An extended abstract of this paper appears in Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, Volume 3621 of Lecture Notes in Computer Science, pages 205–222, Santa Barbara, California, August 14 – 18, 2005. Springer-Verlag, Berlin, Germany. This is the full version.

Searchable Encryption Revisited: Consistency Properties, Relation to Anonymous IBE, and Extensions

MICHEL ABDALLA	École normale supérieure `Michel.Abdalla@ens.fr`, `http://www.di.ens.fr/users/mabdalla`
MIHIR BELLARE	University of California San Diego `mihir@cs.ucsd.edu`, `http://www.cs.ucsd.edu/users/mihir`
DARIO CATALANO	École normale supérieure `Dario.Catalano@ens.fr`, `http://www.di.ens.fr/users/catalano`
EIKE KILTZ	University of California San Diego `ekiltz@cs.ucsd.edu`, `http://kiltz.net`
TADAYOSHI KOHNO	University of California San Diego `tkohno@cs.ucsd.edu`, `http://www.cs.ucsd.edu/users/tkohno`
TANJA LANGE	Technical University of Denmark `t.lange@mat.dtu.dk`, `http://www.ruhr-uni-bochum.de/itsc/tanja`
JOHN MALONE-LEE	University of Bristol `malone@compsci.bristol.ac.uk`, `http://www.cs.bris.ac.uk/~malone`
GREGORY NEVEN	Katholieke Universiteit Leuven `Gregory.Neven@esat.kuleuven.be`, `http://www.neven.org`
PASCAL PAILLIER	Gemplus Card International `Pascal.Paillier@gemplus.com`
HAIXIA SHI	University of California San Diego `hashi@cs.ucsd.edu`, `http://www.cs.ucsd.edu/users/hashi`

July 2005

Abstract

We identify and fill some gaps with regard to consistency (the extent to which false positives are produced) for public-key encryption with keyword search (PEKS). We define computational and statistical relaxations of the existing notion of perfect consistency, show that the scheme of [7] is computationally consistent, and provide a new scheme that is statistically consistent. We also provide a transform of an anonymous IBE scheme to a secure PEKS scheme that, unlike the previous one, guarantees consistency. Finally we suggest three extensions of the basic notions considered here, namely anonymous HIBE, public-key encryption with temporary keyword search, and identity-based encryption with keyword search.

Contents

1 Introduction

There has recently been interest in various forms of "searchable encryption" [20, 7, 13, 15, 22]. In this paper, we further explore one of the variants of this goal, namely public-key encryption with keyword search (PEKS) as introduced by Boneh, Di Crescenzo, Ostrovsky and Persiano [7]. We begin by discussing consistency-related issues and results, then consider the connection to anonymous identity-based encryption (IBE) and finally discuss some extensions.

1.1 Consistency in PEKS

Any cryptographic primitive must meet two conditions. One is of course a security condition. The other, which we will here call a *consistency* condition, ensures that the primitive fulfills its function. For example, for public-key encryption, the security condition is privacy. (This could be formalized in many ways, eg. IND-CPA or IND-CCA.) The consistency condition is that decryption reverses encryption, meaning that if M is encrypted under public key pk to result in ciphertext C, then decrypting C under the secret key corresponding to pk results in M being returned.

PEKS. In a PEKS scheme, Alice can provide a *gateway* with a trapdoor t_w (computed as a function of her secret key) for any keyword w of her choice. A sender encrypts a keyword w' under Alice's public key pk to obtain a ciphertext C that is sent to the gateway. The latter can apply a test function Test to t_w, C to get back 0 or 1. The consistency condition as per [7] is that if $w = w'$ then $\mathsf{Test}(t_w, C)$ returns 1 and if $w \neq w'$ it returns 0. The security condition is that the gateway learn nothing about w' beyond whether or not it equals w. (The corresponding formal notion will be denoted PEKS-IND-CPA.) The application setting is that C can be attached to an email (ordinarily encrypted for Alice under a different public key), allowing the gateway to route the email to different locations (eg. Alice's desktop, laptop or pager) based on w while preserving privacy of the latter to the largest extent possible.

Consistency of $\mathcal{BDOP}$-$\mathcal{PEKS}$. It is easy to see (cf. Proposition 3.1) that the main construction of [7] (a random oracle model, BDH-based PEKS-IND-CPA secure PEKS scheme that we call $\mathcal{BDOP}$-$\mathcal{PEKS}$) fails to meet the consistency condition defined in [7] and stated above. (Specifically, there are distinct keywords w, w' such that $\mathsf{Test}(t_w, C) = 1$ for any C that encrypts w'.) The potential problem this raises in practice is that email will be incorrectly routed.

New notions of consistency. It is natural to ask if $\mathcal{BDOP}$-$\mathcal{PEKS}$ meets some consistency condition that is weaker than theirs but still adequate in practice. To answer this, we provide some new definitions. Somewhat unusually for a consistency condition, we formulate consistency more like a security condition, via an experiment involving an adversary. The difference is that this adversary is not very "adversarial": it is supposed to reflect some kind of worst case but not malicious behavior. However this turns out to be a difficult line to draw, definitionally, so that some subtle issues arise. One outcome of this approach is that it naturally gives rise to a hierarchy of notions of consistency, namely perfect, statistical and computational. The first asks that the advantage of any (even computationally unbounded) adversary be zero; the second that the advantage of any (even computationally unbounded) adversary be negligible; the third that the advantage of any polynomial-time adversary be negligible. We note that perfect consistency as per our definition coincides with consistency as per [7], and so our notions can be viewed as natural weakenings of theirs.

An analogy. There is a natural notion of *decryption error* for encryption schemes [14, Section 5.1.2]. A perfectly consistent PEKS is the analog of an encryption scheme with zero decryption error (the usual requirement). A statistically consistent PEKS is the analog of an encryption scheme with negligible decryption error (a less common but still often used condition [1, 11]). However, computational consistency is a non-standard relaxation, for consistency conditions are typically not computational. This is not because one cannot define them that way (one could certainly define a computational consistency requirement for encryption) but rather because there has never been any motivation to do so. What makes PEKS different, as emerges from the results below, is that computational consistency is relevant and arises naturally.

Consistency of $\mathcal{BDOP}$-$\mathcal{PEKS}$, revisited. The counter-example (cf. Proposition 3.1) showing that $\mathcal{BDOP}$-$\mathcal{PEKS}$ is not perfectly consistent extends to show that it is not statistically consistent either. However, we show (cf. Theorem 3.3) that $\mathcal{BDOP}$-$\mathcal{PEKS}$ is computationally consistent. In the random-oracle model, this is not under any computational assumption: the limitation on the running time of the adversary is relevant because it limits the number of queries the adversary can make to the random oracle. When the random oracle is instantiated via a hash function, we would need to assume collision-resistance of the hash function. The implication of this result is that $\mathcal{BDOP}$-$\mathcal{PEKS}$ is probably fine to use in practice, in that incorrect routing of email, while possible in principle, is unlikely to actually happen.

A statistically consistent PEKS scheme. We provide the first construction of a PEKS scheme that is *statistically* consistent. The scheme is in the RO model, and is also PEKS-IND-CPA secure assuming the BDH problem is hard.

The motivation here was largely theoretical. From a foundational perspective, we wanted to know whether PEKS was an anomaly in the sense that only computational consistency is possible, or whether, like other primitives, statistical consistency could be achieved. However, it is also true that while computational consistency is arguably enough in an application, statistical might be preferable because the guarantee is unconditional.

1.2 PEKS and anonymous IBE

$\mathcal{BDOP}$-$\mathcal{PEKS}$ is based on the Boneh-Franklin IBE ($\mathcal{BF}$-$\mathcal{IBE}$) scheme [8]. It is natural to ask whether one might, more generally, build PEKS schemes from IBE schemes in some blackbox way. To this end, a transform of an IBE scheme into a PEKS scheme is presented in [7]. Interestingly, they note that the property of the IBE scheme that appears necessary to provide PEKS-IND-CPA of the PEKS scheme is not the usual IBE-IND-CPA but rather anonymity. (An IBE scheme is anonymous if a ciphertext does not reveal the identity of the recipient [2].) While [7] stops short of stating and proving a formal result here, it is not hard to verify that their intuition is correct. Namely one can show that if the starting IBE scheme $\mathcal{IBE}$ meets an appropriate formal notion of anonymity (IBE-ANO-CPA, cf. Section 4.1) then $\mathcal{PEKS}$ = bdop-ibe-2-peks($\mathcal{IBE}$) is PEKS-IND-CPA, where bdop-ibe-2-peks denotes the transform of [7].

Consistency in bdop-ibe-2-peks. Unfortunately, we show (cf. Theorem 4.1) that there are IBE schemes for which the PEKS scheme output by bdop-ibe-2-peks is not even computationally consistent. This means that bdop-ibe-2-peks is not in general a suitable way to turn an IBE scheme into a PEKS scheme. (Although it might be in some cases, and in particular is when the starting IBE scheme is $\mathcal{BF}$-$\mathcal{IBE}$, for in that case the resulting PEKS scheme is $\mathcal{BDOP}$-$\mathcal{PEKS}$.)

new-ibe-2-peks. We propose a randomized variant of the bdop-ibe-2-peks transform that we call new-ibe-2-peks, and prove that if an IBE scheme $\mathcal{IBE}$ is IBE-ANO-CPA and IBE-IND-CPA then the PEKS scheme new-ibe-2-peks($\mathcal{IBE}$) is PEKS-IND-CPA and computationally consistent (cf. Section 4.3). We do not know of a transform where the resulting PEKS scheme is statistically or perfectly consistent.

Anonymous IBE schemes. The above motivates finding anonymous IBE schemes. Towards this, we begin by extending Halevi's condition for anonymity [16] to the IBE setting (cf. Section 4.4). Based on this, we are able to give a simple proof that the (random-oracle model) $\mathcal{BF}$-$\mathcal{IBE}$ scheme [8] is IBE-ANO-CPA assuming the BDH problem is hard (cf. Theorem 4.4). (We clarify that a proof of this result is implicit in the proof of security of the $\mathcal{BF}$-$\mathcal{IBE}$ based $\mathcal{BDOP}$-$\mathcal{PEKS}$ scheme given in [7]. Our contribution is to have stated the formal definition of anonymity and provided a simpler proof via the extension of Halevi's condition.) Towards answering the question of whether there exist anonymous IBE schemes in the standard (as opposed to random oracle) model, we present in Appendix B.1 an attack to show that Water's IBE scheme [21] is not IBE-ANO-CPA.

1.3 Extensions

Anonymous HIBE. We provide definitions of anonymity for hierarchical IBE (HIBE) schemes. Our definition can be parameterized by a level, so that we can talk of a HIBE that is anonymous at level l. We note that the HIBE schemes of [12, 6] are not anonymous, even at level 1. (That of [17] appears to be anonymous at both levels 1 and 2 but is very limited in nature and thus turns out not to be useful for our applications.) We modify the construction of Gentry and Silverberg [12] to obtain a HIBE that is (HIBE-IND-CPA and) anonymous at level 1. The construction is in the random oracle model and assumes BDH is hard.

PETKS. In a PEKS scheme, once the gateway has the trapdoor for a certain period, it can test whether this keyword was present in any past ciphertexts or future ciphertexts. It may be useful to limit the period in which the trapdoor can be used. Here we propose an extension of PEKS that we call public-key encryption with temporary keyword search (PETKS) that allows this. A trapdoor here is created for a time interval $[s..e]$ and will only allow the gateway to test whether ciphertexts created in this time interval contain the keyword. We provide definitions of privacy and consistency for PETKS, and then show how to implement it with overhead that is only logarithmic in the total number of time periods. Our construction can use any HIBE that is anonymous at level 1. Using the above-mentioned HIBE we get a particular instantiation that is secure in the random-oracle model if BDH is hard.

IBEKS. We define the notion of an identity-based encryption with keyword search scheme. This is just like a PEKS scheme except that encryption is performed given only the identity of the receiver and a master public-key, just like in an IBE scheme. We show how to implement IBEKS given any level-2 anonymous HIBE scheme. However no suitable implementation of the latter is known, so we have no concrete implementation of IBEKS.

1.4 Remarks

bdop-peks-2-ibe. Boneh et. al. [7] showed how to transform a PEKS-IND-CPA PEKS scheme into an IBE-IND-CPA IBE scheme. We remark that their transform requires the starting PEKS scheme to be perfectly consistent. (But no perfectly consistent schemes are known to date.) If it is only statistically or computationally consistent, the resulting IBE scheme will only meet a corresponding statistical or computational relaxation of the consistency condition for IBE schemes. (And thus not be an IBE scheme as per the standard definition of the latter [8].)

Limited PEKS schemes. Boneh et. al. [7] also present a couple of PEKS schemes that avoid the RO model but are what they call *limited*. Both use a standard public-key encryption scheme as a building block. In the first scheme, the public key has size polynomial in the number of keywords that can be used. In the second scheme, the key and ciphertext have size polynomial in the number of trapdoors that can be securely issued to the gateway. Although these schemes are not very interesting due to their limited nature, one could ask about their consistency. In Appendix A we extend our definitions of consistency to this limited setting. Interestingly, we show that based on only a computational assumption about the underlying standard public-key encryption scheme (namely, that it is IND-CPA, or even just one-way), the first scheme is *statistically* consistent. We also show that the second scheme is computationally consistent under the same assumption on the standard public-key encryption scheme, and present a variant that is statistically consistent.

Consistency of other searchable encryption schemes. Of the other papers on searchable encryption of which we are aware [20, 13, 15, 22], none formally define or rigorously address the notion of consistency for their respective types of searchable encryption schemes. Goh [13] and Golle, Staddon, and Waters [15] define consistency conditions analogous to BDOP's "perfect consistency" condition, but none of the constructions in [13, 15] satisfy their respective perfect consistency condition. Song, Wagner, and Perrig [20] and Waters et al. [22] do not formally state and prove consistency conditions for their respective searchable encryption schemes, but they, as well as Goh [13], do acknowledge and informally bound the non-zero probability of a false positive.

2 Some definitions

Notation and conventions. If x is a string then $|x|$ denotes its length, and if S is a set then $|S|$ denotes its size. The empty string is denoted ε. Constructs in the RO (random oracle) model [4] might use multiple random oracles, but since one can always obtain these from a single one [4], formal definitions will assume just one RO. Unless otherwise indicated, an algorithm may be randomized. "PT" stands for polynomial time and "PTA" for polynomial-time algorithm or adversary. We denote by $\mathbb{N}$ the set of positive integers, and by $k \in \mathbb{N}$ the security parameter. A *message space* MsgSp is a map, assigning to every $k \in \mathbb{N}$ a set of strings, such that $\{0,1\}^k \subseteq \mathsf{MsgSp}(k) \subseteq \{0,1\}^*$ for all $k \in \mathbb{N}$ and the following conditions hold: first, there is a PTA that on input $1^k, M$ returns 1 if $M \in \mathsf{MsgSp}(k)$ and 0 otherwise; second, $\{0,1\}^{|M|} \subseteq \mathsf{MsgSp}(k)$ for all $k \in \mathbb{N}$ and $M \in \mathsf{MsgSp}(k)$.

PEKS. A *public key encryption with keyword search* (PEKS) scheme [7] $\mathcal{PEKS} = (\mathsf{KG}, \mathsf{PEKS}, \mathsf{Td}, \mathsf{Test})$ consists of PTAs. Via $(pk, sk) \xleftarrow{\$} \mathsf{KG}(1^k)$, where $k \in \mathbb{N}$ is the security parameter, the randomized key-generation algorithm produces keys for the receiver; via $C \xleftarrow{\$} \mathsf{PEKS}^H(pk, w)$ a sender encrypts a keyword w to get a ciphertext; via $t_w \xleftarrow{\$} \mathsf{Td}^H(sk, w)$ the receiver computes a trapdoor t_w for keyword w and provides it to the gateway; via $b \leftarrow \mathsf{Test}^H(t_w, C)$ the gateway tests whether C encrypts w, where b is a bit with 1 meaning "accept" or "yes" and 0 meaning "reject" or "no". Here H is a random oracle whose domain and/or range might depend on k and pk.

Consistency. The requirement of [7] can be divided into two parts. The first, which we call *right keyword consistency*, is that $\mathsf{Test}(t_w, C)$ always accepts when C encrypts w. More formally, for all $k \in \mathbb{N}$ and all $w \in \{0,1\}^*$,

$$\Pr\left[\,\mathsf{Test}^H(\mathsf{Td}^H(sk, w), \mathsf{PEKS}^H(pk, w)) = 1\,\right] \;=\; 1\;,$$

where the probability is taken over the choice of $(pk, sk) \xleftarrow{\$} \mathsf{KG}(1^k)$, the random choice of H, and the coins of all the algorithms in the expression above. Since we will always require this too, it is convenient henceforth to take it as an integral part of the PEKS notion and not mention it again, reserving the term "consistency" to only refer to what happens when the ciphertext encrypts a keyword different from the one for which the gateway is testing. In this regard, the requirement of [7], which we will call *perfect consistency*, is that $\mathsf{Test}(t_{w'}, C)$ always reject when C doesn't encrypt w'. More formally, for all $k \in \mathbb{N}$ and all distinct $w, w' \in \{0,1\}^*$,

$$\Pr\left[\, \mathsf{Test}^H(\mathsf{Td}^H(sk, w'), \mathsf{PEKS}^H(pk, w)) = 1 \,\right] \;=\; 0\ ,$$

where the probability is taken over the choice of $(pk, sk) \xleftarrow{\$} \mathsf{KG}(1^k)$, the random choice of H, and the coins of all the algorithms in the expression above. (We note that [7] provide informal rather than formal statements, but it is hard to interpret them in any way other than what we have done.)

Privacy. Privacy for a PEKS scheme [7] asks that an adversary should not be able to distinguish between the encryption of two challenge keywords of its choice, even if it is allowed to obtain trapdoors for any non-challenge keywords. Formally, we associate to an adversary $\mathcal{A}$ and a bit $b \in \{0,1\}$ the following experiment:

```
Experiment $\mathbf{Exp}^{\text{peks-ind-cpa-b}}_{\mathcal{PEKS},\mathcal{A}}(k)$
    $WSet \leftarrow \emptyset$ ; $(pk, sk) \xleftarrow{\$} \mathsf{KG}(1^k)$
    pick random oracle $H$
    $(w_0, w_1, state) \xleftarrow{\$} \mathcal{A}^{\textsc{Trapd}(\cdot),H}(\mathtt{find}, pk)$
    $C \xleftarrow{\$} \mathsf{PEKS}^H(pk, w_b)$
    $b' \xleftarrow{\$} \mathcal{A}^{\textsc{Trapd}(\cdot),H}(\mathtt{guess}, C, state)$
    if $\{w_0, w_1\} \cap WSet = \emptyset$ then return $b'$ else return 0

Oracle $\textsc{Trapd}(w)$
    $WSet \leftarrow WSet \cup \{w\}$
    $t_w \xleftarrow{\$} \mathsf{Td}^H(sk, w)$
    return $t_w$
```

The PEKS-IND-CPA-*advantage* of $\mathcal{A}$ is defined as

$$\mathbf{Adv}^{\text{peks-ind-cpa}}_{\mathcal{PEKS},\mathcal{A}}(k) = \Pr\left[\mathbf{Exp}^{\text{peks-ind-cpa-1}}_{\mathcal{PEKS},\mathcal{A}}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{peks-ind-cpa-0}}_{\mathcal{PEKS},\mathcal{A}}(k) = 1\right] .$$

A scheme $\mathcal{PEKS}$ is said to be PEKS-IND-CPA-*secure* if the above advantage is a negligible function in k for all PTAs $\mathcal{A}$.

Parameter generation algorithms and the BDH problem. All pairing based schemes will be parameterized by a *pairing parameter generator*. This is a PTA $\mathcal{G}$ that on input 1^k returns the description of an additive cyclic group $\mathbb{G}_1$ of prime order p, where $2^k < p < 2^{k+1}$, the description of a multiplicative cyclic group $\mathbb{G}_2$ of the same order, and a non-degenerate bilinear pairing e: $\mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. See [8] for a description of the properties of such pairings. We use $\mathbb{G}_1^*$ to denote $\mathbb{G}_1 \setminus \{0\}$, i.e. the set of all group elements except the neutral element. We define the advantage of an adversary $\mathcal{A}$ in solving the BDH problem relative to a pairing parameter generator $\mathcal{G}$ as

$$\mathbf{Adv}^{\text{bdh}}_{\mathcal{G},\mathcal{A}}(k) \;=\; \Pr\left[\mathcal{A}(1^k, (\mathbb{G}_1, \mathbb{G}_2, p, e), P, aP, bP, cP) = e(P,P)^{abc} \;:\; \begin{array}{c}(\mathbb{G}_1, \mathbb{G}_2, p, e) \xleftarrow{\$} \mathcal{G}(1^k)\ ; \\ P \xleftarrow{\$} \mathbb{G}_1^*\ ;\ a, b, c \xleftarrow{\$} \mathbb{Z}_p^*\end{array}\right] .$$

We say that the BDH problem is hard relative to this generator if $\mathbf{Adv}^{\text{bdh}}_{\mathcal{G},\mathcal{A}}$ is a negligible function in k for all PTAs $\mathcal{A}$.

3 Consistency in PEKS

We show that the $\mathcal{BDOP}$-$\mathcal{PEKS}$ scheme is not perfecty consistent, introduce new notions of statistical and computational consistency, and show that although $\mathcal{BDOP}$-$\mathcal{PEKS}$ continues to fail the former it does meet the latter. We then provide a new PEKS scheme that is statistically consistent.

3.1 Perfect consistency of $\mathcal{BDOP}$-$\mathcal{PEKS}$

Figure 1 presents the $\mathcal{BDOP}$-$\mathcal{PEKS}$ scheme. It is based on a pairing parameter generator $\mathcal{G}$.

```
KG$(1^k)$
  $(\mathbb{G}_1, \mathbb{G}_2, p, e) \xleftarrow{\$} \mathcal{G}(1^k)$ ; $P \xleftarrow{\$} \mathbb{G}_1^*$ ; $s \xleftarrow{\$} \mathbb{Z}_p^*$
  $pk \leftarrow (\mathbb{G}_1, \mathbb{G}_2, p, e, P, sP)$ ; $sk \leftarrow (pk, s)$
  return $(pk, sk)$

PEKS$^{H_1,H_2}(pk, w)$
  parse $pk$ as $(\mathbb{G}_1, \mathbb{G}_2, p, e, P, sP)$
  $r \xleftarrow{\$} \mathbb{Z}_p^*$ ; $T \leftarrow e(H_1(w), sP)^r$
  $C \leftarrow (rP, H_2(T))$ ; return $C$

Td$^{H_1}(sk, w)$
  parse $sk$ as $(pk = (\mathbb{G}_1, \mathbb{G}_2, p, e, P, sP), s)$
  $t_w \leftarrow (pk, sH_1(w))$ ; return $t_w$

Test$^{H_1,H_2}(t_w, C)$
  parse $t_w$ as $((\mathbb{G}_1, \mathbb{G}_2, p, e, P, sP), X)$
  parse $C$ as $(U, V)$ ; $T \leftarrow e(X, U)$
  if $V = H_2(T)$ then return 1
  else return 0
```

Figure 1: Algorithms constituting the $\mathcal{BDOP}$-$\mathcal{PEKS}$ scheme. $\mathcal{G}$ is a pairing parameter generator and H_1: $\{0,1\}^* \to \mathbb{G}_1$ and H_2: $\mathbb{G}_2 \to \{0,1\}^k$ are random oracles.

Proposition 3.1 *The $\mathcal{BDOP}$-$\mathcal{PEKS}$ scheme is not perfectly consistent.* ▮

Proof: Since the number of possible keywords is infinite, there will certainly exist distinct keywords $w, w' \in \{0,1\}^*$ such that $H_1(w) = H_1(w')$. The trapdoors for such keywords will be the same, and so $\mathsf{Test}^{H_1,H_2}(\mathsf{Td}(sk, w), \mathsf{PEKS}^{H_1,H_2}(pk, w'))$ will always return 1. ▮

It is tempting to say that, since H_1 is a random oracle, the probability of a collision is small, and thus the above really does not matter. Whether or not this is true depends on how one wants to define consistency, which is the issue we explore next.

3.2 New notions of consistency

We consider various possible relaxations of perfect consistency and argue that the obvious ones are inadequate either because $\mathcal{BDOP}$-$\mathcal{PEKS}$ continues to fail them or because they are too weak. We then motivate and present our approach and definitions.

A natural relaxation of perfect consistency. A natural way to obtain a relaxed definition of perfect consistency is by analogy with the definition of encryption with negligible decryption error [14, Section 5.1.2]. This results in asking that there exist a negligible function $\nu(\cdot)$ such that for all k, all pairs (pk, sk) that have positive probability of being output by $\mathsf{KG}(1^k)$, all choices for the random oracle H, and all distinct keywords w, w',

$$\Pr\left[\,\mathsf{Test}^H(\mathsf{Td}^H(sk, w'), \mathsf{PEKS}^H(pk, w)) = 1\,\right] \;\le\; \nu(k)\ .$$

Here the probability is only over the coins for $\mathsf{PEKS}, \mathsf{Td}, \mathsf{Test}$. This definition is, in our view, a perfectly adequate relaxation, and might be useful in some settings. The difficulty here is that it is still too strong in the sense that the $\mathcal{BDOP}$-$\mathcal{PEKS}$ scheme continues to not meet it (our attack of Proposition 3.1 shows this too) and we are seeking something such a scheme could meet.

Another possible relaxation. The next obvious thought is to take the probability over the choices of keys and random oracle as well. This leads to asking that there is a negligible function $\nu(\cdot)$ such that

$$\forall\, w \neq w' \;\;:\;\; \Pr\left[\,\mathsf{Test}^H(pk, \mathsf{Td}^H(sk, w'), \mathsf{PEKS}^H(pk, w)) = 1\,\right] \;\le\; \nu(k)\ , \qquad (1)$$

where the probability is taken over the choice of $(pk, sk) \xleftarrow{\$} \mathsf{KG}(1^k)$, the random choice of H, and the coins of all the algorithms in the expression above. Now, since we are fixing w, w' before taking the probability, and the latter includes the choice of H_1 in the $\mathcal{BDOP}$-$\mathcal{PEKS}$ scheme, the probability that $H_1(w) = H_1(w')$ is at most 2^{-k}. Our "attack" of Proposition 3.1 no longer applies. And in fact (using the techniques of our proof of Theorem 3.3) one can show that the BDOP scheme *does* meet the above condition. However, Equation (1) is (in our view) an incorrect definition of consistency because it does not allow w, w' to depend on public quantities related to the receiver, such as its public key, the hash functions being used, or queries to them if they are random oracles. Our claim is that, as a result, the condition is too weak to guarantee that email is correctly routed by the gateway. Let us now try to justify this view.

Considering adversaries. It is helpful to move to a different perspective. Namely, imagine there is an adversary that wants to make consistency fail. What resources are available to it? It could certainly play the role of sender, and choose w any way it wants. In particular, it could, in real life, choose w as a function of the public key of the receiver, the hash functions being used, or queries to them if they are random oracles. It could also, potentially, influence the receiver into choosing w' in some way helpful to the adversary, in particular as a function of all the same public quantities. But the fact that a PEKS scheme meets the condition of Equation (1) does *not* rule out the possibility that such an adversary exists and succeeds, for Equation (1) does not allow w, w' to depend on public quantities related to the receiver.

Does this matter? A consistency condition is not supposed to be about what an adversary might do, but rather what might happen in real life. However, it is actually rather hard to draw a line between the two. An adversary in this case is after all a model of worst-case real-life behavior. There would appear to be real situations where keywords do depend on public quantities. This might happen because the subject or topic of the email is related to the usage of the scheme and some event about it. So we should not rule this out.

One could try to find some meaningful dividing line between malicious behavior and worst-case real life behavior. Here is one. Namely, I might claim that when I encrypt data, I don't actually see your public key: only my encryption software sees it. Similarly, I don't directly query the random oracles; my software does. So how can the keywords I encrypt depend on these quantities unless I actually try to be malicious? But this type of argument is fragile. Today, many of us receive each other's public keys in email (PGP). I may want to send you email containing a discussion or question about the content of the public key you just sent me. Similar (though less obvious) arguments can be made about the hash functions and random oracles. In the end we are led to the conclusion that it is hard to draw a line between malicious behavior and worst-case real-life behavior, and thus it would be safer to just consider an adversary for a consistency condition as well.

Our definitions. Let $\mathcal{PEKS} = (\mathsf{KG}, \mathsf{PEKS}, \mathsf{Td}, \mathsf{Test})$ be a PEKS scheme. We associate to an adversary $\mathcal{U}$ the following experiment:

```
Experiment $\mathbf{Exp}^{\text{peks-consist}}_{\mathcal{PEKS},\mathcal{U}}(k)$
    $(pk, sk) \xleftarrow{\$} \mathsf{KG}(1^k)$ ; pick random oracle $H$
    $(w, w') \xleftarrow{\$} \mathcal{U}^H(pk)$ ; $C \xleftarrow{\$} \mathsf{PEKS}^H(pk, w)$ ; $t_{w'} \xleftarrow{\$} \mathsf{Td}^H(sk, w')$
    if $w \neq w'$ and $\mathsf{Test}^H(t_{w'}, C) = 1$ then return 1 else return 0
```

We define the advantage of $\mathcal{U}$ as

$$\mathbf{Adv}^{\text{peks-consist}}_{\mathcal{PEKS},\mathcal{U}}(k) \;=\; \Pr\left[\mathbf{Exp}^{\text{peks-consist}}_{\mathcal{PEKS},\mathcal{U}}(k) = 1\right] .$$

The scheme is said to be *perfectly consistent* if this advantage is 0 for all (computationally unrestricted) adversaries $\mathcal{U}$, *statistically consistent* if it is negligible for all (computationally unrestricted) adversaries $\mathcal{U}$, and *computationally consistent* if it is negligible for all PTAs $\mathcal{U}$. We remark that we have purposely re-used the term perfect consistency, for in fact the above notion of perfect consistency coincides with the one from [7] recalled above.

Stronger notions? In giving the adversary $\mathcal{U}$ the public key and access to the random oracle, our definition is already quite liberal. One could however, consider an even more liberal (i.e. stronger) definition in which the adversary gets a trapdoor oracle and/or a test oracle under trapdoors for keywords of its choice. Consideration of whether this would be appropriate returns us again to the issues discussed above. We must ask whether in "real-life" there could be an occasion in which the keywords chosen by a sender could depend on information provided by these oracles. The answer is not cut-and-dry. Although quite far-fetched, one might actually imagine such situations. We view the situation as being analogous to the one for security conditions where we often have several different definitions of different strengths to capture increasingly greater adversary resources. For example, for encryption we have the basic IND-CPA and then its extension to IND-CCA. Our sense is that the definition we give above is kind of the "IND-CPA" of consistency conditions, and that the stronger one we have alluded to would be an "IND-CCA" version. For the moment we are content to stick to the "IND-CPA" version, for, even if weaker than the other one, it is already very strong, and also because even in the security realm we are (following [7]) not considering IND-CCA in this paper.

3.3 Statistical and computational consistency of $\mathcal{BDOP}$-$\mathcal{PEKS}$

Having formally defined the statistical and computational consistency requirements for PEKS schemes, we return to evaluating the consistency of $\mathcal{BDOP}$-$\mathcal{PEKS}$. We first observe that Proposition 3.1 extends to show:

Proposition 3.2 *The $\mathcal{BDOP}$-$\mathcal{PEKS}$ scheme is not statistically consistent.* ▌

Proof: Recall that in the proof of Proposition 3.1 we show that there there exist two distinct keywords $w, w' \in \{0,1\}^*$ such that $H_1(w) = H_1(w')$, and that, for these two keywords, $\mathsf{Test}(\mathsf{Td}(sk, w'), \mathsf{PEKS}(pk, w))$ will always return 1. A computationally unbounded adversary can find two such keywords by exhaustive search. ▌

On the positive side, the following means that $\mathcal{BDOP}$-$\mathcal{PEKS}$ is probably fine in practice:

Theorem 3.3 *The $\mathcal{BDOP}$-$\mathcal{PEKS}$ scheme is computationally consistent.* ▌

Proof: Let $\mathcal{U}$ be a PTA. Let (w, w') denote the pair of keywords that $\mathcal{U}$ returns in the consistency experiment, and assume without loss of generality that $w \neq w'$. Let $r \in \mathbb{Z}_p^*$ denote the value chosen at random by $\mathsf{PEKS}^{H_1,H_2}(pk, w)$. Let $T = e(H_1(w), sP)^r$ and let $T' = e(H_1(w'), sP)^r$. Note that $\mathcal{U}$ wins exactly when $w \neq w'$ and $H_2(T) = H_2(T')$. Let $w_1, \ldots, w_{q_1}$ be the queries of $\mathcal{U}$ to H_1 and let $WSet = \{w_1, \ldots, w_{q_1(k)}\} \cup \{w, w'\}$. Let $T_1, \ldots, T_{q_2(k)}$ be the queries of $\mathcal{U}$ to H_2 and let $TSet = \{T_1, \ldots, T_{q_2(k)}\} \cup \{T, T'\}$. Let E_1 be the event that there exist distinct $v, v' \in WSet$ such that $H_1(v) = H_1(v')$, and let E_2 be the event that there exist distinct $x, x' \in TSet$ such that $H_2(x) = H_2(x')$. If $\Pr[\cdot]$ denotes the probability in the consistency experiment, then

$$\mathbf{Adv}^{\text{peks-consist}}_{\mathcal{PEKS},\mathcal{U}}(k) \;\leq\; \Pr[E_1] + \Pr[E_2] + \Pr\left[\mathbf{Exp}^{\text{peks-consist}}_{\mathcal{BDOP}\text{-}\mathcal{PEKS},\mathcal{U}}(k) = 1 \wedge \overline{E_1} \wedge \overline{E_2}\right]. \quad (2)$$

Our definition of $\mathcal{G}$ required that $|\mathbb{G}_1| > 2^k$, and hence the first and second terms are respectively upper bounded via $(q_1+2)^2/|\mathbb{G}_1| < (q_1+2)^2/2^k$ and $(q_2+2)^2/2^k$. Now se claim that if $H_1(w) \neq H_1(w')$, then $X \neq X'$. Under this claim, the last term of Equation (2) is 0, since if $\overline{E_1}$ occurs, then $H_1(w) \neq H_1(w')$ and $T \neq T'$, and if $\overline{E_2}$ also occurs, then $H_2(T) \neq H_2(T')$. To justify our claim above, note that if $H_1(w) \neq H_1(w')$, then $H_1(w) = \alpha P$ and $H_1(w') = \alpha' P$ for some distinct $\alpha, \alpha' \in \mathbb{Z}_p$. Setting $g = e(P,P)^{rs}$, we can rewrite T, T' as $T = g^\alpha$ and $T' = g^{\alpha'}$. Since $e(P,P)$ is a generator of $\mathbb{G}_2$, since $\mathbb{G}_2$ is of prime order p, and since p does not divide rs, g must also be a generator of $\mathbb{G}_2$. Thus $T = T'$. ▌

```
$\mathsf{KG}(1^k)$
    $(\mathbb{G}_1, \mathbb{G}_2, p, e) \xleftarrow{\$} \mathcal{G}(1^k)$ ; $P \xleftarrow{\$} \mathbb{G}_1^*$
    $s \xleftarrow{\$} \mathbb{Z}_p^*$ ; $pk \leftarrow (1^k, P, sP, \mathbb{G}_1, \mathbb{G}_2, p, e)$
    $sk \leftarrow (pk, s)$ ; return $(pk, sk)$

$\mathsf{PEKS}^{H_1,H_2,H_3,H_4}(pk, w)$
    parse $pk$ as $(1^k, P, sP, \mathbb{G}_1, \mathbb{G}_2, p, e)$
    if $|w| \geq f(k)$ then return $w$
    $r \xleftarrow{\$} \mathbb{Z}_p^*$ ; $T \leftarrow e(sP, H_1(w))^r$
    $K_1 \leftarrow H_4(T)$ ; $K_2 \leftarrow H_2(T)$
    $K \xleftarrow{\$} \{0,1\}^k$ ; $c \leftarrow K_1 \oplus K$
    $t \leftarrow H_3(K||w)$
    return $(rP, c, t, K_2)$
```

```
$\mathsf{Td}^{H_1}(sk, w)$
    parse $sk$ as $(pk = (1^k, P, sP, \mathbb{G}_1, \mathbb{G}_2, p, e), s)$
    $t_w \leftarrow (pk, sH_1(w), w)$
    return $t_w$

$\mathsf{Test}^{H_1,H_2,H_3,H_4}(t_w, C)$
    parse $t_w$ as $((1^k, P, sP, \mathbb{G}_1, \mathbb{G}_2, p, e), sH_1(w), w)$
    if $|w| \geq f(k)$ then
        if $C = w$ then return 1 else return 0
    if $C$ cannot be parsed as $(rP, c, t, K_2)$ then return 0
    $T \leftarrow e(rP, sH_1(w))$
    $K \leftarrow c \oplus H_4(T)$
    if $K_2 \neq H_2(T)$ then return 0
    if $t = H_3(K||w)$ then return 1 else return 0
```

Figure 2: Algorithms constituting the PEKS scheme $\mathcal{PEKS}\text{-}\mathcal{STAT}$. Here $f(k) = k^{\lg(k)}$, $\mathcal{G}$ is a pairing parameter generator and $H_1\colon \{0,1\}^* \to \mathbb{G}_1$, $H_2\colon \mathbb{G}_2 \to \{0,1\}^{3k}$, $H_3\colon \{0,1\}^* \to \{0,1\}^k$, and $H_4\colon \{0,1\}^* \to \{0,1\}^k$ are random oracles.

3.4 A statistically consistent PEKS scheme

We present the first PEKS scheme that is (PEKS-IND-CPA and) *statistically* consistent. To define the scheme, we first introduce the function $f(k) = k^{\lg(k)}$. (Any function that is super-polynomial but sub-exponential would suffice. This choice is made for concreteness.) The algorithms constituting our scheme $\mathcal{PEKS}\text{-}\mathcal{STAT}$ are then depicted in Figure 2.

The scheme uses ideas from the $\mathcal{BDOP}\text{-}\mathcal{PEKS}$ scheme [7] as well as from the $\mathcal{BF}\text{-}\mathcal{IBE}$ scheme [8], but adds some new elements. Note that the encryption algorithm is trivial, returning the keyword as the ciphertext, when the keyword has length more than $f(k)$. If not, the processing is more complex, depending on some random choices and numerous random oracles. In particular the random choice of "session" key K, and the fact that the random oracle H_2 is length-increasing, are important.

The first thing we stress about the scheme is that the algorithms are PT. This is because PT means in the length of the inputs, and the input of (say) the encryption algorithm includes w as well as 1^k, so it can test whether $|w| \geq f(k)$ in polynomial time. Now the following says that the scheme is private:

Proposition 3.4 *The* $\mathcal{PEKS}\text{-}\mathcal{STAT}$ *scheme is* PEKS-IND-CPA-*secure assuming that the BDH problem is hard relative to generator* $\mathcal{G}$. ▌

Before providing the proof, let us give some intuition. While sending w in the clear looks at first glance like it violates privacy, the reason it does not is that this only happens when w has length at least $f(k)$, and the privacy adversary is poly(k) time and thus cannot even write down such a keyword in order to query it to its challenge oracle. (This is where we use the fact that $f(k)$ is super-polynomial. We will use the fact that it is sub-exponential in the proof of statistical consistency.) The privacy adversary is thus effectively restricted to attacking the scheme only on keywords of size at most $f(k)$. Here, privacy can be reduced to solving the BDH problem using techniques used to prove IBE-IND-CPA of the $\mathcal{BF}\text{-}\mathcal{IBE}$ scheme [8] and to prove anonymity of the same scheme (cf. Theorem 4.4).

Proof of Proposition 3.4: Let $\mathcal{B}$ be a PTA attacking the PEKS-IND-CPA security $\mathcal{PEKS}\text{-}\mathcal{STAT}$ = (KG, PEKS, Td, Test). Say it makes at most q queries to its TRAPD$(\cdot,\cdot)$ oracle and at most q_i queries to H_i for $i = 1, 2, 3$. (These are actually functions of k, but we drop the argument to simplify notation.)

```
Adversary $\mathcal{A}(1^k, (\mathbb{G}_1, \mathbb{G}_2, p, e), P, sP, rP, \alpha P)$
    $pk \leftarrow (1^k, P, sP, p, G_1, G_2, e)$ ; $\mathcal{Q} \leftarrow \emptyset$
    $(w_0, w_1, state) \xleftarrow{\$} \mathcal{B}^{\textsc{Trapd}(\cdot), H_1, H_2, H_3, H_4}(\texttt{find}, pk)$
    $b \xleftarrow{\$} \{0,1\}$ ; $h \leftarrow H_1(w_b)$
    if $d[w_b] = 0$ then abort
    $K_1 \xleftarrow{\$} \{0,1\}^k$ ; $K_2 \xleftarrow{\$} \{0,1\}^{3k}$
    $K \xleftarrow{\$} \{0,1\}^k$ ; $c \leftarrow K_1 \oplus K$ ; $t \leftarrow H_3(K \| w_b)$
    $C \leftarrow (rP, c, t, K_2)$
    $b' \xleftarrow{\$} \mathcal{B}^{\textsc{Trapd}(\cdot), H_1, H_2, H_3, H_4}(\texttt{guess}, C, state)$
    if $\mathcal{Q} \neq \emptyset$ then $T \xleftarrow{\$} \mathcal{Q}$ else abort
    return $T^{x[w_b]^{-1}}$

Oracle $\textsc{Trapd}(w)$
    $h \leftarrow H_1(w)$
    if $d[w] = 1$ then abort
    $t_w \leftarrow (x[w] \cdot sP, w)$
    return $t_w$

Oracle $H_1(w)$
    if $h_1[w]$ is not defined then
        flip biased coin $d[w] \in \{0,1\}$ such that $\Pr[d[w] = 1] = \delta$
        $x[w] \xleftarrow{\$} \mathbb{Z}_p$
        if $d[w] = 0$ then define $h_1[w] \leftarrow x[w] \cdot P$
        else define $h_1[w] \leftarrow x[w] \cdot \alpha P$
    return $h_1[w]$

Oracle $H_2(T)$
    if $h_2[T]$ is not defined then
        define $h_2[T] \xleftarrow{\$} \{0,1\}^{3k}$
        $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{T\}$
    return $h_2[T]$

Oracle $H_3(X)$
    if $h_3[X]$ is not defined then
        define $h_3[X] \xleftarrow{\$} \{0,1\}^k$
    return $h_3[X]$

Oracle $H_4(T)$
    if $h_4[T]$ is not defined then
        define $h_4[T] \xleftarrow{\$} \{0,1\}^k$
        $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{T\}$
    return $h_4[T]$
```

Figure 3: Adversary $\mathcal{A}$ attacking the BDH problem.

We construct a PTA $\mathcal{A}$ attacking the BDH relative to $\mathcal{G}$ such that

$$\mathbf{Adv}^{\mathrm{bdh}}_{\mathcal{G},\mathcal{A}}(k) \geq \frac{1}{e(1+q)\cdot(q_2+q_4)} \cdot \left(\frac{1}{2} \cdot \mathbf{Adv}^{\text{peks-ind-cpa}}_{\mathcal{PEKS},\mathcal{B}}(k) - \frac{q_3}{2^k} \right) . \tag{3}$$

Our adversary $\mathcal{A}$ is shown in Figure 3. We show that $\mathcal{A}$ outputs the correct answer $T = e(P,P)^{rs\alpha}$ with probability at least the quantity on the right-hand-side of Equation (3).

Let $t(k)$ be a polynomial which bounds the running time of $\mathcal{B}$. So there is an integer N such that $t(k) < f(k)$ for all $k \geq N$. Notice that the PEKS algorithm of the PEKS scheme in Figure 2 returns w in the clear when $|w| \geq f(k)$. However, the keywords output by $\mathcal{B}$ in the `find` stage have length at most $t(k)$, so if $k \geq N$, the encryption is done by the code for the case $|w| < f(k)$ shown in PEKS. Since it suffices to prove Equation (3) for all $k \geq N$, we assume that the encryption is done by the code for the case $|w| < f(k)$ shown in PEKS.

Let $\Pr_1[\cdot]$ denote the probability over the experiment for $\mathbf{Adv}^{\mathrm{bdh}}_{\mathcal{G},\mathcal{A}}(k)$ as defined in Section 2. Let E_1 denote the event in this experiment that $\mathcal{A}$ aborts in simulating the trapdoor oracle. Let E_2 denote the event that $d[w_b] = 0$ (which also causes $\mathcal{A}$ to abort). Let E_3 denote the event that $\mathcal{Q} = \emptyset$ (which also causes $\mathcal{A}$ to abort). Let E_4 denote the event that $\mathcal{B}$ issues a query $H_2(e(rP, sH_1(w_b)))$ or $H_4(e(rP, sH_1(w_b)))$. Let $\Pr_2[\cdot]$ denote the probability over $\mathbf{Exp}^{\text{peks-ind-cpa-b}}_{\mathcal{PEKS},\mathcal{B}}$ for a random choice for $b \in \{0,1\}$, and let b' denote the output of $\mathcal{B}$ in this experiment. Let E_5 be the event that $\mathcal{B}$ issues a query $H_2(e(rP, sH_1(w_b)))$ or $H_4(e(rP, sH_1(w_b)))$ to its oracles in this experiment. Let E_6 denote the event that $\mathcal{B}$ issues a query $K \| w_b$ to its oracle H_3, where K is the random k-bit string that

$\mathbf{Exp}^{\text{peks-ind-cpa-b}}_{\mathcal{PEKS},\mathcal{B}}$ used in PEKS when replying $\mathcal{B}$'s challenge after the find stage. Equation (3) follows from the following claims.

Claim 1. $\mathbf{Adv}^{\text{bdh}}_{\mathcal{G},\mathcal{A}}(k) \geq \Pr_1[\neg E_1 \wedge \neg E_2 \wedge \neg E_3 \wedge E_4]/(q_2 + q_4)$.

In the above simulation if none of the events E_1, E_2 and E_3 happens, then $\mathcal{A}$ will randomly choose an element $T \xleftarrow{\$} \mathcal{Q}$ and return $T^{x[w_b]^{-1}}$. However, by definition of event E_1, one of the elements in $\mathcal{Q}$ is equal to $e(P,P)^{sr\alpha \cdot x[w_b]}$, thus $\mathcal{A}$ has at least the probability of $1/|\mathcal{Q}| \geq 1/(q_2 + q_4)$ to give the correct answer to the BDH problem. □

Claim 2. $\Pr[\neg E_1 \wedge \neg E_2 \wedge \neg E_3 \wedge E_4] = \Pr[E_4 | \neg E_1 \wedge \neg E_2] \cdot \Pr[\neg E_1 \wedge \neg E_2]$.

Notice that when event E_4 happens, the set $\mathcal{Q}$ must contain at least one element, thus E_3 is always false. Therefore we have $\Pr_1[\neg E_1 \wedge \neg E_2 \wedge \neg E_3 \wedge E_4] = \Pr_1[\neg E_1 \wedge \neg E_2 \wedge E_4]$. The claim follows by conditioning off of the event $\neg E_1 \wedge \neg E_2$. □

Claim 3. $\Pr_1[E_4 \mid \neg E_1 \wedge \neg E_2] = \Pr_2[E_5]$.

Under the condition that $\mathcal{A}$ does not abort, the simulation is perfect, i.e. all $\mathcal{A}$'s answers to the simulated oracles $\textsc{Trapd}(sk, \cdot)$, $H_1(\cdot) \ldots H_4(\cdot)$ have exactly the same distribution as those in the real PEKS-IND-CPA experiment. □

Claim 4. $\Pr_2[E_5] \geq 1/2 \cdot \mathbf{Adv}^{\text{peks-ind-cpa}}_{\mathcal{PEKS},\mathcal{B}}(k) - q_3 \cdot 2^{-k}$.

First observe that

$$\begin{aligned}
\Pr_2\left[b = b'\right] &= \Pr_2\left[b = b' \wedge E_5\right] + \Pr_2\left[b = b' \wedge \neg E_5 \wedge E_6\right] + \Pr_2\left[b = b' \wedge \neg E_5 \wedge \neg E_6\right] \\
&\leq \Pr_2[E_5] + \Pr_2[E_6] + \Pr_2\left[b = b' \mid \neg E_5 \wedge \neg E_6\right] \cdot \Pr_2[\neg E_5 \wedge \neg E_6] \\
&\leq \Pr_2[E_5] + q_3 \cdot 2^{-k} + \Pr_2\left[b = b' \mid \neg E_5 \wedge \neg E_6\right] \cdot \Pr_2[\neg E_5 \wedge \neg E_6] \qquad (4) \\
&\leq \Pr_2[E_5] + q_3 \cdot 2^{-k} + 1/2\ . \qquad (5)
\end{aligned}$$

Equation (4) comes from the fact that, by assumption, $\mathcal{B}$ makes at most q_3 queries to H_3. Equation (5) comes from the fact that, if E_5 and E_6 both do not occur, $\mathcal{B}$ learns no information from the ciphertext. Rearranging gives

$$\Pr_2[E_5] \;\geq\; \Pr_2\left[b = b'\right] - 1/2 - q_3 \cdot 2^{-k} \;=\; 1/2 \cdot \mathbf{Adv}^{\text{peks-ind-cpa}}_{\mathcal{PEKS},\mathcal{B}}(k) - q_3 \cdot 2^{-k}\ .$$

The last equality follows from the standard result that $\mathbf{Adv}^{\text{peks-ind-cpa}}_{\mathcal{PEKS},\mathcal{B}}(k) = 2 \cdot \Pr_2[b = b'] - 1$. □

Claim 5. $\Pr[\neg E_1 \wedge \neg E_2] \geq 1/(e(q+1))$ for $\delta = 1/(q+1)$.

Since for every keyword w the biased coin $d[w]$ is flipped independently, and $\Pr[d[w] = 1] = \delta$ for all w, let $\mathcal{Q}_T$ be the set of queries issued by $\mathcal{B}$ to the $\textsc{Trapd}(sk, \cdot)$ oracle, then

$$\Pr[\neg E_1 \wedge \neg E_2] = \delta \cdot \prod_{w \in \mathcal{Q}_T} (1 - \delta) = \delta \cdot (1-\delta)^{|\mathcal{Q}_T|} \geq \delta \cdot (1-\delta)^q$$

The last quantity is maximized at $\delta = 1/(q+1)$ with value at least $1/e(q+1)$. ▌

Let us move to the more interesting claim, namely consistency:

Proposition 3.5 *The $\mathcal{PEKS}$-$\mathcal{STAT}$ scheme is statistically consistent.* ▌

Before providing the proof, let us give some intuition. The main issue is that the computationally unbounded consistency adversary $\mathcal{U}$ can easily find any collisions that exist for the random-oracle hash functions. Let w, w' denote the keywords output by the adversary $\mathcal{U}$. We proceed via a case analysis. One can show that if either w or w' have length at least $f(k)$ then Test will not be wrong. The interesting case is when w, w' both have length at most $f(k)$. Let (rP, c, t, K_2) denote the challenge ciphertext formed by encrypting w. Let $T = e(rP, H_1(w))$ and let $K = c \oplus H_4(T)$ be the underlying session key. Let $T' = e(rP, H_1(w'))$ and let $K' = c \oplus H_4(T')$. Now consider two cases.

The first case is that $H_1(w) \neq H_1(w')$. Properties of pairings imply $T \neq T'$. Now we claim that this means $K_2 = H_2(T) \neq H_2(T')$ with high probability, and thus Test will correctly reject, meaning $\mathcal{U}$ does not win. This is *not* merely because H_2 is random, for remember the adversary is not computationally bounded and can search for, and find, any collisions that exist. The reason is that H_2 is with high probability an injective function and collisions for it simply do not exist. The reason for this is that its domain is $\mathbb{G}_2$ which has size $p < 2^{k+1}$ (our definition of a pairing parameter generator required this) but H_2 outputs $3k$ bits, and thus a union bound can be used to show that H_2 is injective except with probability $4 \cdot 2^{-k}$.

The second case, which is the harder one, is that $H_1(w) = H_1(w')$ (again, we cannot prevent $\mathcal{U}$ from finding collisions in H_1), and this is where we will use the fact that $f(k)$ is sub-exponential. Here the idea is that at the time it chooses w, w', adversary $\mathcal{U}$ does not know the value of the session key K that is randomly chosen later. We divide pairs (V, V') of strings of length at most $f(k)$ (candidate keywords) into two classes. A pair is *heavy* if there are "lots" of session keys L such that $H_3(L \,\|\, V) = H_3(L \,\|\, V')$, and *light* otherwise, where "lots" is defined as $2^{k/2}$. Now we again consider two cases. If (w, w') is light then the randomly chosen K has only a $2^{-k/2}$ chance of being a session key for which $H_3(K \,\|\, w) = H_3(K \,\|\, w')$ and thus Test will most likely reject, so $\mathcal{U}$ does not win. Next we use an occupancy problem based counting argument to show that the probability (over H_3) that a particular pair (V, V') of keywords is heavy is *double* exponentially small in k. But the number of choices of keyword pairs is $2^{O(f(k))}$ which is sub-double-exponentially small by choice of $f(k)$, and thus a union bound allows us to conclude that (w, w') is not likely to be heavy.

Proof of Proposition 3.5: Let $\mathcal{U}$ be a computationally unbounded adversary algorithm. We show that there is a constant $c > 0$ such that

$$\mathbf{Adv}^{\text{peks-consist}}_{\mathcal{PEKS},\mathcal{U}}(k) \leq O(2^{-ck}) .$$

Consider the experiment $\mathbf{Exp}^{\text{peks-consist}}_{\mathcal{PEKS},\mathcal{U}}(k)$. Let w, w' denote the keywords output by $\mathcal{U}$ and assume they are distinct, since otherwise $\mathcal{U}$ does not win. Let $\textsc{Win}$ be the event that the experiment outputs 1. Let $r, \mathbf{K}$ be the random choices made by $\mathsf{PEKS}^{H_1,H_2,H_3,H_4}(pk, w)$ in the experiment. Then we let

$$\begin{array}{rclrcl}
\mathbf{T} & = & e(rP, H_1(w)) & \mathbf{T}' & = & e(rP, H_1(w')) \\
\mathbf{c} & = & \mathbf{K} \oplus H_4(\mathbf{T}) & \mathbf{K}' & = & \mathbf{c} \oplus H_4(\mathbf{T}') \\
\mathbf{K}_2 & = & H_2(\mathbf{T}) & \mathbf{K}'_2 & = & H_2(\mathbf{T}') \\
\mathbf{t} & = & H_3(\mathbf{K} \,\|\, w) & \mathbf{t}' & = & H_3(\mathbf{K}' \,\|\, w') .
\end{array}$$

The random choices of H_1, H_2, H_3, H_4, r and $\mathbf{K}$ determine all these random variables. Let $\textsc{Bad}$ be the event that $\mathsf{Test}(t_{w'}, (\mathbf{C}, \mathbf{c}, \mathbf{t}, \mathbf{K}_2)) = 1$. Let $\textsc{Big}$ be the event that either w or w' has length greater than or equal to $f(k)$. Then

$$\begin{array}{rcl}
\mathbf{Adv}^{\text{peks-consist}}_{\mathcal{PEKS},\mathcal{U}}(k) & = & \Pr[\textsc{Bad}] \;\leq\; \Pr[\textsc{Bad} \wedge \textsc{Big}] + \Pr[\textsc{Bad} \wedge \neg\textsc{Big}] \\
& \leq & \Pr[\textsc{Bad} \mid \textsc{Big}] + \Pr[\textsc{Bad} \wedge \neg\textsc{Big}] .
\end{array}$$

Suppose $\textsc{Big}$ holds. If $|w'| \geq f(k)$ then $\mathsf{Test}(t_{w'}, \mathbf{C})$ will return 1 only if $\mathbf{C} = w'$. But this will not be the case because either $|w| \geq f(k)$ and $\mathbf{C} = w \neq w'$, or $|w| < f(k)$ and, for large enough k, $|\mathbf{C}| < f(k) \leq |w'|$. On the other hand if $|w| \geq f(k)$ and $|w'| < f(k)$ then $\mathbf{C} = w$ and the latter cannot be parsed as an appropriate 4-tuple (rP, c, t, K_2), so Test will return 0. We conclude that $\Pr[\,\textsc{Bad} \mid \textsc{Big}\,] = 0$ for all large enough k. We now want to bound

$$\begin{aligned} &\Pr[\,\textsc{Bad} \wedge \neg\textsc{Big}\,] \\ &= \underbrace{\Pr\left[\,\textsc{Bad} \wedge \neg\textsc{Big} \wedge H_1(w) \neq H_1(w')\,\right]}_{p_1} + \underbrace{\Pr\left[\,\textsc{Bad} \wedge \neg\textsc{Big} \wedge H_1(w) = H_1(w')\,\right]}_{p_2} \;. \end{aligned}$$

We bound p_1, p_2 in turn. We let S be the set of all distinct pairs (g, g') of elements in $\mathbb{G}_1$. So p_1 is at most the sum, over all $(g, g') \in S$, of the product terms

$$\Pr\left[\,H_2(e(rP, g)) = H_2(e(rP, g')) \mid (H_1(w), H_1(w')) = (g, g')\,\right] \cdot \Pr\left[\,(H_1(w), H_1(w')) = (g, g')\,\right] \;.$$

Properties of pairings tell us that $g \neq g'$ implies $e(rP, g) \neq e(rP, g')$. So due to the randomness of H_2, the first term of each product above is 2^{-3k}. However, there are at most p^2 choices for the pair (g, g'), and we know that $p < 2^{k+1}$. Thus we have

$$p_1 \leq p^2 \cdot 2^{-3k} \leq 2^{2k+2-3k} = 4 \cdot 2^{-k} \;.$$

(As we discussed above, the intuition here is that with probability at least $1 - 4 \cdot 2^{-k}$ the function H_2 is injective.) We now proceed to bound p_2. In this argument, we regard H_1 as fixed. (Formally, imagine that we condition on a particular choice of H_1. This suffices since what follows holds for all values of this choice.) Let U be the set of all pairs (V, V') of distinct keywords of length at most $f(k)$ each such that $H_1(V) = H_1(V')$. For any $(V, V') \in U$ we let

$$\mathsf{Keys}(V, V') \;=\; \{\, A \in \{0,1\}^k \;:\; H_3(A \,\|\, V) = H_3(A \,\|\, V')\,\} \;.$$

We say that (V, V') is *heavy* if $|\mathsf{Keys}(V, V')| \geq 2^{k/2}$, and *light* otherwise. We let $\mathsf{Lt}(V, V')$ denote the event that (V, V') is light and $\mathsf{Hw}(V, V')$ the event that (V, V') is heavy, where the probability is over the choice of H_3 only. Then $p_2 \leq p_L + p_H$ where

$$\begin{aligned} p_L &= \sum_{(V,V') \in U} \Pr\left[\,\textsc{Bad} \wedge (w, w') = (V, V') \wedge \mathsf{Lt}(V, V')\,\right] \\ p_H &= \sum_{(V,V') \in U} \Pr\left[\,\textsc{Bad} \wedge (w, w') = (V, V') \wedge \mathsf{Hw}(V, V')\,\right] \;. \end{aligned}$$

We bound these in turn. We have

$$\begin{aligned} p_L &= \sum_{(V,V') \in U} \Pr\left[\,\textsc{Bad} \mid (w, w') = (V, V') \wedge \mathsf{Lt}(V, V')\,\right] \cdot \Pr\left[\,(w, w') = (V, V') \wedge \mathsf{Lt}(V, V')\,\right] \\ &\leq \sum_{(V,V') \in U} \frac{2^{k/2}}{2^k} \cdot \Pr\left[\,(w, w') = (V, V') \wedge \mathsf{Lt}(V, V')\,\right] \qquad (6) \\ &= 2^{-k/2} \cdot \sum_{(V,V') \in U} \Pr\left[\,(w, w') = (V, V') \wedge \mathsf{Lt}(V, V')\,\right] \\ &\leq 2^{-k/2} \;. \end{aligned}$$

Equation (6) is justified by the definition of the Test, the fact that $\mathbf{K}$ is chosen at random from $\{0,1\}^k$ and the fact that (V, V') is light. Now we turn to bounding p_H.

Claim. For any $(V, V') \in U$,

$$\Pr\left[\,\mathsf{Hw}(V, V')\,\right] \leq O(2^{-2^{k/2}}) \;,$$

where the probability is only over the choice of H_3.

Note the bound of the claim is double-exponentially small. We prove the claim later. Using it we can conclude via the union bound:

$$\begin{aligned} p_H &= \sum_{(V,V')\in U} \Pr\left[\,\textsc{Bad} \wedge (w,w') = (V,V') \wedge \mathsf{Hw}(V,V')\,\right] \\ &\le \sum_{(V,V')\in U} \Pr\left[\,\mathsf{Hw}(V,V')\,\right] \;\le\; 2^{2+2f(k)} \cdot O(2^{-2^{k/2}}) \;\le\; O(2^{-g(k)})\ , \end{aligned}$$

where $g(k) = 2^{k/2} - 2 - 2f(k) = \Omega(2^{k/2})$. So certainly $2^{-g(k)}$ is $O(2^{-k})$.

Proof of Claim. We use an occupancy problem approach:

$$\begin{aligned} \Pr\left[\,\mathsf{Hw}(V,V')\,\right] &= \sum_{i=2^{k/2}}^{2^k} \binom{2^k}{i} \cdot (2^{-k})^i \cdot (1-2^{-k})^{2^k-i} \;\le\; \sum_{i=2^{k/2}}^{2^k} \binom{2^k}{i} \cdot (2^{-k})^i \\ &\le \sum_{i=2^{k/2}}^{2^k} \left(\frac{2^k \cdot e}{i}\right)^i \cdot (2^{-k})^i \;\le\; \sum_{i=2^{k/2}}^{2^k} \left(\frac{e}{i}\right)^i \;\le\; \sum_{i=2^{k/2}}^{\infty} \left(\frac{e}{i}\right)^i . \end{aligned}$$

Let $x = e2^{-k/2}$. For sufficiently large k (specifically, $k \ge 6$) we have $x \le 1/2$. So the above is at most

$$\sum_{i=2^{k/2}}^{\infty} x^i \;=\; x^{2^{k/2}} \cdot \sum_{i=0}^{\infty} x^i \;=\; x^{2^{k/2}} \frac{1}{1-x} \;\le\; \frac{2}{2^{2^{k/2}}}\ ,$$

as desired. ▌

4 PEKS and anonymous IBE

We formally define anonymity of IBE schemes and investigate the relation between PEKS and anonymous IBE.

4.1 Definitions

IBE schemes. An *identity-based encryption* (IBE) scheme [19, 8] $\mathcal{IBE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$ consists of four PTAs. Via $(pk, msk) \xleftarrow{\$} \mathsf{Setup}(1^k)$ the randomized key-generation algorithm produces master keys for security parameter $k \in \mathbb{N}$; via $usk[id] \xleftarrow{\$} \mathsf{KeyDer}^H(msk, id)$ the master computes the secret key for identity id; via $C \xleftarrow{\$} \mathsf{Enc}^H(pk, id, M)$ a sender encrypts a message M to identity id to get a ciphertext; via $M \leftarrow \mathsf{Dec}^H(usk, C)$ the possessor of secret key usk decrypts ciphertext C to get back a message. Here H is a random oracle with domain and range possibly depending on k and pk. Associated to the scheme is a message space MsgSp obeying the conventions discussed in Section 2. For consistency, we require that for all $k \in \mathbb{N}$, all identities id and messages $M \in \mathsf{MsgSp}(k)$ we have $\Pr[\mathsf{Dec}^H(\mathsf{KeyDer}^H(msk, id), \mathsf{Enc}^H(pk, id, M)) = M] = 1$, where the probability is taken over the choice of $(pk, msk) \xleftarrow{\$} \mathsf{Setup}(1^k)$, the random choice of H, and the coins of all the algorithms in the expression above.

Privacy and anonymity. Privacy (IBE-IND-CPA) follows [8] while anonymity (IBE-ANO-CPA) is a straightforward adaptation of [2] to IBE schemes. Let $\mathcal{IBE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$ be an IBE scheme with associated message space MsgSp. To an adversary $\mathcal{A}$ and bit $b \in \{0,1\}$, we associate the following experiments:

```
Experiment $\mathbf{Exp}^{\text{ibe-ind-cpa-b}}_{\mathcal{IBE},\mathcal{A}}(k)$
    $IDSet \leftarrow \emptyset$ ; $(pk, msk) \xleftarrow{\$} \mathsf{Setup}(1^k)$
    pick random oracle $H$
    $(id, M_0, M_1, state) \xleftarrow{\$} \mathcal{A}^{\text{KeyDer}(\cdot),H}(\mathtt{find}, pk)$
    $C \xleftarrow{\$} \mathsf{Enc}^H(pk, id, M_b)$
    $b' \xleftarrow{\$} \mathcal{A}^{\text{KeyDer}(\cdot),H}(\mathtt{guess}, C, state)$
    if $\{M_0, M_1\} \not\subseteq \mathsf{MsgSp}(k)$ then return 0
    if $id \notin IDSet$ and $|M_0| = |M_1|$
    then return $b'$ else return 0

Experiment $\mathbf{Exp}^{\text{ibe-ano-cpa-b}}_{\mathcal{IBE},\mathcal{A}}(k)$
    $IDSet \leftarrow \emptyset$ ; $(pk, msk) \xleftarrow{\$} \mathsf{Setup}(1^k)$
    pick random oracle $H$
    $(id_0, id_1, M, state) \xleftarrow{\$} \mathcal{A}^{\text{KeyDer},H}(\mathtt{find}, pk)$
    $C \xleftarrow{\$} \mathsf{Enc}^H(pk, id_b, M)$
    $b' \xleftarrow{\$} \mathcal{A}^{\text{KeyDer},H}(\mathtt{guess}, C, state)$
    if $M \notin \mathsf{MsgSp}(k)$ then return 0
    if $\{id_0, id_1\} \cap IDSet = \emptyset$
    then return $b'$ else return 0
```

where the oracle $\text{KeyDer}(id)$ is defined as

$$IDSet \leftarrow IDSet \cup \{id\} \ ; \ usk[id] \xleftarrow{\$} \mathsf{KeyDer}^H(msk, id) \ ; \ \text{Return } usk[id]$$

For prop $\in \{\text{ind}, \text{ano}\}$, we define the advantage of $\mathcal{A}$ in the corresponding experiment as

$$\mathbf{Adv}^{\text{ibe-prop-cpa}}_{\mathcal{IBE},\mathcal{A}}(k) \ = \ \Pr\left[\mathbf{Exp}^{\text{ibe-prop-cpa-1}}_{\mathcal{IBE},\mathcal{A}}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{ibe-prop-cpa-0}}_{\mathcal{IBE},\mathcal{A}}(k) = 1\right].$$

IBE scheme $\mathcal{IBE}$ is said to be IBE-IND-CPA-*secure* (resp., IBE-ANO-CPA-*secure*) if the respective advantage function is negligible for all PTAs $\mathcal{A}$.

4.2 The bdop-ibe-2-peks transform

The bdop-ibe-2-peks transform [7] takes input an IBE scheme $\mathcal{IBE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$ and returns a PEKS scheme $\mathcal{PEKS} = (\mathsf{KG}, \mathsf{Td}, \mathsf{PEKS}, \mathsf{Test})$ as follows. The public key pk and secret key sk of the receiver in the PEKS scheme are the master public and secret keys, respectively, of the IBE scheme (i.e., $\mathsf{KG} = \mathsf{Setup}$). The trapdoor associated to keyword w is the secret key that the IBE scheme would assign to the identity w (i.e., $\mathsf{Td}(sk, w) = \mathsf{KeyDer}(sk, w)$). A keyword w is PEKS-encrypted by IBE-encrypting the message 0^k for the identity w (i.e., $\mathsf{PEKS}(pk, w) = \mathsf{Enc}(pk, w, 0^k)$). Finally, testing is done by checking that the ciphertext decrypts to 0^k (i.e., $\mathsf{Test}(t_w, C)$ returns 1 iff $\mathsf{Dec}(t_w, C) = 0^k$).

We know that $\mathcal{BF}\text{-}\mathcal{IBE}$ is anonymous (Theorem 4.4), that $\mathcal{BDOP}\text{-}\mathcal{PEKS} = \mathsf{bdop\text{-}ibe\text{-}2\text{-}peks}(\mathcal{BF}\text{-}\mathcal{IBE})$, and that $\mathcal{BDOP}\text{-}\mathcal{PEKS}$ is not statistically consistent (Proposition 3.2). We can conclude that the bdop-ibe-2-peks transformation does not necessarily yield a statistically consistent PEKS scheme. The following strengthens this, showing that (under the minimal assumption of the existence of some IBE-IND-CPA- and IBE-ANO-CPA-secure IBE scheme) the transform does not necessarily even yield a computationally consistent PEKS scheme. This means that the transform is not in general a suitable way to obtain a PEKS scheme.

Theorem 4.1 *Assume there exist* IBE-ANO-CPA-*secure and* IBE-IND-CPA-*secure IBE schemes. Then there exists a* IBE-ANO-CPA-*secure and* IBE-IND-CPA-*secure IBE scheme* $\overline{\mathcal{IBE}}$ *such that the PEKS scheme* $\mathcal{PEKS}$ *derived from* $\overline{\mathcal{IBE}}$ *via* bdop-ibe-2-peks *is not computationally consistent.*

Proof: Let $\mathcal{IBE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$ be any IBE-ANO-CPA- and IBE-IND-CPA-secure IBE scheme. We construct the IBE scheme $\overline{\mathcal{IBE}} = (\overline{\mathsf{Setup}}, \mathsf{KeyDer}, \overline{\mathsf{Enc}}, \overline{\mathsf{Dec}})$ to have the master-key generation, encryption and decryption algorithms shown in Figure 4. (We assume that the message space $\mathsf{MsgSp}(k)$ of $\mathcal{IBE}$ includes $\{0,1\}^{2k}$. This is wlog since by definition it includes $\{0,1\}^k$ and hence hybrid encryption could be used to encrypt messages of length more than k bits.) The IBE-IND-CPA-

```
$\overline{\mathsf{Setup}}(1^k)$
    $(pk, msk) \xleftarrow{\$} \mathsf{Setup}(1^k)$
    $R \xleftarrow{\$} \{0,1\}^k$ ; $\overline{pk} \leftarrow (pk, R)$
    return $(\overline{pk}, msk)$

$\overline{\mathsf{Enc}}(\overline{pk}, id, M)$
    parse $\overline{pk}$ as $(pk, R)$
    $C \xleftarrow{\$} \mathsf{Enc}(pk, id, M\|R)$
    return $C$

$\overline{\mathsf{Dec}}(\overline{pk}, sk, C)$
    parse $\overline{pk}$ as $(pk, R)$
    $X \leftarrow \mathsf{Dec}(pk, sk, C)$
    parse $X$ as $M\|R'$ where $|R'| = k$
    if $R' = R$ then return $M$
    else return $0^k$
```

Figure 4: IBE scheme for proof of Theorem 4.1.

and IBE-ANO-CPA-security of $\overline{\mathcal{IBE}}$ follows from simple reductions from the security of $\mathcal{IBE}$, and we omit the details here. To show that $\mathcal{PEKS}$ is not computationally consistent, we let w, w' be any two distinct keywords. We then let $\mathcal{B}$ be the PT adversary against the computational consistency of $\mathcal{PEKS}$ that (given the public key) simply returns these two keywords. We claim that $\mathcal{B}$ succeeds with probability almost one. More precisely, we claim that there is a negligible function ν such that $\mathbf{Adv}^{\text{peks-consist}}_{\mathcal{PEKS},\mathcal{B}}(k) \geq 1 - \nu(k)$. Note the attack is strong in the sense that it essentially says that consistency fails with respect to all pairs of distinct keywords.

To prove this claim, we construct a PT adversary $\mathcal{A}$ against the IBE-IND-CPA-security of $\mathcal{IBE}$ such that

$$\mathbf{Adv}^{\text{ibe-ind-cpa}}_{\mathcal{IBE},\mathcal{A}}(k) + \mathbf{Adv}^{\text{peks-consist}}_{\mathcal{PEKS},\mathcal{B}}(k) \geq 1 - 2^{-k} .$$

Since we assume that $\mathcal{IBE}$ satisfies IBE-IND-CPA-security, the function $\mathbf{Adv}^{\text{ibe-ind-cpa}}_{\mathcal{IBE},\mathcal{A}}(k)$ is negligible, and hence the claim follows.

In its `find` stage, given input master public key pk, adversary $\mathcal{A}$ returns as challenge identity w and as challenge messages $0^k \parallel R_0$ and $0^k \parallel R_1$ where $R_0, R_1 \xleftarrow{\$} \{0,1\}^k$. In the `guess` stage, given challenge ciphertext C (that encrypts $0^k \parallel R_b$ under identity w for challenge bit $b \in \{0,1\}$), it runs the decryption algorithm Dec with secret key $usk[w']$, having obtained the latter via its key-derivation oracle, and returns 0 if the last k bits of the resulting keyword are R_0 and 1 otherwise. For the analysis, let E_0 denote the event that the string comprising the last k bits of $\mathsf{Dec}(usk[w'], \mathsf{Enc}(pk, w, 0^k\|R_0)))$ is equal to R_0, and E_1 the event that the string comprising the last k bits of $\mathsf{Dec}(pk, usk[w'], \mathsf{Enc}(pk, w, 0^k\|R_1)))$ is equal to R_0. Then

$$\begin{aligned}\Pr[\mathbf{Exp}^{\text{ibe-ind-cpa-1}}_{\mathcal{IBE},\mathcal{A}}(k) = 1] &= \Pr[\neg E_1] \;=\; 1 - 2^{-k} \\ \Pr[\mathbf{Exp}^{\text{ibe-ind-cpa-0}}_{\mathcal{IBE},\mathcal{A}}(k) = 1] &= \Pr[\neg E_0]\end{aligned}$$

where the probability is over random choices of $(pk, msk) \xleftarrow{\$} \mathsf{Setup}(1^k)$ and random choices of strings R_0, R_1. Equation (7) holds because R_0, R_1 are independent, random k-bit strings. Now, the probability that $\mathcal{B}$ wins in the computational consistency experiment for $\mathcal{PEKS} = \mathsf{bdop\text{-}ibe\text{-}2\text{-}peks}(\overline{\mathcal{IBE}})$ is at least the probability that the last k bits of $\mathsf{Dec}(usk[w'], \mathsf{Enc}(pk, w, 0^k\|R_0))$ equal R_0, since in this case $\overline{\mathsf{Enc}}((pk, R_0), w, 0^k)$ returns 0^k. That is, its advantage is at least $\Pr[\neg E_0]$. So

$$\begin{aligned}\mathbf{Adv}^{\text{ibe-ind-cpa}}_{\mathcal{IBE},\mathcal{A}}(k) + \mathbf{Adv}^{\text{peks-consist}}_{\mathcal{PEKS},\mathcal{B}}(k) &= \Pr[\neg E_1] - \Pr[\neg E_0] + \mathbf{Adv}^{\text{peks-consist}}_{\mathcal{PEKS},\mathcal{B}}(k) \\ &\geq (1 - 2^{-k}) - \Pr[\neg E_0] + \Pr[\neg E_0] \\ &= 1 - 2^{-k}\end{aligned}$$

as claimed. ▌

4.3 The new-ibe-2-peks transformation

The negative result in Theorem 4.1 raises the question: Does the existence of IBE schemes imply the existence of computationally consistent PEKS schemes? We answer that in the affirmative by presenting a revision to the BDOP transformation, called new-ibe-2-peks, that transforms any IBE-IND-CPA- and IBE-ANO-CPA-secure IBE scheme into a PEKS-IND-CPA-secure and computationally consistent PEKS scheme. It is similar to bdop-ibe-2-peks except that instead of always using 0^k as the message encrypted, the PEKS-encryption algorithm chooses and encrypts a random message R and appends R in the clear to the ciphertext. In more detail, the new-ibe-2-peks transform takes input an IBE scheme $\mathcal{IBE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$ and returns a PEKS scheme $\mathcal{PEKS} = (\mathsf{KG}, \mathsf{Td}, \mathsf{PEKS}, \mathsf{Test})$ as follows. The public key pk and secret key sk of the receiver in the PEKS scheme are the master public and secret keys, respectively, of the IBE scheme. (I.e. $\mathsf{KG} = \mathsf{Setup}$.) The trapdoor associated to keyword w is the secret key that the IBE scheme would assign to the identity w. (I.e. $\mathsf{Td}(sk, w) = \mathsf{KeyDer}(sk, w)$.) PEKS-encryption of keyword w is done as follows: $\mathsf{PEKS}(pk, w)$ picks $C_2 \xleftarrow{\$} \{0,1\}^k$, lets $C_1 \xleftarrow{\$} \mathsf{Enc}(pk, w, C_2)$, and returns (C_1, C_2). Finally, $\mathsf{Test}(t_w, (C_1, C_2))$ returns 1 iff $\mathsf{Dec}(t_w, C_1) = C_2$.

Intuitively, this construction avoids the problem of oddly-behaving Dec algorithms by making sure that the *only* way to ruin the consistency of the PEKS scheme is by correctly guessing the value encrypted by a ciphertext, using the secret key of a different identity, which should not be possible for an IBE-IND-CPA-secure IBE scheme. Hence, the consistency of the resulting PEKS scheme is due to the data privacy property of the IBE scheme, while the data privacy property of the PEKS scheme comes from the anonymity of the IBE scheme. The formal result statement and proof follow.

Theorem 4.2 *Let $\mathcal{IBE}$ be an IBE scheme and let $\mathcal{PEKS}$ be the PEKS scheme derived from $\mathcal{IBE}$ via* new-ibe-2-peks. *If $\mathcal{IBE}$ is* IBE-IND-CPA-*secure, then $\mathcal{PEKS}$ is computationally consistent. Further, if $\mathcal{IBE}$ is* IBE-ANO-CPA-*secure, then $\mathcal{PEKS}$ is* PEKS-IND-CPA-*secure.*

Proof: Let $\mathcal{U}$ be any PTA attacking the computational consistency of $\mathcal{PEKS}$, and consider the following PTA $\mathcal{A}$ attacking the IBE-IND-CPA-security of $\mathcal{IBE}$. In its `find` stage, given master public key pk, adversary $\mathcal{A}$ runs $\mathcal{U}(pk)$ to get keywords w, w'. It returns w as the challenge identity and $R_0, R_1 \xleftarrow{\$} \{0,1\}^k$ as the challenge messages. In the `guess` stage, given challenge ciphertext C (that encrypts R_b under identity w for challenge bit $b \in \{0,1\}$), $\mathcal{A}$ uses its key-derivation oracle to obtain a trapdoor $t_{w'}$ for w'. If $\mathsf{Dec}(t_{w'}, C) = R_1$ then it returns 1 else it returns 0. It is easy to see that

$$\begin{aligned}
\Pr\left[\mathbf{Exp}^{\text{ibe-ind-cpa-1}}_{\mathcal{IBE},\mathcal{A}}(k)] = 1\right] &\geq \Pr\left[\mathbf{Exp}^{\text{peks-consist}}_{\mathcal{PEKS},\mathcal{U}}(k) = 1\right] \\
\Pr\left[\mathbf{Exp}^{\text{ibe-ind-cpa-0}}_{\mathcal{IBE},\mathcal{A}}(k)] = 1\right] &\leq 2^{-k} \; .
\end{aligned}$$

Thus $\mathbf{Adv}^{\text{peks-consist}}_{\mathcal{PEKS},\mathcal{U}}(k) \leq \mathbf{Adv}^{\text{ibe-ind-cpa}}_{\mathcal{IBE},\mathcal{A}}(k) + 2^{-k}$, proving the first claim of the theorem.

Let $\mathcal{B}$ be any PTA attacking the PEKS-IND-CPA-security of $\mathcal{PEKS}$, and consider the following PTA $\mathcal{A}$ attacking the IBE-ANO-CPA-security of $\mathcal{IBE}$. In its `find` stage, given master public key pk, adversary $\mathcal{A}$ runs $\mathcal{B}(\texttt{find}, pk)$ to get challenge keywords w_0, w_1, which it returns along with a message $R \xleftarrow{\$} \{0,1\}^k$. In the `guess` stage, given challenge ciphertext C (that encrypts R under identity w_b for challenge bit $b \in \{0,1\}$), $\mathcal{A}$ runs $\mathcal{B}$, in its `guess` stage, with challenge ciphertext (C, R), to get its guess bit b', which $\mathcal{A}$ returns. In both stages, $\mathcal{A}$ answers any trapdoor-oracle queries of $\mathcal{B}$ via its key-derivation oracle. It is easy to see that for $b = 0, 1$,

$$\Pr\left[\mathbf{Exp}^{\text{ibe-ano-cpa-b}}_{\mathcal{IBE},\mathcal{A}}(k) = 1\right] = \Pr\left[\mathbf{Exp}^{\text{peks-ind-cpa-b}}_{\mathcal{PEKS},\mathcal{B}}(k) = 1\right] .$$

Thus $\mathbf{Adv}^{\text{peks-ind-cpa}}_{\mathcal{PEKS},\mathcal{B}}(k) \leq \mathbf{Adv}^{\text{ibe-ano-cpa}}_{\mathcal{IBE},\mathcal{A}}(k)$, proving the second claim of the theorem. ▌

4.4 A sufficient condition for anonymity

Halevi [16] provides a simple sufficient condition for an IND-CPA public-key encryption scheme to meet the notion of anonymity (a.k.a. key-privacy) of [2]. The condition is that even a computationally unbounded adversary, given public keys pk_0, pk_1 and the encryption of a random message under pk_b, have only a negligible advantage in determining the random challenge bit b. Towards finding anonymous IBE schemes (a task motivated by Theorem 4.2) we extend Halevi's condition to identity-based encryption. In the process we also extend it in two other ways: first to handle the random oracle model (the standard model is a special case) and second to weaken the statistical (i.e. information-theoretic) requirement of [16] to a computational one. (The application of this paper does not need the last extension, but it may be useful in other contexts.)

We begin by defining a relevant (new) notion of security that we call IBE-ANO-RE-CPA. Let $\mathcal{IBE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$ be an IBE scheme with associated message space MsgSp. We associate to an adversary $\mathcal{A}$ and bit $b \in \{0, 1\}$ the following experiment:

```
Experiment $\mathbf{Exp}^{\text{ibe-ano-re-b}}_{\mathcal{IBE},\mathcal{A}}(k)$
    $IDSet \leftarrow \emptyset$ ; $(pk, msk) \xleftarrow{\$} \mathsf{Setup}(1^k)$
    pick random oracle $H$
    $(id_0, id_1, M, state) \xleftarrow{\$} \mathcal{A}^{\text{KeyDer}(\cdot),H}(\mathtt{find}, pk)$
    if $M \notin \mathsf{MsgSp}(k)$ then return 0
    $R \xleftarrow{\$} \{0,1\}^{|M|}$ ; $C \xleftarrow{\$} \mathsf{Enc}^H(pk, id_b, R)$
    $b' \xleftarrow{\$} \mathcal{A}^{\text{KeyDer}(\cdot),H}(\mathtt{guess}, C, state)$
    if $\{id_0, id_1\} \cap IDSet = \emptyset$ then return $b'$
    else return 0

Oracle $\text{KeyDer}(id)$
    $IDSet \leftarrow IDSet \cup \{id\}$
    $usk[id] \xleftarrow{\$} \mathsf{KeyDer}^H(msk, id)$
    return $usk[id]$
```

The IBE-ANO-RE-CPA-*advantage* of an adversary $\mathcal{A}$ in violating the anonymity of the scheme $\mathcal{IBE}$ is defined as

$$\mathbf{Adv}^{\text{ibe-ano-re}}_{\mathcal{IBE},\mathcal{A}}(k) = \Pr\left[\mathbf{Exp}^{\text{ibe-ano-re-1}}_{\mathcal{IBE},\mathcal{A}}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{ibe-ano-re-0}}_{\mathcal{IBE},\mathcal{A}}(k) = 1\right] .$$

A scheme $\mathcal{IBE}$ is said to be IBE-ANO-RE-CPA-*secure* if the above advantage is a negligible function in k for all PTAs $\mathcal{A}$.

Lemma 4.3 *Let $\mathcal{IBE}$ be an IBE scheme that is* IBE-IND-CPA *and* IBE-ANO-RE-CPA-*secure. Then it is also* IBE-ANO-CPA-*secure.* ▌

Proof of Lemma 4.3: The proof is a simple hybrid argument. Let $\mathcal{A}$ be a PTA attacking the IBE-ANO-CPA-security of $\mathcal{IBE}$. It is easy to construct PTAs $\mathcal{A}_1, \mathcal{A}_3$ attacking the IBE-IND-CPA-security of $\mathcal{IBE}$, and PTA $\mathcal{A}_2$ attacking the IBE-ANO-RE-CPA-security of $\mathcal{IBE}$, such that

$$\begin{aligned}
\Pr\left[\mathbf{Exp}^{\text{ibe-ano-cpa-1}}_{\mathcal{IBE},\mathcal{A}}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{ibe-ano-re-1}}_{\mathcal{IBE},\mathcal{A}}(k) = 1\right] &\leq \mathbf{Adv}^{\text{ibe-ind-cpa}}_{\mathcal{IBE},\mathcal{A}_1}(k) \\
\Pr\left[\mathbf{Exp}^{\text{ibe-ano-re-1}}_{\mathcal{IBE},\mathcal{A}}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{ibe-ano-re-0}}_{\mathcal{IBE},\mathcal{A}}(k) = 1\right] &\leq \mathbf{Adv}^{\text{ibe-ano-re}}_{\mathcal{IBE},\mathcal{A}_2}(k) \\
\Pr\left[\mathbf{Exp}^{\text{ibe-ano-re-0}}_{\mathcal{IBE},\mathcal{A}}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{ibe-ano-cpa-0}}_{\mathcal{IBE},\mathcal{A}}(k) = 1\right] &\leq \mathbf{Adv}^{\text{ibe-ind-cpa}}_{\mathcal{IBE},\mathcal{A}_3}(k) .
\end{aligned}$$

Summing concludes the proof. We omit the details, save to remark that we use here the second convention about message spaces noted in Section 2. ▌

```
Setup$(1^k)$
    $(\mathbb{G}_1, \mathbb{G}_2, p, e) \xleftarrow{\$} \mathcal{G}(1^k)$ ; $P \xleftarrow{\$} \mathbb{G}_1^*$ ; $s \xleftarrow{\$} \mathbb{Z}_p^*$
    $pk \leftarrow (\mathbb{G}_1, \mathbb{G}_2, p, e, P, sP)$ ; $msk \leftarrow s$
    return $(pk, msk)$

KeyDer$^{H_1}(msk, id)$
    $sk[id] \leftarrow sH_1(id)$
    return $sk[id]$

Enc$^{H_1,H_2}(pk, id, M)$
    $r \xleftarrow{\$} \mathbb{Z}_p^*$ ; $T \leftarrow e(H_1(id), sP)^r$
    $C \leftarrow (rP, M \oplus H_2(T))$
    return $C$

Dec$^{H_2}(sk[id], C)$
    parse $C$ as $(U, V)$
    $T \leftarrow e(sk[id], U)$ ; $M \leftarrow V \oplus H_2(T)$
    return $M$
```

Figure 5: Algorithms of the IBE scheme $\mathcal{BF}\text{-}\mathcal{IBE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$. Here $\mathcal{G}$ is a pairing parameter generator and $H_1\colon \{0,1\}^* \rightarrow \mathbb{G}_1^*$ and $H_2\colon \mathbb{G}_2 \rightarrow \{0,1\}^k$ are random oracles. The message space is defined by $\mathsf{MsgSp}(k) = \{0,1\}^k$ for all $k \in \mathbb{N}$.

4.5 Anonymity of $\mathcal{BF}\text{-}\mathcal{IBE}$

The Boneh-Franklin BasicIdent IBE scheme [8] is shown in Figure 5. We apply Lemma 4.3 to give a simple proof that it is IBE-ANO-CPA.

Theorem 4.4 *The $\mathcal{BF}\text{-}\mathcal{IBE}$ scheme is* IBE-ANO-CPA*-secure assuming that the BDH is hard relative to generator $\mathcal{G}$.*

Proof: Given Lemma 4.3, and given that the $\mathcal{BF}\text{-}\mathcal{IBE}$ scheme is IBE-IND-CPA-secure [8], it suffices to show that the scheme is IBE-ANO-RE-CPA-secure. Notice that the ciphertext C in Figure 5 has two parts, namely $U = rP$ and $V = M \oplus H_2(T)$. The value U is chosen uniformly at random from $\mathbb{G}_1^*$ by the encryption algorithm. If the message M is chosen uniformly at random from $\{0,1\}^k$, then V is also uniformly distributed in $\{0,1\}^k$ and independent of the $H_2(T)$. Thus in both the 0- and 1- worlds of the IBE-ANO-RE-CPA-security game, the challenge ciphertext C has exactly the same distribution. Therefore any adversary against IBE-ANO-RE-CPA-security will have 0 advantage. ▌

5 Anonymous HIBE

5.1 Definitions

HIBE SCHEMES. A *hierarchical identity-based encryption* (HIBE) scheme [17, 12, 6] is a generalization of an IBE scheme in which an identity is a vector of strings $id = (id_1, \ldots, id_l)$ with the understanding that when $l = 0$ this is the empty vector $()$. The number of components in this vector is called the level of the identity and is denoted $|id|$. If $0 \leq i \leq l$ then $id|_i = (id_1, \ldots, id_i)$ denotes the vector containing the first i components of id. If $|id'| \geq l + 1$ $(l \geq 0)$ and $id'|_l = id$ then we say that id is an ancestor of id', or equivalently, that id' is a descendant of id. If the level of id' is $l + 1$ then id is a parent of id', or, equivalently, id' is a child of id. For any id with $|id| \geq 1$ we let $\mathrm{par}(id) = id|_{|id|-1}$ denote its parent. Two nodes $id = (id_1, \ldots, id_l)$ and $id' = (id'_1, \ldots, id'_l)$ at level l are said to be siblings iff $id|_{l-1} = id'|_{l-1}$. Moreover, if $id_l < id'_l$ in lexicographic order, then id is a left sibling of id' and id' is a right sibling of id. An identity at level one or more can be issued a secret key by its parent. (And thus an identity can issue keys for any of its descendants if necessary.)

Formally a HIBE scheme $\mathcal{HIBE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$ consists of four PTAs. Via $(pk, msk = usk[()]) \xleftarrow{\$} \mathsf{Setup}(1^k)$, where $k \in \mathbb{N}$ is a security parameter, the randomized key-generation algorithm produces master keys, with the secret key being associated to the (unique) identity () at level 0; via $usk[id] \xleftarrow{\$} \mathsf{KeyDer}^H(usk[\mathrm{par}(id)], id)$ the parent of an identity id with $|id| \geq 1$ can compute a secret key for id; via $C \xleftarrow{\$} \mathsf{Enc}^H(pk, id, M)$ a sender encrypts a message M to identity id to get a ciphertext; via $M \leftarrow \mathsf{Dec}^H(usk[id], C)$ the identity id decrypts ciphertext C to get back a message. Here H is a random oracle with domain and range possibly depending on k and pk. Associated to the scheme is a message space MsgSp obeying the conventions discussed in Section 2. For consistency, we require that for all $k \in \mathbb{N}$, all identities id with $|id| \geq 1$ and all messages $M \in \mathsf{MsgSp}(k)$,

$$\Pr\left[\,\mathsf{Dec}^H(\mathsf{KeyDer}^H(usk[\mathrm{par}(id)], id), \mathsf{Enc}^H(pk, id, M)) = M\,\right] \;=\; 1\;,$$

where the probability is taken over the choice of $(pk, usk[()]) \xleftarrow{\$} \mathsf{Setup}(1^k)$, the random choice of H, and the coins of all the algorithms in the expression above.

Privacy and anonymity. The notion of privacy for HIBE schemes is analogous to that for IBE schemes (IBE-IND-CPA) but using identity vectors rather than identity strings and where the adversary is not allowed to query the KeyDer oracle for the secret key of any ancestor of the identity under attack. Since we will deal with schemes where privacy holds only up to some level, the notion is parameterized by a maximum depth function $d\colon \mathbb{N} \to \mathbb{N}$, and all identities id (in queries or challenges) must have $|id| \leq d(k)$. To allow a fine-grained treatment of anonymity we introduce the concept of anonymity at a set $L(k)$ of levels, meaning that in an experiment the adversary $\mathcal{A}$ is challenged to distinguish two distinct identities differing only at levels $l \in L(k)$. (Here for each k, $L(k)$ is a finite set of integers. For ease of notation, we will write l rather than $\{l\}$ when $L(k) = \{l\}$ is a singleton set.)

Formally, let $\mathcal{HIBE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$ be an identity-based encryption scheme with message space MsgSp, let $d : \mathbb{N} \to \mathbb{N}$ be the maximum depth, and let L be a set of levels. Let $\mathrm{diff}(\cdot,\cdot)$ be the function that returns the set of coordinates at which the input identities differ, and $\mathrm{anc}(\cdot)$ the function returning the set of ancestors of the input identity. To any bit $b \in \{0,1\}$ and any adversary $\mathcal{A}$, we associate the experiments:

```
Experiment $\mathbf{Exp}^{\text{hibe-ind-cpa-b}[d]}_{\mathcal{HIBE},\mathcal{A}}(k)$
    $IDSet \leftarrow \emptyset$ ; $(pk, msk) \xleftarrow{\$} \mathsf{Setup}(1^k)$
    pick random oracle $H$
    $(id, M_0, M_1, state) \xleftarrow{\$} \mathcal{A}^{\text{KeyDer}(\cdot),H}(\mathtt{find}, pk)$
    if  $|M_0| \neq |M_1|$ or  $|id| > d(k)$
        or  $\{M_0, M_1\} \not\subseteq \mathsf{MsgSp}(k)$
        then return 0
    $C \xleftarrow{\$} \mathsf{Enc}^H(pk, id, M_b)$
    $b' \xleftarrow{\$} \mathcal{A}^{\text{KeyDer}(\cdot),H}(\mathtt{guess}, C, state)$
    if $IDSet \cap \mathrm{anc}(id) = \emptyset$
       then return $b'$ else return 0
```

```
Experiment $\mathbf{Exp}^{\text{hibe-ano-cpa-b}[L,d]}_{\mathcal{HIBE},\mathcal{A}}(k)$
    $IDSet \leftarrow \emptyset$ ; $(pk, msk) \xleftarrow{\$} \mathsf{Setup}(1^k)$
    pick random oracle $H$
    $(id_0, id_1, M, state) \xleftarrow{\$} \mathcal{A}^{\text{KeyDer}(\cdot),H}(\mathtt{find}, pk)$
    if  $|id_0| \neq |id_1|$ or $|id_0| > d(k)$
        or  $|id_1| > d(k)$ or $M \notin \mathsf{MsgSp}(k)$
        then return 0
    $C \xleftarrow{\$} \mathsf{Enc}(pk, id_b, M)$
    $b' \xleftarrow{\$} \mathcal{A}^{\text{KeyDer}(\cdot),H}(\mathtt{guess}, C, state)$
    if $IDSet \cap (\mathrm{anc}(id_0) \cup \mathrm{anc}(id_1)) = \emptyset$ and
       $\mathrm{diff}(id_0, id_1) \subseteq L(k)$
       then return $b'$ else return 0
```

where the oracle KeyDer$(\cdot)$ is defined as

$$\text{if } |id| > d(k) \text{ then return } \perp\,;\ IDSet \leftarrow IDSet \cup \{id\}\,;\ \text{return } \mathsf{KeyDer}(msk, id)\;.$$

We define the advantage of $\mathcal{A}$ in the corresponding experiments as

$$\mathbf{Adv}^{\text{hibe-ind-cpa}[d]}_{\mathcal{HIBE},\mathcal{A}}(k) \;=\; \Pr\left[\mathbf{Exp}^{\text{hibe-ind-cpa-1}[d]}_{\mathcal{HIBE},\mathcal{A}}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{hibe-ind-cpa-0}[d]}_{\mathcal{HIBE},\mathcal{A}}(k) = 1\right]$$

$$\mathbf{Adv}^{\text{hibe-ano-cpa}[L,d]}_{\mathcal{HIBE},\mathcal{A}}(k) \;=\; \Pr\left[\mathbf{Exp}^{\text{hibe-ano-cpa-1}[L,d]}_{\mathcal{HIBE},\mathcal{A}}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{hibe-ano-cpa-0}[L,d]}_{\mathcal{HIBE},\mathcal{A}}(k) = 1\right]$$

The scheme $\mathcal{HIBE}$ is said to be HIBE-IND-CPA$[d]$-*secure* (resp. HIBE-ANO-CPA$[L, d]$-*secure*) if the respective advantage function is negligible for all PTAs $\mathcal{A}$.

5.2 A sufficient condition for anonymity

We further extend Lemma 4.3 to the hierarchical case. To this end, we introduce a new notion HIBE-ANO-RE-CPA$[L, d]$ as follows. Let $\mathcal{HIBE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$ be a HIBE scheme with message space MsgSp, let L be a set of levels, and let d be the maximum hierarchy depth. To an adversary $\mathcal{A}$ and a bit b, we associate the following experiment:

```
Experiment $\mathbf{Exp}^{\text{hibe-ano-re-b}[L,d]}_{\mathcal{HIBE},\mathcal{A}}(k)$
    $IDSet \leftarrow \emptyset$ ; $(pk, msk) \xleftarrow{\$} \mathsf{Setup}(1^k)$
    pick random oracle $H$
    $(id_0, id_1, M, state) \xleftarrow{\$} \mathcal{A}^{\text{KeyDer}(\cdot),H}(\texttt{find}, pk)$
    if $|id_0| \neq |id_1|$ or $|id_0| > d(k)$ or $|id_1| > d(k)$
       or $M \notin \mathsf{MsgSp}(k)$ then return 0
    $R \xleftarrow{\$} \{0,1\}^{|M|}$ ; $C \xleftarrow{\$} \mathsf{Enc}^H(pk, id_b, R)$
    $b' \xleftarrow{\$} \mathcal{A}^{\text{KeyDer}(\cdot),H}(\texttt{guess}, C, state)$
    if $IDSet \cap (\{id_0, id_1\} \cup \text{anc}(id_0) \cup \text{anc}(id_1)) = \emptyset$
       and $\text{diff}(id_0, id_1) \subseteq L(k)$ then return $b'$ else return 0

Oracle KeyDer$(id)$
    if $|id| > d(k)$ then return $\perp$
    $IDSet \leftarrow IDSet \cup \{id\}$
    return $\mathsf{KeyDer}^H(msk, id)$
```

The HIBE-ANO-RE-CPA$[L, d]$-*advantage* of an adversary $\mathcal{A}$ in violating the level-L anonymity of the scheme $\mathcal{HIBE}$ with depth $d(k)$ is defined as

$$\mathbf{Adv}^{\text{hibe-ano-re}[L,d]}_{\mathcal{HIBE},\mathcal{A}}(k) = \Pr\left[\mathbf{Exp}^{\text{hibe-ano-re-1}[L,d]}_{\mathcal{IBE},\mathcal{A}}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{hibe-ano-re-0}[L,d]}_{\mathcal{IBE},\mathcal{A}}(k) = 1\right].$$

A scheme $\mathcal{HIBE}$ is said to be HIBE-ANO-RE-CPA$[L, d]$-*secure* if this advantage is a negligible function in k for all PTAs $\mathcal{A}$. The following lemma follows from a hybrid argument similar to that of Lemma 4.3.

Lemma 5.1 *Let $\mathcal{HIBE}$ be a HIBE scheme that is* HIBE-IND-CPA$[d]$ *and* HIBE-ANO-RE-CPA$[L, d]$-*secure for some set of levels L and hierarchy depth d. Then $\mathcal{HIBE}$ is also* HIBE-ANO-CPA$[L, d]$-*secure.*

5.3 Construction

The HIBE scheme of [17] appears to be anonymous, but supports only two levels of identities, and is only resistant against limited collusions at the second level, and hence is not usable for our constructions that follow. Since the HIBE of [12], here denoted $\mathcal{GS}$-$\mathcal{HIBE}$, is equivalent to the Boneh-Franklin IBE scheme [8] when restricted to the first level, and since the latter is provably anonymous as per Theorem 4.4, one could hope that $\mathcal{GS}$-$\mathcal{HIBE}$ is level-1 anonymous, but this turns out not to be true, and the HIBE of [6] is not level-1 anonymous either. To see why, consider the following. The $\mathcal{GS}$-$\mathcal{HIBE}$ encryption of a message M under identity $id = (id_1, \ldots, id_l)$ is a tuple

$$\left(\, rP,\ rH_1(id|_2),\ \ldots,\ rH_1(id|_l),\ H_2(e(rP, H_1(id_1))) \oplus m \,\right) \tag{7}$$

where H_1, H_2 are random oracles, P is a generator of a pairing group that is part of pk, and r is chosen at random from $\mathbb{Z}_p$ by the encryption algorithm. Anonymity is violated because an adversary can decide whether a given ciphertext (C_1, C_2, C_3) is intended for $id = (id_1, id_2)$ or $id' = (id'_1, id_2)$ by checking whether $e(C_2, P)$ equals $e(C_1, H_1(id))$ or $e(C_1, H_1(id'))$.

```
Setup$(1^k)$
  $(\mathbb{G}_1, \mathbb{G}_2, p, e) \xleftarrow{\$} \mathcal{G}(1^k)$ ; $P \xleftarrow{\$} \mathbb{G}_1^*$
  $s_0 \xleftarrow{\$} \mathbb{Z}_p^*$ ; $S_0 \leftarrow 0$ ; $Q_0 \leftarrow s_0 P$
  $pk \leftarrow (\mathbb{G}_1, \mathbb{G}_2, p, e, P, Q_0)$
  $msk \leftarrow (pk, (), S_0, s_0)$
  return $(pk, msk)$

KeyDer$^{H_{1,1},\ldots,H_{1,l}}(usk, id)$
  parse $id$ as $(id_1, \ldots, id_{l+1})$
  parse $usk$ as $(pk, id|_l, S_l, Q_1, \ldots, Q_{l-1}, s_l)$
  parse $pk$ as $(\mathbb{G}_1, \mathbb{G}_2, p, e, P, Q_0)$
  $S_{l+1} \leftarrow S_l + s_l H_{1,l+1}(id_{l+1})$
  $Q_l \leftarrow s_l P$ ; $s_{l+1} \xleftarrow{\$} \mathbb{Z}_p^*$
  return $(pk, id, S_{l+1}, Q_1, \ldots, Q_l, s_{l+1})$

Enc$^{H_{1,1},\ldots,H_{1,l},H_2}(pk, id, M)$
  parse $pk$ as $(\mathbb{G}_1, \mathbb{G}_2, p, e, P, Q_0)$
  parse $id$ as $(id_1, \ldots, id_l)$
  $r \xleftarrow{\$} \mathbb{Z}_p^*$ ; $C_1 \leftarrow rP$
  for $i = 2, \ldots, l$ do $C_i \leftarrow rH_{1,i}(id_i)$
  $C_{l+1} \leftarrow M \oplus H_2(e(rH_{1,1}(id_1), Q_0))$
  return $(C_1, \ldots, C_{l+1})$

Dec$^{H_2}(usk, C)$
  parse $usk$ as $(pk, id, S_l, Q_1, \ldots, Q_{l-1}, s_l)$
  parse $id$ as $(id_1, \ldots, id_l)$
  parse $pk$ as $(\mathbb{G}_1, \mathbb{G}_2, p, e, P, Q_0)$
  parse $C$ as $(C_1, \ldots, C_{l+1})$
  $\kappa \leftarrow e(S_l, C_1) \cdot \prod_{i=2}^{l} e(Q_{i-1}, C_i)^{-1}$
  return $C_{l+1} \oplus H_2(\kappa)$
```

Figure 6: Algorithms of the $\mathcal{mGS}$-$\mathcal{HIBE}$ scheme $\mathcal{mGS}$-$\mathcal{HIBE}$. $\mathcal{G}$ is a pairing parameter generator and $H_{1,i}$: $\{0,1\}^* \to \mathbb{G}_1^*$ and H_2: $\mathbb{G}_2 \to \{0,1\}^k$ are random oracles.

The lack of anonymity in $\mathcal{GS}$-$\mathcal{HIBE}$ stems from the fact that the hashes in the first l components of the ciphertext depend on the first component of the recipient's identity. In Figure 6, we present a modified scheme that uses a different random oracle $H_{1,l}$ for each level l, and that computes ciphertexts as

$$\big(\, rP,\ rH_{1,2}(id_2),\ \ldots,\ rH_{1,l}(id_l),\ H_2(e(rP, H_{1,1}(id_1))) \oplus m \,\big)\ .$$

The following implies in particular that $\mathcal{mGS}$-$\mathcal{HIBE}$ is the first full HIBE scheme providing any anonymity at all. The restriction on d is inherited from [12].

Theorem 5.2 *For any $d(k) = O(\log(k))$, the $\mathcal{mGS}$-$\mathcal{HIBE}$ scheme is* HIBE-ANO-CPA$[1, d]$*-secure and* HIBE-IND-CPA$[d]$*-secure in the random oracle model assuming the BDH problem is hard relative to the generator $\mathcal{G}$.*

We split up the proof in the following two lemmas. The proof of the first is given in Appendix C, and recycles ideas from [12, 8]. We use Lemma 5.1 to prove the second lemma.

Lemma 5.3 *For any $d(k) = O(\log(k))$, the $\mathcal{mGS}$-$\mathcal{HIBE}$ scheme is* HIBE-IND-CPA$[d]$*-secure in the random oracle model assuming the BDH problem is hard relative to the generator $\mathcal{G}$.*

Lemma 5.4 *For any $d(k) = O(\log(k))$, the $\mathcal{mGS}$-$\mathcal{HIBE}$ scheme is* HIBE-ANO-CPA$[1, d]$*-secure in the random oracle model assuming the BDH problem is hard relative to the generator $\mathcal{G}$.*

Proof: Given Lemmas 5.1 and 5.3, it suffices to show that $\mathcal{mGS}$-$\mathcal{HIBE}$ is HIBE-ANO-RE-CPA$[1, d]$-secure. In the challenge ciphertext $(C_1^*, \ldots, C_{l+1}^*)$, the first component C_1 is chosen uniformly at random from $\mathbb{G}_1^*$. Component C_i^* for $2 \le i \le l$ is uniquely defined by C_1^* and the i-th component of the identity, which is the same for both challenge identities since they can only differ at level 1. Finally, if the message M is chosen uniformly at random from $\{0,1\}^k$, then the last component C_{l+1}^* is also uniformly distributed over $\{0,1\}^k$, independent of $H_2(e(rH_{1,1}(id_1), Q_0))$. Hence, the challenge ciphertext is identically distributed in both worlds, and the advantage of any adversary is 0. ▌

6 Public-key encryption with temporary keyword search

In a PEKS scheme, once the gateway has the trapdoor for a certain period, it can test whether this keyword was present in past ciphertexts, and can test its presence in any future ciphertexts. It may be useful to limit the period in which the trapdoor can be used. Here we propose an extension of PEKS that allows this. We call it public-key encryption with temporary keyword search (PETKS) or temporarily searchable encryption for short. A trapdoor here is created for a time interval $[s, e]$ and will only allow the gateway to test whether ciphertexts created in this time interval contain the keyword.

6.1 Definitions

PETKS SCHEMES. *Public-key encryption with temporary keyword search* (PETKS) is a generalization of PEKS in which a trapdoor can be issued for any desired window of time rather than forever. Formally, the scheme $\mathcal{PETKS} = (\mathsf{KG}, \mathsf{Td}, \mathsf{PETKS}, \mathsf{Test}, N)$ consists of four PTAs and a polynomially bounded function $N\colon \mathbb{N} \to \mathbb{N}$. Via $(pk, sk) \xleftarrow{\$} \mathsf{KG}(1^k)$, the randomized key-generation algorithm produces keys for the receiver; via $C \xleftarrow{\$} \mathsf{PETKS}^H(pk, w, i)$ a sender encrypts a keyword w in time period $i \in [0..N(k)-1]$ to get a ciphertext; via $t_w \xleftarrow{\$} \mathsf{Td}^H(sk, w, s, e)$ the receiver computes a trapdoor t_w for keyword w in period $[s..e]$ where $0 \le s \le e \le N(k) - 1$, and provides it to the gateway; via $b \leftarrow \mathsf{Test}^H(t_w, C)$ the gateway tests whether C encrypts w, where b is a bit with 1 meaning "accept" or "yes" and 0 meaning "reject" or "no". Here H is a random oracle whose domain and/or range might depend on k and pk. We require that for all $k \in \mathbb{N}$, all s, e, i with $0 \le s \le i \le e \le N(k) - 1$, and all $w \in \{0, 1\}^*$,

$$\Pr\left[\mathsf{Test}^H(\mathsf{Td}^H(sk, w, s, e), \mathsf{PETKS}^H(pk, w, i)) = 1\right] = 1 ,$$

where the probability is taken over the choice of $(pk, sk) \xleftarrow{\$} \mathsf{KG}(1^k)$, the random choice of H, and the coins of all the algorithms in the expression above.

CONSISTENCY. As we did for standard PEKS in Section 3.2, we present a consistency definition for PETKS schemes below. Let $\mathcal{PETKS} = (\mathsf{KG}, \mathsf{Td}, \mathsf{PETKS}, \mathsf{Test}, N)$ be a PETKS scheme. We associate to an adversary $\mathcal{U}$ the following experiment:

```
Experiment $\mathbf{Exp}^{\text{petks-consist}}_{\mathcal{PETKS},\mathcal{U}}(k)$
    $(pk, sk) \xleftarrow{\$} \mathsf{KG}(1^k)$ ; pick random oracle $H$
    $(w, w', s, e, i) \xleftarrow{\$} \mathcal{U}^H(pk)$
    if $\{s, i, e\} \not\subseteq [0..N(k) - 1]$ or $s > e$ then return 0
    $C \xleftarrow{\$} \mathsf{PETKS}^H(pk, w, i)$ ; $t_{w'} \xleftarrow{\$} \mathsf{Td}^H(sk, w', s, e)$
    if ($w \ne w'$ or $i \notin [s..e]$) and $\mathsf{Test}^H(t_{w'}, C) = 1$ then return 1 else return 0
```

We define the advantage of $\mathcal{U}$ as

$$\mathbf{Adv}^{\text{petks-consist}}_{\mathcal{PETKS},\mathcal{U}}(k) = \Pr\left[\mathbf{Exp}^{\text{petks-consist}}_{\mathcal{PETKS},\mathcal{U}}(k) = 1\right] .$$

Like in the PEKS case, we say the PETKS scheme is *perfectly consistent* if this advantage is 0 for all (computationally unrestricted) adversaries $\mathcal{U}$, *statistically consistent* if it is negligible for all (computationally unrestricted) adversaries $\mathcal{U}$, and *computationally consistent* if it is negligible for all PTAs $\mathcal{U}$.

Note that this definition also considers it a consistency problem if a trapdoor for interval $[s..e]$ tests positively for a ciphertext created in a time period $i \notin [s..e]$. This type of problems is easily protected against, however, by letting the Td and PETKS algorithms include information about the intended time periods into the trapdoors and ciphertexts, respectively.

PRIVACY. Privacy for a PETKS scheme asks that an adversary should not be able to distinguish between the encryption of two challenge keywords of its choice in a time period $i \in [0..N(k)-1]$ of its choice, even if it is allowed not only to obtain trapdoors for non-challenge keywords issued for any time interval, but also is allowed to obtain trapdoors for *any* keywords (even the challenge ones), issued for time intervals not containing i. More formally, let $\mathcal{PETKS} = (\mathsf{KG}, \mathsf{Td}, \mathsf{PETKS}, \mathsf{Test}, N)$ be a PETKS scheme. To any bit $b \in \{0,1\}$ and any adversary $\mathcal{A}$, we associate the following experiment:

```
Experiment $\mathbf{Exp}^{\text{petks-ind-cpa-b}}_{\mathcal{PETKS},\mathcal{A}}(k)$
    $(pk, sk) \xleftarrow{\$} \mathsf{KG}(1^k)$ ; $Tr[\cdot] \leftarrow \emptyset$ ; pick random oracle $H$
    $(w_0, w_1, i, state) \xleftarrow{\$} \mathcal{A}^{\textsc{Trapd}(\cdot,\cdot,\cdot),H}(\texttt{find}, pk)$
    $C \xleftarrow{\$} \mathsf{PEKS}^H(pk, w_b, i)$
    $b' \xleftarrow{\$} \mathcal{A}^{\textsc{Trapd}(\cdot,\cdot,\cdot),H}(\texttt{guess}, C, state)$
    if $i \notin (Tr[w_0] \cup Tr[w_1])$
        then return $b'$ else return 0

Oracle $\textsc{Trapd}(w, s, e)$
    $Tr[w] \leftarrow Tr[w] \cup [s..e]$
    return $\mathsf{Td}^H(sk, w, s, e)$
```

The PETKS-IND-CPA-*advantage* of an adversary $\mathcal{A}$ in violating the chosen-plaintext indistinguishability of $\mathcal{PETKS}$ is defined as

$$\mathbf{Adv}^{\text{petks-ind-cpa}}_{\mathcal{PETKS},\mathcal{A}}(k) = \Pr\left[\mathbf{Exp}^{\text{petks-ind-cpa-1}}_{\mathcal{PETKS},\mathcal{A}}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{petks-ind-cpa-0}}_{\mathcal{PETKS},\mathcal{A}}(k) = 1\right] .$$

A PETKS scheme $\mathcal{PETKS}$ is said to be PETKS-IND-CPA-*secure* if this advantage is a negligible function in k for all PTAs $\mathcal{A}$.

6.2 Constructions for PETKS schemes

CONSTRUCTIONS WITH LINEAR COMPLEXITY. PETKS is reminiscent of forward-security [3, 9], and, as in these works, there are straightforward solutions with keys or trapdoors of length linear in $N(k)$. One such solution is to use a standard PEKS scheme and generate a different key pair (pk_i, sk_i) for each time period $i \in [0..N(k)-1]$. Let $pk = (pk_0, \ldots, pk_{N(k)-1})$ be the PETKS public key and $sk = (sk_0, \ldots, sk_{N(k)-1})$ be the PETKS secret key. During time period i, the sender encrypts a keyword w by encrypting w under pk_i using the PEKS scheme. The trapdoor for a keyword w in the interval $[s..e]$ consists of all PEKS trapdoors for w of periods $s, \ldots, e$. A somewhat more efficient solution is to let the PETKS master key pair be a single key pair for the standard PEKS scheme, and append the time period to the keyword when encrypting or computing trapdoors. This scheme achieves short public and secret keys, but still has trapdoor length linear in $N(k)$, because the PETKS trapdoor still contains PEKS trapdoors for all time periods $s, \ldots, e$.

THE $\mathsf{hibe\text{-}2\text{-}petks}$ TRANSFORMATION. We now present a transformation $\mathsf{hibe\text{-}2\text{-}petks}$ of a HIBE scheme into a PETKS scheme that yields a PETKS scheme with complexity logarithmic in N for all parameters. The construction is very similar to the generic construction of forward-secure encryption from binary-tree encryption [9]. The number of time periods is $N(k) = 2^{t(k)}$ for some $t(k) = O(\log(k))$. If $i \in [0..N(k)-1]$, then let $i_1 \ldots i_{t(k)}$ denote its binary representation as a $t(k)$-bit string. Intuitively, our construction instantiates a HIBE of depth $t(k)+1$ with keywords as the first level of the identity tree and the time structure on the lower levels. The trapdoor for keyword w and interval of time periods $[s..e]$ consists of the user secret keys of all identities from $(w, s_1, \ldots, s_{t(k)})$ to $(w, e_1, \ldots, e_{t(k)})$, but taking advantage of the hierarchical structure to include entire subtrees of keys.

More precisely, let $\mathcal{HIBE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$ be a HIBE scheme. Then we associate to it a PETKS scheme $\mathcal{PETKS} = \textsf{hibe-2-petks}(\mathcal{HIBE}, t(k)) = (\mathsf{KG}, \mathsf{Td}, \mathsf{PETKS}, \mathsf{Test}, N)$ such that $N(k) = 2^{t(k)}$, $\mathsf{KG}(1^k) = \mathsf{Setup}(1^k)$ and $\mathsf{PETKS}(pk, w, i) = (i, C_1, C_2)$ where $C_1 \stackrel{\$}{\leftarrow} \{0,1\}^k$ and $C_2 \leftarrow \mathsf{Enc}(pk, (w, i_1, \ldots, i_{t(k)}), C_1)$. The trapdoor algorithm $\mathsf{Td}(sk, w, s, e)$ first constructs a set T of identities as follows. Let j be the smallest index so that $s_j \neq e_j$. Then T is the set containing $(w, s_1, \ldots, s_{t(k)})$, $(w, e_1, \ldots, e_{t(k)})$, the right siblings of all nodes on the path from $(w, s_1, \ldots, s_{j+1})$ to $(w, s_1, \ldots, s_{t(k)})$, and the left siblings of all nodes on the path from $(w, e_1, \ldots, e_{j+1})$ to $(w, e_1, \ldots, e_{t(k)})$. If j does not exist, meaning $s = e$, then $T \leftarrow \{(w, s_1, \ldots, s_{t(k)})\}$. The trapdoor t_w is the set of tuples $((w, i_1, \ldots, i_r), \mathsf{KeyDer}(sk, (w, i_1, \ldots, i_r)))$ for all $(i_1, \ldots, i_r) \in T$. To test a ciphertext (i, C_1, C_2), the Test algorithm looks up a tuple $((w, i_1, \ldots, i_r), usk[(w, i_1, \ldots, i_r)])$ in t_w. It returns 0 when no such tuple is found. Otherwise, it derives $usk[(w, i_1, \ldots, i_{t(k)})]$ using repetitive calls to the KeyDer algorithm, and returns 1 iff $\mathsf{Dec}(usk[(w, i_1, \ldots, i_{t(k)})], C_2) = C_1$.

Theorem 6.1 *Let $\mathcal{HIBE}$ be a HIBE scheme, and let $\mathcal{PETKS} = \textsf{hibe-2-petks}(\mathcal{HIBE}, t(k))$ where $t(k) = O(\log(k))$. If $\mathcal{HIBE}$ is* HIBE-ANO-CPA$[1, t+1]$*-secure, then $\mathcal{PETKS}$ is* PETKS-IND-CPA*-secure. Furthermore, if $\mathcal{HIBE}$ is* HIBE-IND-CPA$[t+1]$*-secure, then $\mathcal{PETKS}$ is computationally consistent.*

We split the proof of the theorem over the following two lemmas.

Lemma 6.2 *Let $\mathcal{HIBE}$ be a HIBE scheme, and let $\mathcal{PETKS} = \textsf{hibe-2-petks}(\mathcal{HIBE}, t(k))$ where $t(k) = O(\log(k))$. If $\mathcal{HIBE}$ is* HIBE-ANO-CPA$[1, t+1]$*-secure, then $\mathcal{PETKS}$ is* PETKS-IND-CPA*-secure.*

Proof: Let $\mathcal{HIBE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$ be a level-1 anonymous (HIBE-ANO-CPA$[1, t+1]$-secure) HIBE scheme, and let $\mathcal{PETKS} = \textsf{hibe-2-petks}(\mathcal{HIBE}, t(k)) = (\mathsf{KG}, \mathsf{Td}, \mathsf{PETKS}, \mathsf{Test}, N)$ be the associated PETKS scheme. Given an adversary $\mathcal{A}$ breaking the PETKS-IND-CPA security of $\mathcal{PETKS}$, we construct an adversary $\mathcal{B}$ breaking the HIBE-ANO-CPA$[1, t+1]$ security of $\mathcal{HIBE}$ as follows. On input public parameters pk, $\mathcal{B}$ runs $\mathcal{A}$ on inputs $(\mathtt{find}, pk)$. When $\mathcal{A}$ queries its TRAPD oracle for the trapdoor of keyword w for interval $[s..e]$, then $\mathcal{B}$ constructs a set T exactly as the Td algorithm does, and constructs the corresponding trapdoor by querying its KEYDER oracle for the user secret keys corresponding to all identities in T.

When $\mathcal{A}$ outputs challenge keywords w_0, w_1 and time period i, $\mathcal{B}$ outputs challenge identities $id_0 = (w_0, i_1, \ldots, i_t)$, $id_1 = (w_1, i_1, \ldots, i_t)$ and a randomly chosen message M of length k. Note that identities id_0 and id_1 differ on level 1, but are otherwise equal, as required for level-1 anonymity. Upon receiving challenge ciphertext C, adversary $\mathcal{B}$ sends (i, R, C) to $\mathcal{A}$ and runs it until $\mathcal{A}$ outputs a bit b' (responding to $\mathcal{A}$'s oracle queries the same way as before). Adversary $\mathcal{B}$ outputs the same bit b'.

It is easy to see that, due to the ordered structure of the time tree, adversary $\mathcal{B}$ does not need to corrupt any ancestors of its challenge identities. Therefore, adversary $\mathcal{B}$ succeeds whenever $\mathcal{A}$ does, and we have

$$\mathbf{Adv}^{\text{petks-ind-cpa}}_{\mathcal{PETKS},\mathcal{A}}(k) \leq \mathbf{Adv}^{\text{hibe-ano-cpa}[1,t+1]}_{\mathcal{HIBE},\mathcal{B}}(k)$$

for all $k \in \mathbb{N}$, from which the theorem follows. ▌

Lemma 6.3 *Let $\mathcal{HIBE}$ be a HIBE scheme, and let $\mathcal{PETKS} = \textsf{hibe-2-petks}(\mathcal{HIBE}, t(k))$ where $t(k) = O(\log(k))$. If $\mathcal{HIBE}$ is* IBE-IND-CPA*-secure, then $\mathcal{PETKS}$ is computationally consistent.*

Proof: Let $\mathcal{A}$ be an adversary of the consistency of $\mathcal{PETKS}$. We construct an IBE-IND-CPA adversary $\mathcal{B}$ of $\mathcal{HIBE}$ as follows.

```
Adversary $\mathcal{B}^{\textsc{KeyDer}(\cdot)}(\texttt{find}, pk)$
    $(w, w', s, e, i) \xleftarrow{\$} \mathcal{A}(pk)$
    $R, R' \xleftarrow{\$} \{0,1\}^k$ (where $\{0,1\}^k$ is the message space of $\mathcal{HIBE}$)
    $id \leftarrow (w, i_1, \ldots, i_t)$
    $M_0 \leftarrow R\,;\ M_1 = R'$
    $state \leftarrow (pk, w, w', R, R', s, e, i)$
    return $(id, M_0, M_1, state)$
Adversary $\mathcal{B}^{\textsc{KeyDer}(\cdot)}(\texttt{guess}, C, state)$
    parse $C$ as $(i, R, C')$
    $t_{w'} \xleftarrow{\$} \textsc{KeyDer}((w', i_1, \ldots, i_t))\,;\ X \leftarrow \mathsf{Dec}(t_{w'}, C')$
    if $X = R'$ then return 1 else return 0
```

Since, by construction, $\mathsf{Test}(t_w, C)$ returns 0 whenever $i \notin [s..e]$, we can assume that $w' \neq w$ and $i \in [s..e]$. Then, exactly as in Theorem 4.2, we have

$$\Pr\left[\mathbf{Exp}^{\text{hibe-ind-cpa-1}[t+1]}_{\mathcal{HIBE},\mathcal{B}}(k) = 1\right] \geq \Pr\left[\mathbf{Exp}^{\text{petks-consist}}_{\mathcal{PETKS},\mathcal{A}}(k) = 1\right] \quad (8)$$

$$\Pr\left[\mathbf{Exp}^{\text{hibe-ind-cpa-0}[t+1]}_{\mathcal{HIBE},\mathcal{B}}(k) = 1\right] \leq 2^{-l}. \quad (9)$$

Equation (8) and Equation (9) give us

$$\mathbf{Adv}^{\text{petks-consist}}_{\mathcal{PETKS},\mathcal{A}}(k) \leq \mathbf{Adv}^{\text{hibe-ind-cpa}[t+1]}_{\mathcal{HIBE},\mathcal{B}}(k) + 2^{-l}.$$

The result follows. ▌

Complexity. Since the $m\mathcal{GS}$-$\mathcal{HIBE}$ has user secret keys and ciphertexts of size linear in the depth of the tree, our resulting PETKS scheme has public and secret keys of size $O(1)$, ciphertexts of size $O(\log N(k))$ and trapdoors of size $O(\log^2 N(k))$. We note that in this case a user can decrypt ciphertexts intended for any of its descendants directly, without needing to derive the corresponding secret key first. This makes the call to the KeyDer algorithm in the Test algorithm superfluous, thereby improving the efficiency of Test.

7 Identity-based encryption with keyword search

In this section, we show how to combine the concepts of identity-based encryption and PEKS to obtain identity-based encryption with keyword search (IBEKS) or ID-based searchable encryption for short. Like in IBE schemes, this allows to use any string as a recipient's public key for the PEKS scheme.

7.1 Definitions

IBEKS schemes. An identity-based encryption with keyword search scheme $\mathcal{IBEKS}$ = (Setup, KeyDer, Td, IBEKS, Test) is made up of five algorithms. Via $(pk, msk) \xleftarrow{\$} \mathsf{Setup}(1^k)$, where $k \in \mathbb{N}$ is the security parameter, the randomized setup algorithm produces master keys; via $usk[id] \xleftarrow{\$} \mathsf{KeyDer}^H(msk, id)$, the master computes the secret key for identity id; via $C \xleftarrow{\$} \mathsf{IBEKS}^H(pk, id, w)$, a sender encrypts a keyword w to identity id to get a ciphertext; via $t_w \xleftarrow{\$} \mathsf{Td}^H(usk[id], w)$, the receiver computes a trapdoor t_w for keyword w and identity id and provides it to the gateway; via $b \leftarrow \mathsf{Test}^H(t_w, C)$, the gateway tests whether C encrypts w, where b is a bit with 1 meaning "accept" or "yes" and 0 meaning "reject" or "no". As usual H is a random oracle whose domain and/or range might depend on k and pk. For consistency, we require that for all $k \in \mathbb{N}$, all identities id, and all $w \in \{0,1\}^*$,

$$\Pr\left[\,\mathsf{Test}^H(\mathsf{Td}^H(\mathsf{KeyDer}^H(msk, id), w), \mathsf{IBEKS}^H(pk, id, w)) = 1\,\right] \;=\; 1\;,$$

where the probability is taken over the choice of $(pk, msk) \xleftarrow{\$} \mathsf{Setup}(1^k)$, the random choice of H, and the coins of all algorithms in the expression above.

Consistency. As we have done for other primitives, we now present formal definitions of consistency for IBEKS schemes. Let $\mathcal{IBEKS} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Td}, \mathsf{IBEKS}, \mathsf{Test})$ be an IBEKS scheme. We associate to an adversary $\mathcal{U}$ the following experiment:

```
Experiment $\mathbf{Exp}^{\text{ibeks-consist}}_{\mathcal{IBEKS},\mathcal{U}}(k)$
    $(pk, msk) \xleftarrow{\$} \mathsf{Setup}(1^k)$ ; pick random oracle $H$
    $(w, w', id, id') \xleftarrow{\$} \mathcal{U}^H(pk)$
    $C \xleftarrow{\$} \mathsf{IBEKS}^H(pk, id, w)$ ; $t_{w'} \xleftarrow{\$} \mathsf{Td}^H(\mathsf{KeyDer}(msk, id'), w')$
    if $(w \neq w'$ or $id \neq id')$ and $\mathsf{Test}^H(t_{w'}, C) = 1$ then return 1 else return 0
```

We define the advantage of $\mathcal{U}$ as

$$\mathbf{Adv}^{\text{ibeks-consist}}_{\mathcal{IBEKS},\mathcal{U}}(k) \;=\; \Pr\left[\mathbf{Exp}^{\text{ibeks-consist}}_{\mathcal{IBEKS},\mathcal{U}}(k) = 1\right].$$

Like in the PEKS case, we say that $\mathcal{IBEKS}$ is *perfectly consistent* if this advantage is 0 for all (computationally unrestricted) adversaries $\mathcal{U}$, *statistically consistent* if it is negligible for all (computationally unrestricted) adversaries $\mathcal{U}$, and *computationally consistent* if it is negligible for all PTAs $\mathcal{U}$.

Note that this definition also considers it a consistency problem if a trapdoor for identity id' tests positively for a ciphertext intended for identity $id \neq id'$. This type of problems is easily avoided by having the KeyDer, Td and IBEKS algorithms include the intended identity into the user secret keys, trapdoors and ciphertexts, respectively.

Privacy. Let $\mathcal{IBEKS} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Td}, \mathsf{IBEKS}, \mathsf{Test})$ be an ID-based searchable encryption scheme. To any $b \in \{0,1\}$ and any adversary $\mathcal{A}$, we associate the following experiment:

```
Experiment $\mathbf{Exp}^{\text{ibeks-ind-cpa-b}}_{\mathcal{IBEKS},\mathcal{A}}(k)$
    $WSet \leftarrow \emptyset$ ; $IDSet \leftarrow \emptyset$
    $(pk, msk) \leftarrow \mathsf{Setup}(1^k)$ ; pick random oracle $H$
    $(id, w_0, w_1, state) \leftarrow \mathcal{A}^{\text{KeyDer}(\cdot),\text{Trapd}(\cdot,\cdot),H}(\texttt{find}, pk)$
    $C \leftarrow \mathsf{IBEKS}^H(pk, id, w_b)$
    $b' \leftarrow \mathcal{A}^{\text{KeyDer}(\cdot),\text{Trapd}(\cdot,\cdot),H}(\texttt{guess}, C, state)$
    if $id \notin IDSet$ and $WSet \cap \{(id, w_0), (id, w_1)\} = \emptyset$
      then return $b'$ else return 0

Oracle KeyDer$(id)$
    $IDSet \leftarrow IDSet \cup \{id\}$
    return $\mathsf{KeyDer}^H(msk, id)$

Oracle Trapd$(id, w)$
    $WSet \leftarrow WSet \cup \{(id, w)\}$
    return $\mathsf{Td}^H(\mathsf{KeyDer}^H(msk, id), w)$
```

The IBEKS-IND-CPA-*advantage* of an adversary $\mathcal{A}$ in violating the chosen-plaintext indistinguishability of the scheme $\mathcal{IBEKS}$ is defined as

$$\mathbf{Adv}^{\text{ibeks-ind-cpa}}_{\mathcal{IBEKS},\mathcal{A}}(k) = \Pr\left[\mathbf{Exp}^{\text{ibeks-ind-cpa-1}}_{\mathcal{IBEKS},\mathcal{A}}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{ibeks-ind-cpa-0}}_{\mathcal{IBEKS},\mathcal{A}}(k) = 1\right].$$

An IBEKS scheme $\mathcal{IBEKS}$ is said to be IBEKS-IND-CPA-*secure* if this advantage is a negligible function in k for all PTAs $\mathcal{A}$.

7.2 A generic transformation from anonymous HIBE schemes

We now propose a generic transformation, called hibe-2-ibeks, to convert any HIBE scheme with two levels into an IBEKS scheme. To obtain an IBEKS that is IBEKS-IND-CPA-secure, it is sufficient to start with a HIBE that is anonymous at level 2. Moreover, if the underlying HIBE is HIBE-IND-CPA[2]-secure, then the resulting IBEKS is also computationally consistent.

THE hibe-2-ibeks TRANSFORMATION. Given a HIBE scheme $\mathcal{HIBE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$ with two levels, hibe-2-ibeks returns the IBEKS scheme $\mathcal{IBEKS} = (\mathsf{Setup}, \overline{\mathsf{KeyDer}}, \mathsf{IBEKS}, \mathsf{Td}, \mathsf{Test})$ such that $\overline{\mathsf{KeyDer}}(msk, id) = (usk, id)$ where $usk \xleftarrow{\$} \mathsf{KeyDer}(msk, id)$, $\mathsf{IBEKS}(pk, id, w) = (id, C_1, C_2)$ where $C_1 \xleftarrow{\$} \{0,1\}^k$ and $C_2 = \mathsf{Enc}(pk, (id, w), C_1)$, $\mathsf{Td}(\overline{usk} = (usk, id), w) = (id, t_w)$ where $t_w \xleftarrow{\$} \mathsf{KeyDer}(usk, (id, w))$ and $\mathsf{Test}(\overline{t_w} = (id, t_w), (id', C_1, C_2))$ returns 1 iff $\mathsf{Dec}(t_w, C_2) = C_1$ and $id = id'$.

Theorem 7.1 *Let $\mathcal{HIBE}$ be a HIBE scheme and let $\mathcal{IBEKS}$ = hibe-2-ibeks($\mathcal{HIBE}$). If $\mathcal{HIBE}$ is* HIBE-IND-CPA[2]*-secure, then $\mathcal{IBEKS}$ is computationally consistent. Furthermore, if $\mathcal{HIBE}$ is* HIBE-ANO-CPA[2, 2]*-secure, then $\mathcal{IBEKS}$ is* IBEKS-IND-CPA*-secure.*

The proof of Theorem 7.1 follows from Lemma 7.2 and Lemma 7.3.

Lemma 7.2 *Let $\mathcal{HIBE}$ be a HIBE scheme and let $\mathcal{IBEKS}$ = hibe-2-ibeks($\mathcal{HIBE}$). If $\mathcal{HIBE}$ is* HIBE-ANO-CPA[2, 2]*-secure, then $\mathcal{IBEKS}$ is* IBEKS-IND-CPA*-secure.*

Proof: Given an adversary $\mathcal{A}$ breaking the IBEKS-IND-CPA-security of $\mathcal{IBEKS}$, we construct an HIBE-ANO-CPA[2, 2]-adversary $\mathcal{B}$ breaking $\mathcal{HIBE}$ as follows. On input a public key pk, algorithm $\mathcal{B}$ runs $\mathcal{A}$ on the same input, answering $\mathcal{A}$'s $\textsc{Trapd}(id, w)$ oracle queries by querying its own $\textsc{KeyDer}(\cdot)$ oracle for the secret key corresponding to identity (id, w). When $\mathcal{A}$ outputs a challenge identity $id^\star$ and two challenge keywords $w_0^\star, w_1^\star$, adversary $\mathcal{B}$ chooses a random message $M^\star \in \{0,1\}^k$ and outputs $M^\star$ as the challenge message and $id_0^\star = (id^\star, w_0^\star)$ and $id_1^\star = (id^\star, w_1^\star)$ as the challenge identities, which in fact differ only in the second entry. Let $C^\star$ be the challenge ciphertext that $\mathcal{B}$ receives at the beginning of its guess phase. Adversary $\mathcal{B}$ returns $(M^\star, C^\star)$ to $\mathcal{A}$, and continues to run $\mathcal{A}$ (answering $\textsc{Trapd}$ queries the same way as before) until it outputs a bit b'. Algorithm $\mathcal{B}$ then outputs the same bit b' as its own output.

It is clear from the construction that $\mathcal{B}$'s simulation of $\mathcal{A}$'s environment is perfect. Since $\mathcal{A}$ cannot query its $\textsc{Trapd}$ oracle on keywords $(id^\star, w_0^\star)$ and $(id^\star, w_1^\star)$, $\mathcal{B}$ will not be forced to query its $\textsc{KeyDer}$ on identities $id_0^\star$ and $id_1^\star$, and hence wins the game whenever $\mathcal{A}$ does. Therefore, we have that

$$\mathbf{Adv}^{\text{ibeks-ind-cpa}}_{\mathcal{IBEKS},\mathcal{A}}(k) \leq \mathbf{Adv}^{\text{hibe-ano-cpa}[2,2]}_{\mathcal{HIBE},\mathcal{B}}(k) ,$$

from which the theorem follows for CPA security. This proves the lemma. ▌

Lemma 7.3 *Let $\mathcal{HIBE}$ be a HIBE scheme and let $\mathcal{IBEKS}$ = hibe-2-ibeks($\mathcal{HIBE}$). If $\mathcal{HIBE}$ is* HIBE-IND-CPA[2]*-secure, then $\mathcal{IBEKS}$ is computationally consistent.*

Proof: Let $\mathcal{A}_1$ be an adversary of the consistency of $\mathcal{IBEKS}$. We construct an HIBE-IND-CPA[2] adversary $\mathcal{B}_1$ of $\mathcal{HIBE}$ as follows.

```
Adversary $\mathcal{B}_1^{\textsc{KeyDer}(\cdot)}(\text{find}, pk)$
    $(w, w', id, id') \xleftarrow{\$} \mathcal{A}_1(pk)$ ; $R, R' \xleftarrow{\$} \{0,1\}^l$ (where $\{0,1\}^l$ is the message space of $\mathcal{HIBE}$)
    $id = (id', w)$
    $w_0 = R\ ;\ w_1 = R'$
    $state = (pk, w, w', R, R')$
    return $(id, w_0, w_1, state)$

Adversary $\mathcal{B}_1^{\text{KeyDer}(\cdot)}(\text{guess}, C, state)$
    $t_{w'} \overset{\$}{\leftarrow} \text{KeyDer}((id', w'))\ ;\ X \leftarrow \mathsf{Dec}(t_{w'}, C)$
    if $X = R'$ then return 1 else return 0
```

Since, by construction, $\mathsf{Test}(t_w, C)$ returns 0 whenever $id \neq id'$, we can assume that $w' \neq w$ and $id' = id$. Thus, exactly as in Theorem 4.2, we have

$$\Pr\left[\mathbf{Exp}^{\text{hibe-ind-cpa-1[2]}}_{\mathcal{HIBE},\mathcal{B}_1}(k) = 1\right] \geq \Pr\left[\mathbf{Exp}^{\text{ibeks-consist}}_{\mathcal{IBEKS},\mathcal{A}}(k) = 1\right] \tag{10}$$

$$\Pr\left[\mathbf{Exp}^{\text{hibe-ind-cpa-0[2]}}_{\mathcal{HIBE},\mathcal{B}_1}(k) = 1\right] \leq 2^{-l}. \tag{11}$$

Equation (10) and Equation (11) give us

$$\mathbf{Adv}^{\text{ibeks-consist}}_{\mathcal{IBEKS},\mathcal{A}_1}(k) \leq \mathbf{Adv}^{\text{hibe-ind-cpa[2]}}_{\mathcal{HIBE},\mathcal{B}_1}(k) + 2^{-l}.$$

The result follows. ▌

7.3 Concrete instantiations

Neither the $\mathcal{GS}$-$\mathcal{HIBE}$ scheme of [12] nor the $m\mathcal{GS}$-$\mathcal{HIBE}$ scheme of Figure 6 are anonymous at the second level. For the $\mathcal{GS}$-$\mathcal{HIBE}$ scheme, consider an adversary $\mathcal{A}$ who outputs challenge identities $id = (id_1, id_2)$ and $id' = (id_1, id'_2)$ for any $id_1, id_2, id'_2 \in \{0,1\}^*$ such that $id_2 \neq id'_2$, and any challenge message $M \in \{0,1\}^k$. When given the challenge ciphertext $C = (C_1, C_2, C_3)$, $\mathcal{A}$ checks whether $e(C_1, H_1(id)) = e(P, C_2)$. (See Equation (7) for how ciphertexts are created in the $\mathcal{GS}$-$\mathcal{HIBE}$ scheme.) If the test succeeds, then $\mathcal{A}$ returns 0, otherwise it returns 1. It is easy to see that the advantage of $\mathcal{A}$ is $\mathbf{Adv}^{\text{hibe-ano-cpa[2,2]}}_{\mathcal{GS}\text{-}\mathcal{HIBE},\mathcal{A}}(k) \geq 1 - 2^{-k}$. A similar attack can be mounted on the $m\mathcal{GS}$-$\mathcal{HIBE}$ scheme by checking whether $e(C_1, H_{1,2}(id_2)) = e(P, C_2)$.

In Appendix B.3, we show that the recently introduced HIBE scheme by Boneh et al. [6] is not level-2 anonymous either (and actually, not anonymous at any level). So even though we showed how to construct IBEKS schemes from any level-2 anonymous HIBE scheme, we do not know of any schemes to instantiate it with, even in the random oracle model.

7.4 Identity-based encryption with temporary keyword search

The ideas of Sections 6 and 7 can be further combined to create an identity-based encryption scheme with temporary keyword search. This can be constructed from a level-2 anonymous HIBE scheme by putting the users' identities at the first level of the hierarchy, the keywords at the second, and a binary tree of time frames on the levels below.

8 Open problems

Existing constructions of (IBE-IND-CPA and) IBE-ANO-CPA IBE schemes (cf. Theorem 4.4), as well as of PEKS-IND-CPA and computationally consistent PEKS schemes [7], are in the RO model. It would be interesting to find a construction of an (IBE-IND-CPA and) IBE-ANO-CPA IBE scheme with a proof of security in the standard model. (Theorem 4.2 then implies a PEKS-IND-CPA and computationally consistent PEKS scheme in the standard model.)

Another interesting question is to find a HIBE scheme providing anonymity at level 2, even in the RO model. By our results, such a scheme would automatically yield an IBEKS scheme.

It would be nice to come up with a searchable encryption scheme that allows for more advanced searching tools. A natural extension of the basic scheme may be, for example, a searchable encryption mechanism that allow to search for simple boolean formulas on keywords (say $w_1 \wedge w_2 \vee w_3$). First steps in this direction have been taken [15, 18] by schemes that allow to search for conjunctive combinations of keywords.

Acknowledgements

We thank Nigel Smart for suggesting the concept of temporarily searchable encryption. Second and tenth authors were supported in part by NSF grants ANR-0129617 and CCR-0208842 and by an IBM Faculty Partnership Development Award. Fourth author was supported by a DAAD postdoc fellowship. Fifth author was supported by an IBM Ph.D Fellowship. Eighth author was supported in part by the Flemish Government under GOA Mefisto 2006/06 and Ambiorix 2005/11, and by the European Commission through the IST Project PRIME. The rest of the authors were supported in part by the European Commission through the IST Program under Contract IST-2002-507932 ECRYPT.

References

[1] Miklós Ajtai and Cynthia Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *29th Annual ACM Symposium on Theory of Computing*, pages 284–293, El Paso, Texas, USA, May 4–6, 1997. ACM Press. (Cited on page 1.)

[2] Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. Key-privacy in public-key encryption. In Colin Boyd, editor, *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 566–582, Gold Coast, Australia, December 9–13, 2001. Springer-Verlag, Berlin, Germany. (Cited on page 2, 14, 18, 33.)

[3] Mihir Bellare and Sara K. Miner. A forward-secure digital signature scheme. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 431–448, Santa Barbara, CA, USA, August 15–19, 1999. Springer-Verlag, Berlin, Germany. (Cited on page 24.)

[4] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 62–73, Fairfax, Virginia, USA, November 3–5, 1993. ACM Press. (Cited on page 4.)

[5] Dan Boneh and Xavier Boyen. Secure identity based encryption without random oracles. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 443–459, Santa Barbara, CA, USA, August 15–19, 2004. Springer-Verlag, Berlin, Germany. (Cited on page 37.)

[6] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume

3494 of *Lecture Notes in Computer Science*, pages 440–456, Aarhus, Denmark, May 22–26, 2005. Springer-Verlag, Berlin, Germany. (Cited on page 3, 19, 21, 29, 37.)

[7] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 506–522, Interlaken, Switzerland, May 2–6, 2004. Springer-Verlag, Berlin, Germany. (Cited on page i, 1, 2, 3, 4, 5, 7, 8, 9, 15, 29, 32, 33, 34.)

[8] Dan Boneh and Matthew K. Franklin. Identity based encryption from the Weil pairing. *SIAM Journal on Computing*, 32(3):586–615, 2003. (Cited on page 2, 3, 5, 9, 14, 19, 21, 22.)

[9] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 255–271, Warsaw, Poland, May 4–8, 2003. Springer-Verlag, Berlin, Germany. (Cited on page 24.)

[10] Ding-Zhu Du and Frank K Hwang. *Combinatorial group testing and its Applications.* World Scientific, 1993. (Cited on page 34.)

[11] Cynthia Dwork, Moni Naor, and Omer Reingold. Immunizing encryption schemes from decryption errors. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 342–360, Interlaken, Switzerland, May 2–6, 2004. Springer-Verlag, Berlin, Germany. (Cited on page 1.)

[12] Craig Gentry and Alice Silverberg. Hierarchical ID-based cryptography. In Yuliang Zheng, editor, *Advances in Cryptology – ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 548–566, Queenstown, New Zealand, December 1–5, 2002. Springer-Verlag, Berlin, Germany. (Cited on page 3, 19, 21, 22, 29.)

[13] Eu-Jin Goh. Secure indexes. Cryptology ePrint Archive, Report 2003/216, 2003. `http://eprint.iacr.org/2003/216/`. (Cited on page 1, 4.)

[14] Oded Goldreich. *Foundations of Cryptography: Basic Applications*, volume 2. Cambridge University Press, Cambridge, UK, 2004. (Cited on page 1, 6.)

[15] Philippe Golle, Jessica Staddon, and Brent R. Waters. Secure conjunctive keyword search over encrypted data. In Markus Jakobsson, Moti Yung, and Jianying Zhou, editors, *ACNS 04: 2nd International Conference on Applied Cryptography and Network Security*, volume 3089 of *Lecture Notes in Computer Science*, pages 31–45, Yellow Mountain, China, June 8–11, 2004. Springer-Verlag, Berlin, Germany. (Cited on page 1, 4, 30.)

[16] Shai Halevi. A sufficient condition for key-privacy. Cryptology ePrint Archive, Report 2005/005, 2005. `http://eprint.iacr.org/`. (Cited on page 3, 18.)

[17] Jeremy Horwitz and Ben Lynn. Toward hierarchical identity-based encryption. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 466–481, Amsterdam, The Netherlands, April 28 – May 2, 2002. Springer-Verlag, Berlin, Germany. (Cited on page 3, 19, 21.)

[18] Dong Jin Park, Kihyun Kim, and Pil Joong Lee. Public key encryption with conjunctive field keyword search. In Chae Hoon Lim and Moti Yung, editors, *WISA 04: 5th International Workshop on Information Security Applications*, volume 3325 of *Lecture Notes in Computer Science*, pages 73–86, Jeju Island, Korea, August 23–25, 2004. Springer-Verlag, Berlin, Germany. (Cited on page 30.)

[19] Adi Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology – CRYPTO'84*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53, Santa Barbara, CA, USA, August 19–23, 1985. Springer-Verlag, Berlin, Germany. (Cited on page 14.)

[20] Dawn Xiaodong Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In *2000 IEEE Symposium on Security and Privacy*, pages 44–55, Oakland, CA, May 2000. IEEE Computer Society Press. (Cited on page 1, 4.)

[21] Brent R. Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 114–127, Aarhus, Denmark, May 22–26, 2005. Springer-Verlag, Berlin, Germany. (Cited on page 3, 37.)

[22] Brent R. Waters, Dirk Balfanz, Glenn Durfee, and Diana K. Smetters. Building an encrypted and searchable audit log. In *ISOC Network and Distributed System Security Symposium – NDSS 2004*, San Diego, California, USA, February 4–6, 2004. The Internet Society. (Cited on page 1, 4.)

A Consistency of the limited PEKS schemes

The limited constructions of [7] do not use ROs, but either allow only a polynomial number of keywords to be used, or have key sizes polynomial in the number of trapdoors issued. To treat these types of constructions, we will first extend the definitions of PEKS schemes and their consistency to allow keyword spaces.

PEKS, MORE GENERALLY. A map KwSp is called a *keyword space* if given $k \in \mathbb{N}$ it returns a subset of $\{0,1\}^*$ and also there is a PTA that on input $1^k, w$ returns 1 if $w \in \mathsf{KwSp}(k)$ and 0 otherwise. The space has size $s\colon \mathbb{N} \to \mathbb{N}$ if $|\mathsf{KwSp}(k)| \le s(k)$ for all $k \in \mathbb{N}$. To say that a PEKS scheme has keyword space KwSp means that keywords encrypted by a sender, or for which a receiver generates a trapdoor, are in $\mathsf{KwSp}(k)$ when the keys are generated with security parameter k. Formally, it just means that the right keyword consistency condition of Section 2 is only required for all $w \in \mathsf{KwSp}(k)$ rather than all $w \in \{0,1\}^*$. Note that a PEKS scheme (as previously defined) is just one where $\mathsf{KwSp}(k) = \{0,1\}^*$ for all $k \in \mathbb{N}$. The notion of privacy recalled in Section 2 is generalized in a natural way. Namely, the last line of the experiment becomes

if $\{w_0, w_1\} \cap WSet = \emptyset$ and $w, w' \in \mathsf{KwSp}(k)$ then return b' else return 0.

Our consistency definition from Section 3.2 is similarly generalized. Namely, the last line of the experiment becomes

if $w \neq w'$ and $w, w' \in \mathsf{KwSp}(k)$ and $\mathsf{Test}^H(t_{w'}, C) = 1$ then return 1 else return 0.

With these changes, we can continue to use the same advantage notations and security terminoligies we had before.

In the following, KwSp denotes a keyword space of size s, and for each $k \in \mathbb{N}$ we let $w_k[1], \ldots, w_k[s(k)]$ denote the members of $\mathsf{KwSp}(k)$ in lexicographic order. For $w \in \mathsf{KwSp}(k)$ we let $\mathrm{Ind}_k(w)$ be

```
$\mathsf{KG}(1^k)$
  for $i = 1, \ldots, s(k)$ do $(pk[i], sk[i]) \xleftarrow{\$} \mathsf{EKG}(1^k)$
  return $(pk, sk)$

$\mathsf{PEKS}(pk, w)$
  $i \leftarrow \mathrm{Ind}_k(w)$ ; $M \xleftarrow{\$} \{0,1\}^k$
  $c \xleftarrow{\$} \mathsf{Enc}(pk[i], M)$ ; $C \leftarrow (c, M)$
  return $C$

$\mathsf{Td}(sk, w)$
  $i \leftarrow \mathrm{Ind}_k(w)$ ; $t_w \leftarrow sk[i]$
  return $t_w$

$\mathsf{Test}(t_w, C)$
  parse $C$ as $(c, M)$
  $M' \leftarrow \mathsf{Dec}(t_w, c)$
  if $M = M'$ then return 1 else return 0
```

Figure 7: Algorithms of the first limited PEKS scheme associated to a polynomial-sized KwSp.

the unique i such that $w = w_k[i]$. We say that KwSp is *accessible* if $\mathrm{Ind}_k(w)$ can be computed in time $\mathrm{poly}(k, |w|)$.

A.1 The first limited scheme

Let KwSp be an accessible keyword space whose size s is a polynomial (in k). Let $\mathcal{PKE} = (\mathsf{EKG}, \mathsf{Enc}, \mathsf{Dec})$ be a standard public-key encryption scheme, specified as usual by its key-generation, encryption and decryption algorithms. The first limited construction of [7] associates to KwSp and $\mathcal{PKE}$ the PEKS scheme $\mathcal{PEKS} = (\mathsf{KG}, \mathsf{PEKS}, \mathsf{Td}, \mathsf{Test})$ with keyword space KwSp shown in Figure 7. Note that the fact that the algorithms of this scheme are PT relies on the assumed accessibility of the scheme. Also note that the public and secret keys have size linear in $s(k)$, which is the limitation of the scheme.

The PEKS-IND-CPA security of the scheme is proven in [7] based on the anonymity of $\mathcal{PKE}$ as defined in [2]. Our concern here is the consistency. The interesting element of the following result is that based on only a computational assumption about the encryption scheme (namely that it meets the standard IND-CPA notion of privacy) we can establish *statistical* consistency of the PEKS scheme:

Theorem A.1 *Let* KwSp *be an accessible keyword space whose size* s *is a polynomial (in* k*) and let* $\mathcal{PKE} = (\mathsf{EKG}, \mathsf{Enc}, \mathsf{Dec})$ *be a standard public-key encryption scheme. Let* $\mathcal{PEKS} = (\mathsf{KG}, \mathsf{PEKS}, \mathsf{Td}, \mathsf{Test})$ *be the associated PEKS scheme with keyword space* KwSp *shown in Figure 7. If* $\mathcal{PKE}$ *is* IND-CPA *secure then* $\mathcal{PEKS}$ *is statistically consistent.* ▌

We remark that the proof shows something stronger, namely that the assumption on $\mathcal{PKE}$ can be relaxed to ask that it is merely one-way.

Proof of Theorem A.1: Let $\mathcal{U}$ be any (computationally unrestricted) adversary attacking the consistency of $\mathcal{PEKS}$. We define the PTA $\mathcal{A}$ attacking the IND-CPA security of $\mathcal{PKE}$ as follows. In its `find` stage, given input a public key $\overline{pk}$, adversary $\mathcal{A}$ lets M_0, M_1 be random k-bit strings, and returns these as its challenge messages. In the `guess` stage, given challenge ciphertext $\overline{C}$ that encrypts M_b under $\overline{pk}$, where $b \in \{0,1\}$ is the challenge bit, it picks random, distinct indices $l, m \in \{1, \ldots, s(k)\}$. It sets $pk[l] \leftarrow \overline{pk}$ and for $i \in \{1, \ldots, l-1, l+1, \ldots, s(k)\}$ it lets $(pk[i], sk[i]) \xleftarrow{\$} \mathsf{EKG}(1^k)$. (We assume k can be recovered from $\overline{pk}$. Think of pk as a public key for the PEKS scheme given as input to $\mathcal{U}$, and $w_k[l], w_k[m]$ as guesses for the keywords output by $\mathcal{U}$ in its consistency experiment.) It lets $M' \leftarrow \mathsf{Dec}(sk[m], \overline{C})$. If $M' = M_1$ then it returns 1, else it returns 0. Now we claim that

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{PKE},\mathcal{A}}(k) \geq \frac{\mathbf{Adv}^{\text{peks-consist}}_{\mathcal{PEKS},\mathcal{U}}(k)}{s(k)^2} - 2^{-k} \; . \tag{12}$$

Note we claim this is true even though $\mathcal{A}$ never ran or invoked $\mathcal{U}$ in any way. (Indeed, it cannot, since the latter may not be PT.) The assumption that $\mathcal{PKE}$ is IND-CPA now implies that $\mathbf{Adv}^{\text{peks-consist}}_{\mathcal{PEKS},\mathcal{U}}(k)$ is negligible. It remains to see why Equation (12) is true. Imagine (a thought experiment) running $\mathcal{U}$ on input pk. Note the distribution of the latter is as given by $\mathsf{KG}(1^k)$. Let w, w' denote the (wlog distinct) keywords $\mathcal{U}$ outputs. Note the distribution of pk is independent of l, m, so $\mathcal{U}$ has no information about l, m and we can think of these as being chosen after w, w' are output. Then the probability that $w = w_k[l]$ and $w' = w_k[m]$ is at least $1/s(k)^2$. So if the challenge bit b equals 1 (i.e. $\overline{C}$ encrypts M_1) then the probability that $\mathcal{A}$ returns 1 (i.e. that $M' = M_1$) is at least $\mathbf{Adv}^{\text{peks-consist}}_{\mathcal{PEKS},\mathcal{U}}(k)/s(k)^2$. On the other hand if $b = 0$ (i.e. $\overline{C}$ encrypts M_0) then the probability that $\mathcal{A}$ returns 1 (i.e. that $M' = M_1$) is at most 2^{-k} because in this case M' is distributed independently of the randomly chosen M_1. ▌

A.2 The second limited scheme

The drawback of the first construction is that the size of the public and secret key grows linearly with the number of keywords $s(k)$. In the second construction of [7], key sizes grow logarithmically in $s(k)$, but security is only provided if the number of trapdoor released to an adversary does not exceed the square-root of the key size. Thus, the scheme is still limited.

We start by recalling cover-free families. If S, T are sets, we say that S does not cover T if $T \not\subseteq S$. Let d, t, s be positive integers, and let $F = (F_i)_{1 \le i \le s}$ be a family of subsets of $\{1, \ldots, d\}$. We say that family F is *t-cover-free over* $\{1, \ldots, d\}$, if for each subset $F_i \in F$ and each S that is the union of at most t sets in $(F_1, \ldots, F_{i-1}, F_i, \ldots, F_s)$, it is the case that S does not cover F_i. Furthermore, we say that F is q-uniform if all subsets in the family have size q. We use the following fact [10]: there is a deterministic PTA that on input integers s, t returns q, d, F where $F = (F_i)_{1 \le i \le s}$ is a q-uniform t-cover-free family over $\{1, \ldots, d\}$, for $q = \lceil d/4t \rceil$ and $d \le 16t^2(1 + \log(s/2)/\log 3)$. In the following we let SUB denote the resulting deterministic PTA that on input s, t, i returns F_i.

Let $t = t(k)$ be a polynomial representing an upper bound on the number of released trapdoors in the PEKS privacy experiment. Let KwSp be an accessible keyword space with $s(k) \le 2^{\text{poly}(k)}$. Let $d(k) = \lfloor 16t^2(k)(1 + \log(s(k)/2)/\log 3) \rfloor$ and $q(k) = \lceil d(k)/4t(k) \rceil$. We call $S_w = \mathsf{SUB}(s(k), t(k), \mathrm{Ind}_k(w))$ the subset associated to keyword w. Let $\mathcal{PKE} = (\mathsf{EKG}, \mathsf{Enc}, \mathsf{Dec})$ be a standard public-key encryption scheme. The second limited construction of [7] associates to KwSp, t and $\mathcal{PKE}$ the PEKS scheme $\mathcal{PEKS} = (\mathsf{KG}, \mathsf{PEKS}, \mathsf{Td}, \mathsf{Test})$ shown in Figure 8. Note that the fact that the algorithms of this scheme are PT relies on the assumed accessibility of the scheme. Also note that the public and secret keys have size logarithmic in the keyword space size $s(k)$ and polynomial (quadratic) in the maximal number of established trapdoors $t(k)$, which is the limitation of the scheme. Also note that the upper bound $t(k)$ must be known in advance as part of the input of the construction.

As long as the number of released trapdoors is upper bounded by $t(k)$, the PEKS-IND-CPA security of the scheme is proven in [7] based on the anonymity of $\mathcal{PKE}$. Again our concern here is the conistency.

Theorem A.2 *Let* KwSp *be an accessible keyword space of size s, let t be a polynomial, and let* $\mathcal{PKE} = (\mathsf{EKG}, \mathsf{Enc}, \mathsf{Dec})$ *be a standard public-key encryption scheme. Let* $\mathcal{PEKS} = (\mathsf{KG}, \mathsf{PEKS}, \mathsf{Td}, \mathsf{Test})$ *be the associated PEKS scheme shown in Figure 8. If* $\mathcal{PKE}$ *is* IND-CPA *secure, then* $\mathcal{PEKS}$ *is computationally consistent.* ▌

Again we remark that the assumption on $\mathcal{PKE}$ can be relaxed to ask that it is merely one-way.

Proof of Theorem A.2: Let t, q, d be as above. Let $\mathcal{U}$ be any PTA attacking the consistency of $\mathcal{PEKS}$. We define the PTA $\mathcal{A}$ attacking the IND-CPA security of $\mathcal{PKE}$ as follows. In its `find` stage,

```
$\mathsf{KG}(1^k)$
    for $i = 1, \ldots, d(k)$ do $(pk[i], sk[i]) \xleftarrow{\$} \mathsf{EKG}(1^k)$.
    return $(pk, sk)$

$\mathsf{PEKS}(pk, w)$
    $i \leftarrow \mathrm{Ind}_k(w)$ ; $S_w \leftarrow \mathsf{SUB}(s(k), t(k), i)$
    parse $S_w$ as $S_w = \{s_1, \ldots, s_{q(k)}\}$
    $M_1, \ldots, M_{q(k)} \xleftarrow{\$} \{0,1\}^k$ ; $M \leftarrow M_1 \oplus \ldots \oplus M_{q(k)}$
    for $j = 1, \ldots, q(k)$ do $C_j \xleftarrow{\$} \mathsf{Enc}(pk[s_j], M_j)$
    $C \leftarrow (M, C_1, \ldots, C_{q(k)})$
    return $C$

$\mathsf{Td}(sk, w)$
    $i \leftarrow \mathrm{Ind}_k(w)$ ; $S_w \leftarrow \mathsf{SUB}(s(k), t(k), i)$.
    parse $S_w$ as $S_w = \{s_1, \ldots, s_{q(k)}\}$
    $t_w \leftarrow (sk[s_1], \ldots, sk[s_{q(k)}])$
    return $t_w$

$\mathsf{Test}(t_w, C)$
    parse $t_w$ as $(sk_1, \ldots, sk_{q(k)})$
    parse $C$ as $(M, C_1, \ldots, C_{q(k)})$
    for $j = 1, \ldots, q(k)$ do $M'_j \leftarrow \mathsf{Dec}(sk_j, C_j)$
    $M' \leftarrow M'_1 \oplus \ldots \oplus M'_{q(k)}$
    if $M = M'$ then return 1 else return 0
```

Figure 8: Algorithms of the second limited PEKS scheme associated to keyword space KwSp (of size s) and parameter t. The functions q, d are defined in the text.

given input a public key $\overline{pk}$, adversary $\mathcal{A}$ lets M_0, M_1 be random k-bit strings, and returns these as its challenge messages. In the `guess` stage, given challenge ciphertext $\overline{C}$ that encrypts M_b under $\overline{pk}$, where $b \in \{0,1\}$ is the challenge bit, it picks a random index $l \in \{1, \ldots, d(k)\}$. It sets $pk[l] \leftarrow \overline{pk}$ and for $i \in \{1, \ldots, l-1, l+1, \ldots, d(k)\}$ it lets $(pk[i], sk[i]) \xleftarrow{\$} \mathsf{EKG}(1^k)$. It runs user $\mathcal{U}$ on input pk. The latter outputs two (wlog distinct) keywords w, w'. Let $S_w = \{s_1, \ldots, s_q\}$ and $S_{w'} = \{s'_1, \ldots, s'_q\}$ be the subsets associated with w and w', respectively. If $l \in S_w$ but $l \notin S_{w'}$ then $\mathcal{A}$ continues, else it aborts.

If $\mathcal{A}$ does not abort it lets m be such that $s_m = l$. It sets $C_m \leftarrow \overline{C}$ and for $j \in \{1, \ldots, m-1, m+1, \ldots, q(k)\}$ it lets $R_j \xleftarrow{\$} \{0,1\}^k$ and $C_j \xleftarrow{\$} \mathsf{Enc}(pk[s_j], R_j)$. For $j \in \{1, \ldots, q(k)\}$ it lets $R'_j \leftarrow \mathsf{Dec}(sk[s'_j], C_j)$. (Note that $\mathcal{A}$ holds all the secret keys $sk[s'_j]$ since we assumed $l \notin S_{w'}$.) It computes $R \leftarrow R_1 \oplus \ldots \oplus R_{m-1} \oplus M_1 \oplus R_{m+1} \oplus \ldots \oplus R_{q(k)}$ and $R' \leftarrow R'_1 \oplus \ldots \oplus R'_{q(k)}$. It returns 1 if $R = R'$ and else it returns 0. We now claim that

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{PKE},\mathcal{A}}(k) \geq \frac{\mathbf{Adv}^{\text{peks-consist}}_{\mathcal{PEKS},\mathcal{U}}(k)}{d(k)} - 2^{-k} . \tag{13}$$

The assumption that $\mathcal{PKE}$ is IND-CPA now implies that $\mathbf{Adv}^{\text{peks-consist}}_{\mathcal{PEKS},\mathcal{U}}(k)$ is negligible. It remains to see why Equation (13) is true.

Since $w \neq w'$ and the family of sets is cover-free, $S_w \setminus S_{w'} \neq \emptyset$. Note the distribution of pk is independent of l, so $\mathcal{U}$ has no information about l and we can think of it as being chosen after w, w' are output. Then the probability that $l \in S_w \setminus S_{w'}$ is at least $1/d(k)$. So if the challenge bit b equals 1 (i.e. $\overline{C} = C_m$ encrypts M_1) then the probability that $\mathcal{A}$ returns 1 (i.e. that $R' = R$) is at least $\mathbf{Adv}^{\text{peks-consist}}_{\mathcal{PEKS},\mathcal{U}}(k)/d(k)$. On the other hand if $b = 0$ (i.e. $\overline{C} = C_m$ encrypts M_0) then the probability that $\mathcal{A}$ returns 1 (i.e. that $R' = R$) is at most 2^{-k} because in this case R' is distributed independently of R. ▌

A.3 A statistically consistent limited PEKS scheme

In what follows we will show that (a slight modification of) the PEKS scheme from Figure 8 is even statistically consistent when instantiated with a specific PKE scheme.

We let $\mathcal{G}$ be a cyclic group generator. This is a PTA that on input 1^k returns a triple $(\mathbb{G}, p, g)$ where $\mathbb{G}$ is a cyclic group of prime order $p \in \{2^{k-1}, \ldots, 2^k - 1\}$ and g is a generator of $\mathbb{G}$. We assume

```
$\mathsf{KG}(1^k)$
    $(\mathbb{G}, p, g) \xleftarrow{\$} \mathcal{G}(1^k)$
    for $i = 1, \ldots, d(k)$ do $sk[i] \xleftarrow{\$} \mathbb{Z}_p$ ; $pk[i] \leftarrow g^{sk[i]}$
    return $(pk, sk)$

$\mathsf{PEKS}(pk, w)$
    $i \leftarrow \mathrm{Ind}_k(w)$ ; $S_w \leftarrow \mathsf{SUB}(s(k), t(k), i)$
    parse $S_w$ as $S_w = \{s_1, \ldots, s_{q(k)}\}$
    $M_1, \ldots, M_{q(k)} \xleftarrow{\$} \mathbb{G}$ ; $M \leftarrow M_1 \cdot \ldots \cdot M_{q(k)}$
    for $j = 1, \ldots, q(k)$ do
        $r[s_j] \xleftarrow{\$} \mathbb{Z}_p$ ; $C_j \leftarrow (g^{r[s_j]}, M_j \cdot pk[s_j]^{r[s_j]})$
    $C \leftarrow (M, C_1, \ldots, C_{q(k)})$
    return $C$
```

```
$\mathsf{Td}(sk, w)$
    $i \leftarrow \mathrm{Ind}_k(w)$ ; $S_w \leftarrow \mathsf{SUB}(s(k), t(k), i)$.
    parse $S_w$ as $S_w = \{s_1, \ldots, s_{q(k)}\}$
    $t_w \leftarrow (sk[s_1], \ldots, sk[s_{q(k)}])$
    return $t_w$

$\mathsf{Test}(t_w, C)$
    parse $t_w$ as $(sk_1, \ldots, sk_{q(k)})$
    parse $C$ as $(M, C_1, \ldots, C_{q(k)})$
    for $j = 1, \ldots, q(k)$ do
        parse $C_j$ as $(R_j, W_j)$
        $M'_j \leftarrow W_j \cdot R_j^{-sk_j}$
    $M' \leftarrow M'_1 \cdot \ldots \cdot M'_{q(k)}$
    if $M = M'$ then return 1 else return 0
```

Figure 9: Algorithms of the third limited PEKS scheme associated to keyword space KwSp (of size s) and parameter t. The functions q, d are defined in the text.

the DDH problem is hard relative to $\mathcal{G}$. The algorithms of the PEKS scheme $\mathcal{PEKS} = (\mathsf{KG}, \mathsf{PEKS}, \mathsf{Td}, \mathsf{Test})$ are now shown in Figure 9. The parameters t, d, q and the algorithm SUB are as above.

Assuming the DDH problem is hard relative to $\mathcal{G}$, one can prove that the scheme provides privacy against adversaries that query at most $t(k)$ trapdoors. Our interest is the consistency.

Theorem A.3 *The PEKS scheme of Figure 9 is statistically consistent.*

Note that the consistency statement is uncondicional, i.e. it does not rely on the decisional Diffie-Hellman assumption.

Proof: Let $\mathcal{U}$ be an adversary that gets input a pair (pk, sk) of matching public and secret keys, and outputs (wlog distinct) keywords w, w'. Let $C = (M, C_1, \ldots, C_q)$ be formed by encrypting w, and let M' be as defined by $\mathsf{Test}(t_{w'}, C)$. The advantage of $\mathcal{U}$ is the probability that $M = M'$.

Let $S_w = \{s_1, \ldots, s_{q(k)}\}$ and $S_{w'} = \{s'_1, \ldots, s'_{q(k)}\}$ be the subsets associated with w and w', respectively. We consider two cases. The first case is that there is some l such that $s_l \neq s'_l$ but $sk[s_l] = sk[s'_l]$. This happens with negligible probability (specifically at most $d(k)^2/p \leq O(d(k)^2 2^{-k})$) since sk is a sequence of random elements of $\mathbb{Z}_p$. The second case is that this does not happen, meaning that for all l, if $s_l \neq s'_l$ then $sk[s_l] \neq sk[s'_l]$. In this case we show that the probability that $M = M'$ is $1/p \leq O(2^{-k})$ for each fixed choice of (pk, sk) and w, w', the probability only over the coins of the encryption algorithm.

Parse C_j as $(g^{r[s_j]}, M_j \cdot g^{sk[s_j]r[s_j]})$. So $M'_j = M_j \cdot g^{r[s_j](sk[s_j] - sk[s'_j])}$. Thus we want to upper bound the probability that

$$\sum_{i=1}^{q(k)} r[s_j](sk[s_j] - sk[s'_j]) \equiv 0 \pmod{p} \ ,$$

where the probability is taken over the random choices of $r[s_j]$ only. By cover-freeness, there is at least one l for which $s_l \neq s'_l$, and since we are in the second case, $sk[s_l] \neq sk[s'_l]$. Thus the probability over $r[s_l]$ that the above equation holds is $1/p$. ▌

```
Setup(1^k)
  $(\mathbb{G}_1, \mathbb{G}_2, p, e) \xleftarrow{\$} \mathcal{G}(1^k)$
  $P, Q \xleftarrow{\$} \mathbb{G}_1^* \; ; \; \alpha \xleftarrow{\$} \mathbb{Z}_p \; ; \; P_1 \leftarrow \alpha P \; ; \; Q_1 \leftarrow \alpha Q$
  $\boldsymbol{U}[0 \ldots n] \xleftarrow{\$} \mathbb{G}_1^{n+1} \; ; \; E \leftarrow e(P, Q)$
  $pk \leftarrow (\mathbb{G}_1, \mathbb{G}_2, p, e, P, P_1, \boldsymbol{U}, E) \; ; \; msk \leftarrow (pk, Q_1)$
  return $(pk, msk)$

KeyDer($msk, id$)
  parse $msk$ as $((\mathbb{G}_1, \mathbb{G}_2, p, e, P, P_1, \boldsymbol{U}, E), Q_1)$
  $r \xleftarrow{\$} \mathbb{Z}_p \; ; \; V \leftarrow \boldsymbol{U}[0] + \sum_{i=1}^{n} id[i]\boldsymbol{U}[i]$
  $usk[id] \leftarrow (Q_1 + rV, rP)$
  return $usk[id]$

Enc($pk, id, M$)
  parse $pk$ as $(\mathbb{G}_1, \mathbb{G}_2, p, e, P, P_1, \boldsymbol{U}, E)$
  $V \leftarrow \boldsymbol{U}[0] + \sum_{i=1}^{n} id[i]\boldsymbol{U}[i]$
  $t \xleftarrow{\$} \mathbb{Z}_p \; ; \; T \leftarrow E^t$
  $C \leftarrow (T \cdot M, tP, tV)$
  return $C$

Dec($usk[id], C$)
  parse $usk[id]$ as $(S_1, S_2)$, $C$ as $(C_1, C_2, C_3)$
  $T' \leftarrow e(S_1, C_2) \cdot e(S_2, C_3)^{-1}$
  return $T'^{-1} \cdot C_1$
```

Figure 10: The algorithms constituting $\mathcal{W}\text{-}\mathcal{IBE}$. Identities are represented as bit strings $id = id[1, \ldots, n] \in \{0,1\}^n$.

B Attacks against the anonymity of existing schemes

B.1 Waters' IBE scheme

We recall Waters' IBE scheme [21] $\mathcal{W}\text{-}\mathcal{IBE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$ in Figure 10. Associated with $\mathcal{W}\text{-}\mathcal{IBE}$ is a polynomial n. It is assumed that all user identities are $n(k)$-bit (e.g. 160-bit) strings (for instance obtained by hashing the actual identity using a collision-resistant hash function), which are written as $id = id[1]id[2] \ldots id[n]$, where each $id[i]$ $(1 \leq i \leq n)$ is a bit $id[i] \in \{0,1\}$. (We drop the argument k to n when k is understood.) The message space is defined by $\mathsf{MsgSp}(k) = \{0,1\}^k$, and messages are encoded as elements of $\mathbb{G}_2$ in the scheme.

We now describe a PTA $\mathcal{A}$ against the IBE-ANO-CPA-security of $\mathcal{W}\text{-}\mathcal{IBE}$. In the `find` stage it gets input a public key $(\mathbb{G}_1, \mathbb{G}_2, p, e, P, P_1, \boldsymbol{U}, E)$, and returns any two distinct n-bit strings id_0, id_1 as challenge identities, along with any k-bit challenge message. In the `guess` phase, given a challenge ciphertext $C = (C_1, C_2, C_3)$ formed by encrypting M under id_b, where $b \in \{0,1\}$ is the challenge bit, it computes $V' \leftarrow \boldsymbol{U}[0] + \sum_{i=1}^{n} id_1[i]\boldsymbol{U}[i]$. If $e(P, C_3) = e(C_2, V')$ then it returns 1 else it returns 0. It is easy to see that $\mathbf{Adv}^{\text{ibe-ano-cpa}}_{\mathcal{W}\text{-}\mathcal{IBE},\mathcal{A}}(k) \geq 1 - 2^{-k}$.

B.2 Boneh-Boyen's IBE scheme

The IBE scheme by Boneh and Boyen [5], here referred to as $\mathcal{BB}\text{-}\mathcal{IBE}$, is depicted in Figure 11. An identity is represented by a vector of $n(k)$ symbols $id[1 \ldots n] \in \Sigma^n$ where Σ is an alphabet of size s. In the original scheme, these are obtained as the output of an admissible hash function, but we ignore this here as it is irrelevant to the attack.

Consider a PTA $\mathcal{A}$ that, on input $pk = (\mathbb{G}_1, \mathbb{G}_2, p, e, P, P_1, Q, \boldsymbol{U})$, outputs any two distinct identities $id_0, id_1 \in \Sigma^n$, and any message $M \in \{0,1\}^k$. Let $i \in \{1, \ldots, n\}$ be an index so that $id_0[i] \neq id_1[i]$. When $\mathcal{A}$ is given the challenge ciphertext $C = (C_1, \ldots, C_{n+2})$, it checks whether $e(C_2, \boldsymbol{U}[i, id_0[i]]) = e(P, C_{i+2})$. If so, then $\mathcal{A}$ returns 0, else it returns 1. It is easily verified that $\mathbf{Adv}^{\text{ibe-ano-cpa}}_{\mathcal{BB}\text{-}\mathcal{IBE},\mathcal{A}}(k) \geq 1 - 2^{-k}$.

B.3 Boneh-Boyen-Goh's HIBE scheme

The recently proposed $\mathcal{BBG}\text{-}\mathcal{HIBE}$ scheme [6], depicted in Figure 12, is not anonymous at any single level, and therefore not at any set of multiple levels either. This can be seen from the following adver-

```
Setup$(1^k)$
  $(\mathbb{G}_1, \mathbb{G}_2, p, e) \xleftarrow{\$} \mathcal{G}(1^k)$
  $P, Q \xleftarrow{\$} \mathbb{G}_1^*$ ; $\alpha \xleftarrow{\$} \mathbb{Z}_p$ ; $P_1 \leftarrow \alpha P$ ; $Q_1 \leftarrow \alpha Q$
  $\boldsymbol{U}[1 \ldots n, 1 \ldots s]) \xleftarrow{\$} \mathbb{G}_1^{n \times s}$
  $pk \leftarrow (\mathbb{G}_1, \mathbb{G}_2, p, e, P, P_1, Q, \boldsymbol{U})$ ; $msk \leftarrow (pk, Q_1)$
  return $(pk, msk)$

KeyDer$(msk, id)$
  parse $msk$ as $((\mathbb{G}_1, \mathbb{G}_2, p, e, P, P_1, \boldsymbol{U}, Q), Q_1)$
  $r_1, \ldots, r_n \xleftarrow{\$} \mathbb{Z}_p$ ; $V \leftarrow \sum_{i=1}^n r_i \boldsymbol{U}[i, id[i]]$
  $usk[id] \leftarrow (Q_1 + V,\ r_1 P,\ \ldots,\ r_n P)$
  return $usk[id]$

Enc$(pk, id, M)$
  parse $pk$ as $(\mathbb{G}_1, \mathbb{G}_2, p, e, P, P_1, \boldsymbol{U}, Q)$
  $t \xleftarrow{\$} \mathbb{Z}_p$ ; $T \leftarrow e(P_1, Q)^t$
  $C \leftarrow (T \cdot M, tP, t\boldsymbol{U}[1, id[1]], \ldots, t\boldsymbol{U}[n, id[n]])$
  return $C$

Dec$(usk[id], C)$
  parse $usk[id]$ as $(S_1, S_2, \ldots, S_{n+1})$
  parse $C$ as $(C_1, \ldots, C_{n+2})$
  $T' \leftarrow e(S_1, C_2) \cdot \prod_{i=1}^n e(S_{i+1}, C_{i+2})^{-1}$
  return $T'^{-1} \cdot C_1$
```

Figure 11: The algorithms constituting $\mathcal{BB}\text{-}\mathcal{IBE}$. Identities are represented as vectors of symbols $id = id[1, \ldots, n] \in \Sigma^n$, where $|\Sigma| = s$.

sary $\mathcal{A}$ that breaks the anonymity at level l. On input $pk = (\mathbb{G}_1, \mathbb{G}_2, p, e, P, P_1, Q, Q_2, \boldsymbol{U})$, adversary $\mathcal{A}$ outputs challenge identities $(id_1, \ldots, id_{l-1}, id_l)$ and $(id_1, \ldots, id_{l-1}, id'_l)$ for any $id_1, \ldots, id_l, id'_l \in \mathbb{Z}_p$ such that $id_l \neq id'_l$, and any challenge message $M \in \{0,1\}^k$. When given the challenge ciphertext $C = (C_1, C_2, C_3)$, $\mathcal{A}$ checks whether $e(C_2, id_1\boldsymbol{U}[1] + \ldots + id_l\boldsymbol{U}[l] + Q_2) = e(P, C_3)$. If this is the case, then $\mathcal{A}$ returns 0, otherwise it returns 1. It is easily verified that $\mathbf{Adv}^{\text{hibe-ano-cpa}[l,d]}_{\mathcal{BBG}\text{-}\mathcal{HIBE},\mathcal{A}}(k) \geq 1 - 2^{-k}$.

C Proof of Lemma 5.3

Suppose that there is an adversary $\mathcal{A}$ of $m\mathcal{GS}\text{-}\mathcal{HIBE}$ that breaks its HIBE-IND-CPA[d] security. We will show how to use $\mathcal{A}$ in the construction of a simulator $\mathcal{B}$ that solves the bilinear Diffie-Hellman problem. Let $n_{1,i}$ be the number of queries that $\mathcal{A}$ makes to the $H_{1,i}$ oracle, let n_2 be the number of queries to the H_2 oracle, let n_e be the number of queries to the key extraction oracle, and let $n_h = \sum_{i-1}^{d(k)} n_{1,i} + n_2$ be the total number of hash queries.

The simulator is given as input (P, aP, bP, cP). It sets $Q_0 \leftarrow bP$ as the public key and then runs $\mathcal{A}(\texttt{find}, pk)$. The simulator responds to $\mathcal{A}$'s queries as described below. To maintain consistency between queries it keeps lists $L_{1,1}, \ldots, L_{1,d(k)}$, L_2 and L_3. All lists are initially empty. At the very beginning the simulator chooses $n^*_{1,i} \xleftarrow{\$} \{1, \ldots, n_{1,i}\}$ and $s^*_i, x^*_i \xleftarrow{\$} \mathbb{Z}^*_p$, and it computes $Q^*_i \leftarrow s^*_i(bP)$ for $1 \leq i \leq d(k)$.

For the description of the simulation we distinguish between $H_{1,1}$ queries and $H_{1,i}$ queries for $i \geq 2$. Without loss of generality, we assume that before querying the KeyDer oracle to obtain the secret key of $id = (id_1, \ldots, id_l)$, adversary $\mathcal{A}$ first queried $H_{1,i}(id_i)$ for all $1 \leq i \leq l$.

$H_{1,1}$ **Queries:** To respond to a query id_1, proceed as follows.

- If $L_{1,1}$ contains $(id_1, P_1, *)$ for some P_1, respond with P_1.
- If this is the $n^*_{1,1}$-th call to the $H_{1,1}$ oracle, let $id^*_1 \leftarrow id_1$, add $(id^*_1, aP, \perp)$ to $L_{1,1}$ and respond with aP.
- Else, randomly choose an integer $x_1 \xleftarrow{\$} \mathbb{Z}^*_p$, add (id_1, x_1P, x_1) to $L_{1,1}$ and reply with x_1P.

$H_{1,i}$ **Queries,** $i \geq 2$**:** To respond to a query id_i, proceed as follows.

```
Setup$(1^k)$
  $(\mathbb{G}_1, \mathbb{G}_2, p, e) \xleftarrow{\$} \mathcal{G}(1^k)$
  $P \xleftarrow{\$} \mathbb{G}_1$ ; $\alpha \xleftarrow{\$} \mathbb{Z}_p$ ; $P_1 \leftarrow \alpha P$
  $Q, Q_2 \xleftarrow{\$} \mathbb{G}_1$ ; $Q_1 \leftarrow \alpha Q$
  $\boldsymbol{U}[1 \ldots d(k)]) \xleftarrow{\$} \mathbb{G}_1^{d(k)}$
  $pk \leftarrow (\mathbb{G}_1, \mathbb{G}_2, p, e, P, P_1, Q, Q_2, \boldsymbol{U})$
  $msk \leftarrow (pk, Q_1, 0, \ldots, 0)$
  return $(pk, msk)$

KeyDer$(usk, id)$
  $l \leftarrow |id|$ ; parse $usk$ as $(pk, A, B, S_l, \ldots, S_{d(k)})$
  parse $pk$ as $(\mathbb{G}_1, \mathbb{G}_2, p, e, P, P_1, Q, Q_2, \boldsymbol{U})$
  parse $id$ as $(id_1, \ldots, id_l)$ ; $r \xleftarrow{\$} \mathbb{Z}_p$
  $A' \leftarrow A + id_l S_l + r(id_1 \boldsymbol{U}[1] + \ldots + id_l \boldsymbol{U}[l] + Q_2)$
  $B' \leftarrow B + rP$
  for $l+1 \le i \le d(k)$ do $S'_i \leftarrow S_i + r\boldsymbol{U}[i]$
  return $(pk, A', B', S'_{l+1}, \ldots, S'_{d(k)})$

Enc$(pk, id, M)$
  parse $pk$ as $(\mathbb{G}_1, \mathbb{G}_2, p, e, P, P_1, Q, Q_2, \boldsymbol{U})$
  parse $id$ as $(id_1, \ldots, id_l)$
  $t \xleftarrow{\$} \mathbb{Z}_p$ ; $T \leftarrow e(P_1, Q)^t$
  $C_3 \leftarrow t(id_1 \boldsymbol{U}[1] + \ldots + id_l \boldsymbol{U}[l] + Q_2)$
  $C \leftarrow (T \cdot M, tP, C_3)$
  return $C$

Dec$(usk, C)$
  parse $usk$ as $(pk, A, B, S_l, \ldots, S_{d(k)})$
  parse $pk$ as $(\mathbb{G}_1, \mathbb{G}_2, p, e, P, P_1, Q, Q_2, \boldsymbol{U})$
  parse $C$ as $(C_1, C_2, C_3)$
  $T' \leftarrow e(A, C_2) \cdot e(B, C_3)^{-1}$
  return $T'^{-1} \cdot C_1$
```

Figure 12: The algorithms constituting $\mathcal{BBG}$-$\mathcal{HIBE}$ with maximum hierarchy depth $d(k)$. An identity at level l is represented as a vector $id = (id_1, \ldots, id_l) \in \mathbb{Z}_p^l$.

- If $L_{1,i}$ contains $(id_i, P_i, *)$ for some P_i then respond with P_i.
- If this is the $n^*_{1,i}$-th query to the $H_{1,i}$ oracle, let $id^*_i \leftarrow id_i$, add $(id^*_i, x^*_i P, x^*_i)$ to $L_{1,i}$ and respond with $x^*_i P$.
- Else, choose an integer $x_i \xleftarrow{\$} \mathbb{Z}_p^*$ and compute $P_i \leftarrow x_i P - {s^*_{i-1}}^{-1}(aP + \sum_{j=2}^{i-1} s^*_{j-1} x^*_j P)$. If $P_i = 0$, then abort; else, add (id_i, P_i, x_i) to $L_{1,i}$ and reply with P_i.

H_2 **Queries:** To respond to a query κ, proceed as follows.

- If $(\kappa, K) \in L_2$ for some K, respond with K.
- Else, choose K uniformly at random from $\{0,1\}^n$, respond with K and add (κ, K) to L_2.

KeyDer **Queries:** To respond to a query $id = (id_1, \ldots, id_l)$, proceed as follows.

- If $(id_1, \ldots, id_l) = (id^*_1, \ldots, id^*_l)$, then $\mathcal{B}$ aborts.
- Let j be the largest integer $1 \le j \le l$ so that $(id|_j, S_j, Q_1, \ldots, Q_{j-1}, s_j) \in L_3$, or let $j = 0$ if such element does not exist.
- For $i = j+1, \ldots, l$, do the following:
  - Find $(id_i, P_i, x_i) \in L_{1,i}$.
  - If $i = 1$ and $id_1 = id^*_1$, then add $(id^*_1, \perp, \perp)$ to L_3. If $i = 1$ and $id_1 \neq id^*_1$, then compute $S_1 \leftarrow x_i(bP)$, choose $s_1 \xleftarrow{\$} \mathbb{Z}_p^*$, and add (id_1, S_1, s_1) to L_3.
  - If $i > 1$ and $S_{i-1} \neq \perp$, then look up (id_i, P_i, x_i) in $L_{1,i}$, compute $S_i \leftarrow S_{i-1} + s_{i-1} P_i$, $Q_{i-1} \leftarrow s_{i-1} P$, choose $s_i \xleftarrow{\$} \mathbb{Z}_p^*$, and add $(id|_i, S_i, Q_1, \ldots, Q_{i-1}, s_i)$ to L_3.
  - If $i > 1$ and $S_{i-1} = \perp$ and $id_i = id^*_i$, then compute $Q_{i-1} \leftarrow s^*_{i-1}(bP)$ and add $(id|_i, \perp, Q_1, \ldots, Q_{i-1}, \perp)$ to L_3.

– If $i > 1$, $S_{i-1} = \bot$ and $id_i \neq id_i^*$, then look up (id_i, P_i, x_i) in $L_{1,i}$, compute $S_i \leftarrow s_{i-1}^* x_i P$, let $Q_{i-1} \leftarrow s_{i-1}^*(bP)$, choose $s_i \xleftarrow{\$} \mathbb{Z}_p^*$, and add $(id|_i, S_i, Q_1^*, \ldots, Q_{i-1}^*, s_i)$ to L_3.

- Find $(id, S_l, Q_1, \ldots, Q_{l-1}, s_l) \in L_3$ and return $(id, S_l, Q_1, \ldots, Q_{l-1}, s_l)$.

At some point $\mathcal{A}$ outputs $(id = (id_1, id_2, \ldots, id_l), M_0, M_1, state)$. Without loss of generality, we assume that the adversary submitted id_i to the $H_{1,i}$ oracle before for all $1 \leq i \leq l$. If $id \neq (id_1^*, \ldots, id_l^*)$, then $\mathcal{B}$ aborts. Otherwise, he sets $C_1^* \leftarrow cP$, $C_2^* \leftarrow x_2^*(cP)$, …, $C_l \leftarrow x_l^*(cP)$, he chooses C_{l+1}^* uniformly at random from $\{0,1\}^n$, and lets $C^* \leftarrow (C_1^*, C_2^*, \ldots, C_{l+1}^*)$. He then proceeds to run $\mathcal{A}(\texttt{guess}, C^*, state)$. Once $\mathcal{A}$ completes its attack by outputting its guess b', the simulator chooses a random element (κ, K) from L_2 and outputs κ as its solution to the bilinear Diffie-Hellman problem.

We first show that our simulator $\mathcal{B}$ provides a real attack environment for $\mathcal{A}$ as long as $\mathcal{B}$ doesn't abort. The public key pk given to $\mathcal{A}$ is correctly distributed because the challenge elements aP, bP, cP are random elements from $\mathbb{G}_1^*$. The responses to $H_{1,i}$ queries are uniformly distributed over $\mathbb{G}_1^*$ due to the independent random choices of x_i (when simulating queries $H_{1,i}(id_i)$, $id_i \neq id_i^*$, $1 \leq i \leq d(k)$), of x_i^* (which is used to simulate $H_{1,i}(id_i^*)$ queries, $2 \leq i \leq d(k)$) and due to the uniform distribution of aP (which is used to simulate $H_{1,1}(id_1^*)$). Responses to H_2 queries are easily seen to be correctly distributed. The way KeyDer queries are handled requires a bit more explanation. For all level-1 identities $id_1 \neq id_1^*$, the returned secret key (S_1, s_1) contains the unique group element S_1 such that $e(Q_0, H_{1,1}(id_1)) = e(S_1, P)$ and a uniformly distributed scalar s_1, as in the real game. For all descendants of $id_1 \neq id_1^*$, the secret keys are derived from (S_1, s_1) exactly as in the real scheme. Now consider identity $(id_1^*, \ldots, id_{i-1}^*, id_i)$ with $id_i \neq id_i^*$, for which a tuple $(S_i, Q_1, \ldots, Q_{i-1}, s_i)$ is returned as the secret key. The values $Q_1, \ldots, Q_{i-2}$ are inherited from the ancestors, as in the real scheme; Q_{i-1} is a random group element due to the random choice of s_{i-1}^*; and s_i is a random element in $\mathbb{Z}_p^*$. The simulated value $S_i = s_{i-1}^* x_i(bP)$ is then the unique group element such that $e(H_{1,1}(id_1^*), Q_0) = e(S_i, P) \cdot \prod_{j=2}^{i-1} e(H_{1,j}(id_j^*), Q_{j-1})^{-1} \cdot e(H_{1,i}(id_i), Q_{i-1})^{-1}$, as required by the scheme. This can be seen from:

$$
\begin{aligned}
& e(H_{1,1}(id_1^*),\, Q_0) \cdot \prod_{j=2}^{i-1} e(H_{1,j}(id_j^*),\, Q_{j-1}) \cdot e(H_{1,i}(id_i),\, Q_{i-1}) \\
&= \quad e(aP,\, bP) \cdot \prod_{j=2}^{i-1} e(x_j^* P,\, s_{j-1}^* bP) \cdot e\big(x_i P - {s_{i-1}^*}^{-1}(aP + \textstyle\sum_{j=2}^{i-1} s_{j-1}^* x_j^* P),\, s_{i-1}^* bP\big) \\
&= \quad e(aP,\, bP) \cdot e\big(\textstyle\sum_{j=2}^{i-1} s_{j-1}^* x_j^* P,\, bP\big) \cdot e\big(s_{i-1}^* x_i P - aP - \textstyle\sum_{j=2}^{i-1} s_{j-1}^* x_j^* P,\, bP\big) \\
&= \quad e(S_3, P)\ .
\end{aligned}
$$

The secret keys of descendants of these nodes are derived from $(S_i, Q_1, \ldots, Q_{i-1}, s_i)$ as dictated by the scheme, and hence are correctly distributed as well.

The only part of $\mathcal{A}$'s environment left to analyze is the challenge ciphertext $C^* = (C_1^*, \ldots, C_{l+1}^*)$. The first component $C_1^* = cP$ is uniformly distributed over $\mathbb{G}_1^*$, and the second to l-th components are the unique group elements such that $e(C_i^*, P) = e(C_1^*, H_{1,i}(id_i^*))$ for $2 \leq i \leq l$. The last component C_{l+1}^* however may deviate from the distribution in a real game, depending on $\mathcal{A}$'s H_2 queries. In the following, we show that this does not harm our analysis, intuitively because the only way $\mathcal{A}$ can distinguish between the real and the simulated game is by making an H_2 query that helps $\mathcal{B}$ solve the BDH problem.

Let s_0 be the master secret key of the scheme in a real HIBE-IND-CPA[d] attack on $\mathcal{mGS\text{-}HIBE}$, and let $D \leftarrow e(s_0 H_{1,1}(id_1),\ C_1^*)$. Let Ask be the event that $\mathcal{A}$ queries the H_2 oracle on point D. Let

$\Pr_{\mathrm{R}}[\,\cdot\,]$ denote the probability of an event taking place in a real attack on $\mathcal{mGS\text{-}HIBE}$, and let $\Pr_{\mathcal{B}}[\,\cdot\,]$ denote the probability in the environment simulated by $\mathcal{B}$. We argue that $\Pr_{\mathrm{R}}[\,\textsc{Ask}\,] = \Pr_{\mathcal{B}}[\,\textsc{Ask}\,]$, as long as $\mathcal{B}$ doesn't abort. Let $\textsc{Ask}_i$ be the event that $\mathcal{A}$ queries $H_2(D)$ within the first i queries to H_2. Obviously, $\Pr_{\mathrm{R}}[\,\textsc{Ask}_0\,] = \Pr_{\mathcal{B}}[\,\textsc{Ask}_0\,] = 0$. Now assume that $\Pr_{\mathrm{R}}[\,\textsc{Ask}_{i-1}\,] = \Pr_{\mathcal{B}}[\,\textsc{Ask}_{i-1}\,]$. We have that

$$\begin{aligned}\Pr_{\mathrm{R}}[\,\textsc{Ask}_i\,] &= \Pr_{\mathrm{R}}[\,\textsc{Ask}_i \mid \textsc{Ask}_{i-1}\,]\cdot\Pr_{\mathrm{R}}[\,\textsc{Ask}_{i-1}\,]\\ &\quad+\ \Pr_{\mathrm{R}}[\,\textsc{Ask}_i \mid \neg\textsc{Ask}_{i-1}\,]\cdot\Pr_{\mathrm{R}}[\,\neg\textsc{Ask}_{i-1}\,]\\ &= \Pr_{\mathrm{R}}[\,\textsc{Ask}_{i-1}\,] + \Pr_{\mathrm{R}}[\,\textsc{Ask}_i \mid \neg\textsc{Ask}_{i-1}\,]\cdot\Pr_{\mathrm{R}}[\,\neg\textsc{Ask}_{i-1}\,]\ .\end{aligned}$$

We know that $\Pr_{\mathrm{R}}[\,\textsc{Ask}_{i-1}\,] = \Pr_{\mathcal{B}}[\,\textsc{Ask}_{i-1}\,]$, so we only have to show that $\Pr_{\mathrm{R}}[\,\textsc{Ask}_i \mid \neg\textsc{Ask}_{i-1}\,] = \Pr_{\mathcal{B}}[\,\textsc{Ask}_i \mid \neg\textsc{Ask}_{i-1}\,]$. Given that $\neg\textsc{Ask}_{i-1}$ and that $\mathcal{B}$'s simulation didn't abort, the simulated public key, the oracle responses and the first l components of the ciphertext provided by $\mathcal{B}$ are distributed exactly as in a real attack, as we explained before. Moreover, since $\mathcal{A}$ did not query for $H_2(D)$ yet, from $\mathcal{A}$'s point of view the last ciphertext component C^*_{l+1} is a random string in $\{0,1\}^n$, both in the real attack and in the simulated environment. Since all the information on which $\mathcal{A}$ can base its decision for its next H_2 query is identically distributed in both environments, the probability that $\mathcal{A}$ chooses to query D is the same in both environments as well. Hence, we have that $\Pr_{\mathrm{R}}[\,\textsc{Ask}_i\,] = \Pr_{\mathcal{B}}[\,\textsc{Ask}_i\,]$, and by induction that $\Pr_{\mathrm{R}}[\,\textsc{Ask}\,] = \Pr_{\mathcal{B}}[\,\textsc{Ask}\,]$.

The probability that $\mathcal{A}$ wins a real attack against $\mathcal{mGS\text{-}HIBE}$ can be written as

$$\begin{aligned}\Pr_{\mathrm{R}}[\,\mathcal{A}\ \textsc{wins}\,] &= \Pr_{\mathrm{R}}[\,\mathcal{A}\ \textsc{wins} \wedge \textsc{Ask}\,] + \Pr_{\mathrm{R}}[\,\mathcal{A}\ \textsc{wins} \wedge \neg\textsc{Ask}\,]\\ &= \Pr_{\mathrm{R}}[\,\mathcal{A}\ \textsc{wins} \wedge \textsc{Ask}\,] + \frac{1}{2}\\ &\leq \Pr_{\mathrm{R}}[\,\textsc{Ask}\,] + \frac{1}{2}\ ,\end{aligned}$$

where the second equation is true because in the event $\neg\textsc{Ask}$, the distribution of the challenge ciphertext is completely independent of M_0, M_1, and hence the probability that $\mathcal{A}$ guesses correctly is $1/2$. Since $\Pr_{\mathrm{R}}[\,\textsc{Ask}\,] = \Pr_{\mathcal{B}}[\,\textsc{Ask}\,]$ and moreover

$$\Pr_{\mathrm{R}}[\,\mathcal{A}\ \textsc{wins}\,] \;=\; \frac{1}{2}\cdot\mathbf{Adv}^{\text{hibe-ind-cpa}[d]}_{\mathcal{mGS\text{-}HIBE},\mathcal{A}}(k) + \frac{1}{2}\ ,$$

it follows that

$$\Pr_{\mathcal{B}}[\,\textsc{Ask}\,] \;\geq\; \frac{1}{2}\cdot\mathbf{Adv}^{\text{hibe-ind-cpa}[d]}_{\mathcal{mGS\text{-}HIBE},\mathcal{A}}(k)\ .$$

Now we only have to relate $\mathcal{B}$'s advantage in solving the BDH problem to $\Pr_{\mathcal{B}}[\,\textsc{Ask}\,]$. In the game simulated by $\mathcal{B}$, the probability that $\mathcal{B}$ guesses the correct identities such that $id = (id^*_1,\ldots,id^*_l)$ $1/\prod_{i=1}^{l} n_{1,i} \geq n_h^{-d(k)}$; the probability that $\mathcal{B}$ guesses the correct H_2 query is $1/n_2 \geq n_h^{-1}$; and the probability that $\mathcal{B}$ aborts when answering $H_{1,i}$ queries is $\sum_{i=1}^{d(k)} n_{1,i}/(p-1) \leq n_h/2^k$. The advantage of $\mathcal{B}$ in solving the BDH problem is

$$\mathbf{Adv}^{\text{bdh}}_{\mathcal{G},\mathcal{B}}(k) \;\geq\; \frac{1}{n_h^{d(k)+1}}\cdot\left(1-\frac{n_h}{2^k}\right)\cdot\Pr_{\mathcal{B}}[\,\textsc{Ask}\,]$$

and hence

$$\mathbf{Adv}^{\text{hibe-ind-cpa}[d]}_{\mathcal{mGS\text{-}HIBE},\mathcal{A}}(k) \;\leq\; 2\cdot n_h^{d(k)+1}\cdot\mathbf{Adv}^{\text{bdh}}_{\mathcal{G},\mathcal{B}}(k) + \frac{n_h}{2^k}\ ,$$

from which the theorem follows.