEKS ciphertext Φ. The first game (Game 1) differs from the previous one by simulation of trapdoor share queries. If these shares are involved in computing the PEKS ciphertext, the simulator modifies the challenge ciphertext. The rest of Game 0 remains unmodified. The simulation is distributed in two subcases. In Case 1 holds $C \neq C^*, w^* \neq w'$, which invokes $\mathcal{A}_{ind}$ to simulate the queries on identities $id^* \neq id'$. In Case 2 holds $C = C^*$, $w^* = w'$, where $\mathcal{A}_{rob}$ is invoked for the simulation of queries on id^*. In Case 1, $\mathcal{A}_{ind}$ aborts the game if $\mathcal{B}_c$ issues more than $t-1$ queries on w^* or $w^* = w'$. In Case 2, where holds $C = C^*$, $\mathcal{A}_{rob}$

aborts the game if $\mathcal{B}_c$ issues more than $t-1$ queries on $w^* = w'$. The second game differs from Game 1 by simulation of test queries. In Case 1, $\mathcal{A}_{ind}$ aborts the game if $\mathcal{B}_c$ issues queries on w^*. In Case 2, where holds $C = C^*$, $\mathcal{A}_{rob}$ aborts the game if $\mathcal{B}_c$ issues queries on $w^* = w'$. Each of the simulation steps looks as follows:

Setup: $\mathcal{S}$ is given as input mpk of IBTD. It sets TPEKS public key pk equal to mpk. We assume that a set of $t-1$ servers have been corrupted. When $\mathcal{B}_c$ issues trapdoor share and decryption queries on input (w,i) and (w,Φ) respectively, where $w \neq w'$, and $\Phi = (C,R), \Phi^* = (C^*,R^*)$, $\mathcal{S}$ distinguishes between two cases - Case 1: $C \neq C^*, w \neq w'$ where $\mathcal{A}_{ind}$ is invoked and Case 2: $C = C^*$, $w = w'$, where $\mathcal{A}_{rob}$ is invoked.

Queries to $\mathcal{O}$ShareTrpd: Let (w,i) be a trapdoor share query issued by $\mathcal{B}_c$. $\mathcal{S}$ sets $w \leftarrow id$ and queries its oracle $\mathcal{O}\texttt{KeyDer}(w,i)$. The oracle outputs $(i, sk_{w,i})$, which $\mathcal{S}$ sets equal to $(i, T_{w,i})$ and returns it to $\mathcal{B}_c$. In Case 1, the simulator represented by $\mathcal{A}_{ind}$ aborts if $\mathcal{B}_c$ issued more than $t-1$ queries to $\mathcal{O}\texttt{ShareTrpd}(w_0,i)$ and to $\mathcal{O}\texttt{ShareTrpd}(w_1,i)$. In Case 2, simulator represented by $\mathcal{A}_{rob}$ aborts if $w = w'$ or $(w,T,I) \notin$ **List**, or $(w',T',I') \notin$ **List**, or $|I| \geq t, |I'| \geq t$, where $T = \{(1,T_{w',1}),\ldots,(n,T_{w',n})\}, I, I' \subset [1,n]$.

Queries to $\mathcal{O}$Test: $\mathcal{B}_{ind}$ issues test queries on (w,Φ). $\mathcal{S}$ sets $w \leftarrow id, \Phi = (C,R)$ and queries its oracle $\mathcal{O}\texttt{Dec}(w,C)$. The oracle outputs m. $\mathcal{S}$ sets $R \leftarrow m$ and returns R to $\mathcal{B}_c$. In Case 1, $\mathcal{S}$ aborts simulation if $\mathcal{B}_c$ queried $\mathcal{O}\texttt{Test}(w,\Phi^*)$ or $\mathcal{O}\texttt{Test}(w',\Phi^*)$, where $\Phi^* = (C^*,R^*)$. In Case 2, $\mathcal{S}$ aborts if $w = w'$ or $(w',T',I') \notin$ **List**, or $(w,T,I) \notin$ **List**, or $|I| \geq t, |I'| \geq t$, where $T = \{(1,T_{w,1}),\ldots,(n,T_{w,n})\}$ and $I, I' \subset [1,n]$.

Challenge: $\mathcal{B}_c$ outputs a challenge identity w^* and $R_0, R_1 \xleftarrow{r} \{0,1\}^\ell$. $\mathcal{S}$ responds with IBTD ciphertext $\Phi^* = (C^*, R_b)$, where $C^* \leftarrow \texttt{Enc}(pk, w^*, R_b)$, for $b \in \{0,1\}$ and $R^* \xleftarrow{r} \{0,1\}$. $\mathcal{S}$ returns its ciphertext $C^* \leftarrow Enc(mpk, w^*, Rb)$.

Analysis: In Case 1: Simulator $\mathcal{S}$ is represented by $\mathcal{A}_{inc}$. Let q_{ts}, q_t be the number of issued trapdoor share queries on different id's. We assume that $\mathcal{B}_c$ corrupts $t-1$-out-of-n servers with index $i_1,\ldots,i_{t-1}$. The probability that $\mathcal{S}$ corrupts a server j with $j \in \{i_1,\ldots,i_{t-1}\}$ is $1/\binom{n}{t-1}$. Let E denote the event that $\mathcal{B}_c$ wins the indistinguishability experiment from Definition 8. Let $E1$ denote the event that $\mathcal{B}_c$ wins Game 1. It holds $\frac{1}{2}\mathbf{Adv}^{\text{TPEKS-CONS}}_{\mathcal{B}_c}(1^\lambda) = Pr[E] - 1/2 \geq 1/\binom{n}{t-1}\frac{1}{q_t s}(Pr[E1] - 1/2)$. Let $E2$ denote the event that $\mathcal{B}_c$ wins Game 2, then holds:

$$Pr[E1] - 1/2 \geq \frac{1}{q_t}(Pr[E2] - 1/2)$$

$$\Leftrightarrow \frac{1}{2}\mathbf{Adv}^{\text{TPEKS-CONS}}_{\mathcal{B}_c}(1^\lambda) = Pr[E] - 1/2 \geq 1/\binom{n}{t-1}\frac{1}{q_t s}\frac{1}{q_t}(Pr[E2] - 1/2).$$

In Case 2: Simulator $\mathcal{S}$ is represented by $\mathcal{A}_{rob}$. Let q'_{ts}, q'_t be the number of issued trapdoor share queries on different id's. We assume that $\mathcal{B}_c$ corrupts $t-1$-out-of-n servers with index $i_1,\ldots,i_{t-1}$. The probability that $\mathcal{S}$ corrupts a server j with $j \in \{i_1,\ldots,i_{t-1}\}$ is $1/\binom{n}{t-1}$. Let E denote the event that $\mathcal{B}_c$

wins the indistinguishability experiment from Definition 8. Let $\widetilde{E1}$ denote the event that $\mathcal{B}_c$ wins Game 1. It holds $\frac{1}{2}\mathbf{Adv}^{\text{TPEKS-CONS}}_{\mathcal{B}_c}(1^\lambda) = Pr[E] - 1/2 \geq 1/\binom{n}{t-1}\frac{1}{q_{ts}}(Pr[\widetilde{E1}] - 1/2)$. Let $\widetilde{E2}$ denote the event that $\mathcal{B}_c$ wins Game 2, then holds:

$$Pr[E1] - 1/2 \geq \frac{1}{q_t}(Pr[\widetilde{E2}] - 1/2)$$

$$\Leftrightarrow \frac{1}{2}\mathbf{Adv}^{\text{TPEKS-CONS}}_{\mathcal{B}_c}(1^\lambda) = Pr[E] - 1/2 \geq 1/\binom{n}{t-1}\frac{1}{q_{ts}}\frac{1}{q_t}(Pr[\widetilde{E2}] - 1/2)$$

The total advantage of $\mathcal{B}_c$ is given by

$$\begin{aligned}
&\frac{1}{2}\mathbf{Adv}^{\text{TPEKS-CONS}}_{\mathcal{B}_c}(1^\lambda) = Pr[E|Case\,1] + Pr[E|Case\,2] = 2Pr[E] - 1 \\
&\geq 1/\binom{n}{t-1}\frac{1}{q_{ts}}\frac{1}{q_t}(Pr[E2] - 1/2) + 1/\binom{n}{t-1}\frac{1}{q_{ts}}\frac{1}{q_t}(Pr[\widetilde{E2}] - 1/2) \\
&= 1/\binom{n}{t-1}\frac{1}{q_{ts}}\frac{1}{q_t}(Pr[E2] + Pr[\widetilde{E2}] - 1).
\end{aligned}$$

References

1. Abdalla, M., Bellare, M., Catalano, D., Kiltz, E., Kohno, T., Lange, T., Malone-Lee, J., Neven, G., Pailier, P., Shi, H.: Searchable encryption revisited: consistency properties, relation to anonymous IBE and extensions. J. Cryptol. **21**, 350–391 (2008)
2. Abdalla, M., Bellare, M., Neven, G.: Robust encryption. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 480–497. Springer, Heidelberg (2010)
3. Baek, J., Safavi-Naini, R., Susilo, W.: Public key encryption with keyword search revisited. IACR Cryptology ePrint Archive, p. 191 (2005)
4. Baek, J., Safavi-Naini, R., Susilo, W.: Public key encryption with keyword search revisited. In: Gervasi, O., Murgante, B., Laganà, A., Taniar, D., Mun, Y., Gavrilova, M.L. (eds.) ICCSA 2008, Part I. LNCS, vol. 5072, pp. 1249–1259. Springer, Heidelberg (2008)
5. Baek, J., Zheng, Y.: Identity-based threshold decryption. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 262–276. Springer, Heidelberg (2004)
6. Benaloh, J., Chase, M., Horvitz, E., Lauter, K.E.: Patient controlled encryption: ensuring privacy of electronic medical records. In: Proceedings of the First ACM Cloud Computing Security Workshop, CCSW 2009, pp. 103–114 (2009)
7. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (2004)
8. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
9. Boneh, D., Waters, B.: Conjunctive, subset, and range queries on encrypted data. IACR Cryptology ePrint Archive, p. 287 (2006)
10. Byun, J.W., Rhee, H.S., Park, H.-A., Lee, D.-H.: Off-line keyword guessing attacks on recent keyword search schemes over encrypted data. In: Jonker, W., Petković, M. (eds.) SDM 2006. LNCS, vol. 4165, pp. 75–83. Springer, Heidelberg (2006)

11. Cao, N., Wang, C., Li, M., Ren, K., and Lou, W.: Privacy-preserving multi-keyword ranked search over encrypted cloud data. In: 30th IEEE International Conference on Computer Communications INFOCOM 2011, pp. 829–837 (2011)
12. Chang, Y.-C., Mitzenmacher, M.: Privacy preserving keyword searches on remote encrypted data. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 442–455. Springer, Heidelberg (2005)
13. Di Crescenzo, G., Saraswat, V.: Public key encryption with searchable keywords based on jacobi symbols. In: Srinathan, K., Rangan, C.P., Yung, M. (eds.) INDOCRYPT 2007. LNCS, vol. 4859, pp. 282–296. Springer, Heidelberg (2007)
14. Curtmola, R., Garay, J.A., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: improved definitions and efficient constructions. IACR Cryptology ePrint Archive, p. 210 (2006)
15. Fang, L., Susilo, W., Ge, C., Wang, J.: Public key encryption with keyword search secure against keyword guessing attacks without random oracle. Inf. Sci. **238**, 221–241 (2013)
16. Goh, E.: Secure indexes. IACR Cryptology ePrint Archive, p. 216 (2003)
17. Golle, P., Staddon, J., Waters, B.: Secure conjunctive keyword search over encrypted data. In: Jakobsson, M., Yung, M., Zhou, J. (eds.) ACNS 2004. LNCS, vol. 3089, pp. 31–45. Springer, Heidelberg (2004)
18. Hwang, Y.-H., Lee, P.J.: Public key encryption with conjunctive keyword search and its extension to a multi-user system. In: Takagi, T., Okamoto, E., Okamoto, T., Okamoto, T. (eds.) Pairing 2007. LNCS, vol. 4575, pp. 2–22. Springer, Heidelberg (2007)
19. Jeong, I.R., Kwon, J.O., Hong, D., Lee, D.H.: Constructing PEKS schemes secure against keyword guessing attacks is possible? Comput. Commun. **32**(2), 394–396 (2009)
20. Li, M., Lou, W., Ren, K.: Data security and privacy in wireless body area networks. IEEE Wireless Commun. **17**(1), 51–58 (2010)
21. Li, M., Yu, S., Ren, K., Lou, W.: Securing personal health records in cloud computing: patient-centric and fine-grained data access control in multi-owner settings. In: Jajodia, S., Zhou, J. (eds.) SecureComm 2010. LNICST, vol. 50, pp. 89–106. Springer, Heidelberg (2010)
22. Liu, Q., Wang, G., Wu, J.: Secure and privacy preserving keyword searching for cloud storage services. J. Netw. Comput. Appl. **35**(3), 927–933 (2012)
23. Park, D.J., Kim, K., Lee, P.J.: Public key encryption with conjunctive field keyword search. In: Lim, C.H., Yung, M. (eds.) WISA 2004. LNCS, vol. 3325, pp. 73–86. Springer, Heidelberg (2005)
24. Rhee, H.S., Susilo, W., Kim, H.: Secure searchable public key encryption scheme against keyword guessing attacks. IEICE Electron Express **6**(5), 237–243 (2009)
25. Sun, W., Wang, B., Cao, N., Li, M., Lou, W., Hou, Y. T., Li, H.: Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking. In: 8th ACM Symposium on Information, Computer and Communications Security, ASIA CCS, pp. 71–82. ACM (2013)
26. Swaminathan, A., Mao, Y., Su, G., Gou, H., Varna, A. L., He, S., Wu, M., Oard, D. W.: Confidentiality-preserving rank-ordered search. In: Proceedings of the ACM Workshop on Storage Security and Survivability, StorageSS, pp. 7–12 (2007)
27. van Liesdonk, P., Sedghi, S., Doumen, J., Hartel, P., Jonker, W.: Computationally efficient searchable symmetric encryption. In: Jonker, W., Petković, M. (eds.) SDM 2010. LNCS, vol. 6358, pp. 87–100. Springer, Heidelberg (2010)

28. Wang, C., Cao, N., Li, J., Ren, K., Lou, W.: Secure ranked keyword search over encrypted cloud data. In: International Conference on Distributed Computing Systems, ICDCS 2010, pp. 253–262. IEEE Computer Society (2010)
29. Wang, C., Cao, N., Ren, K., Lou, W.: Enabling secure and efficient ranked keyword search over outsourced cloud data. IEEE Trans. Parallel Distrib. Syst. **23**(8), 1467–1479 (2012)
30. Zhang, W. Lin, Y., Xiao, S., Liu, Q., Zhou, T.: Secure distributed keyword search in multiple clouds. In: IEEE 22nd International Symposium of Quality of Service, IWQoS 2014, pp. 370–379. IEEE (2014)