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

Michel Abdalla[1], Mihir Bellare[2], Dario Catalano[1], Eike Kiltz[2], Tadayoshi Kohno[2], Tanja Lange[3], John Malone-Lee[4], Gregory Neven[5], Pascal Paillier[6], and Haixia Shi[2]

[1] Departement d'Informatique, École normale supérieure. {Michel.Abdalla,Dario.Catalano}@ens.fr, http://www.di.ens.fr/users/{mabdalla,catalano}
[2] Department of Computer Science & Engineering, University of California San Diego. {mihir,ekiltz,tkohno,hashi}@cs.ucsd.edu, http://www.cs.ucsd.edu/users/{mihir,ekiltz,tkohno,hashi}
[3] Department of Mathematics, Technical University of Denmark. t.lange@mat.dtu.dk, http://www.ruhr-uni-bochum.de/itsc/tanja
[4] Department of Computer Science, University of Bristol. malone@compsci.bristol.ac.uk, http://www.cs.bris.ac.uk/~malone
[5] Department of Electrical Engineering, Katholieke Universiteit Leuven. Gregory.Neven@esat.kuleuven.be, http://www.neven.org
[6] Applied Research and Security Center, Gemplus Card International. Pascal.Paillier@gemplus.com.

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.

1 Introduction

There has recently been interest in various forms of "searchable encryption" [18, 7, 12, 14, 20]. 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 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 advantage 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 [13, 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 [2, 10]). 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 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 4) 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 [3].) 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. Sect. 3) then $\mathcal{PEKS} = \mathsf{bdop\text{-}ibe\text{-}2\text{-}peks}(\mathcal{IBE})$ is PEKS-IND-CPA.

CONSISTENCY IN bdop-ibe-2-peks. Unfortunately, we show (cf. Theorem 6) 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. 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 [15] to the IBE setting. 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 8). (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 [1] an attack to show that Water's IBE scheme [19] 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 [11, 6] are not anonymous, even at level 1. (That of [16] 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 [11] 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

Limited PEKS schemes. Boneh et. al. [7] also present a couple of PEKS schemes that are what they call *limited*. 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 the full version of this paper [1] we extend our definitions of consistency to this limited setting and then show that the first scheme is statistically consistent while the second scheme is computationally consistent, and statistically consistent under some conditions.

Consistency of other searchable encryption schemes. Of the other papers on searchable encryption of which we are aware [18, 12, 14, 20], none formally define or rigorously address the notion of consistency for their respective types of searchable encryption schemes. Goh [12] and Golle, Staddon, and Waters [14] define consistency conditions analogous to BDOP's "perfect consistency" condition, but none of the constructions in [12, 14] satisfy their respective perfect consistency condition. Song, Wagner, and Perrig [18] and Waters et al. [20] do not formally state and prove consistency conditions for their respective searchable encryption schemes, but they, as well as Goh [12], do acknowledge and informally bound the non-zero probability of a false positive.

2 Consistency in PEKS

PEKS. A *public key encryption with keyword search* (PEKS) scheme [7] $\mathcal{PEKS} = (\mathsf{KG}, \mathsf{PEKS}, \mathsf{Td}, \mathsf{Test})$ consists of four polynomial-time algorithms. Via $(pk, sk) \stackrel{\$}{\leftarrow} \mathsf{KG}(1^k)$, where $k \in \mathbb{N}$ is the security parameter, the randomized key-generation algorithm produces keys for the receiver; via $C \stackrel{\$}{\leftarrow} \mathsf{PEKS}^H(pk, w)$ a sender encrypts a keyword w to get a ciphertext; via $t_w \stackrel{\$}{\leftarrow} \mathsf{Td}^H(sk, w)$ 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. (In constructs we might use multiple random oracles, but since one can always obtain these from a single one [5], definitions will assume just one.)

Consistency. The requirement of [7] can be divided into two parts. First, $\mathsf{Test}(t_w, C)$ always accept when C encrypts w. More formally, for all $k \in \mathbb{N}$ and all $w \in \{0,1\}^*$ we ask that $\Pr[\mathsf{Test}^H(\mathsf{Td}^H(sk, w), \mathsf{PEKS}^H(pk, w)) = 1] = 1$, where the probability is taken over the choice of $(pk, sk) \stackrel{\$}{\leftarrow} \mathsf{KG}(1^k)$, 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 $w, w' \in \{0,1\}^*$, we ask that $\Pr[\mathsf{Test}^H(\mathsf{Td}^H(sk, w'), \mathsf{PEKS}^H(pk, w)) = 1] = 0$, where the probability is taken over the choice of $(pk, sk) \stackrel{\$}{\leftarrow} \mathsf{KG}(1^k)$, 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, let $\mathcal{A}$ be an adversary and let k be the security parameter. Below, we depict an experiment, where $b \in \{0,1\}$ is a bit, and also the oracle provided to the adversary in this experiment.

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

Oracle $\text{Trapd}(w)$
- $WSet \leftarrow WSet \cup \{w\}$
- $t_w \stackrel{\$}{\leftarrow} \mathsf{Td}^H(sk, w)$
- Return t_w

The PEKS-IND-CPA-*advantage* $\mathbf{Adv}^{\text{peks-ind-cpa}}_{\mathcal{PEKS},\mathcal{A}}(k)$ of $\mathcal{A}$ is defined as

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

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

Parameter generation algorithms and the BDH problem. All pairing based schemes will be parameterized by a *pairing parameter generator*. This is a randomized polynomial-time algorithm $\mathcal{G}$ 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 $e\colon \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. 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

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

$\mathsf{PEKS}^{H_1,H_2}(pk, w)$
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

$\mathsf{Td}^{H_1}(sk, w)$
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

$\mathsf{Test}^{H_1,H_2}(t_w, C)$
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

Fig. 1. Algorithms constituting the PEKS scheme $\mathcal{BDOP\text{-}PEKS}$. $\mathcal{G}$ 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.

group elements except the neutral element. We define the advantage $\mathbf{Adv}^{\mathrm{bdh}}_{\mathcal{G},\mathcal{A}}(k)$ of an adversary $\mathcal{A}$ in solving the BDH problem relative to a pairing parameter generator $\mathcal{G}$ as the probability that, on input $(1^k, (\mathbb{G}_1, \mathbb{G}_2, p, e), P, aP, bP, cP)$ for randomly chosen $P \xleftarrow{\$} \mathbb{G}_1^*$ and $a, b, c \xleftarrow{\$} \mathbb{Z}_p^*$, adversary $\mathcal{A}$ outputs $e(P,P)^{abc}$. We say that the BDH problem is hard relative to this generator if $\mathbf{Adv}^{\mathrm{bdh}}_{\mathcal{G},\mathcal{A}}$ is a negligible function in k for all polynomial-time adversaries $\mathcal{A}$.

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

Proposition 1. *The $\mathcal{BDOP\text{-}PEKS}$ 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. $\square$

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.

New notions of consistency. The full version of this paper [1] considers various possible relaxations of perfect consistency and argues that the obvious ones are inadequate either because $\mathcal{BDOP\text{-}PEKS}$ continues to fail them or because they are too weak. It then motivates and explains our approach and definitions. In this extended abstract we simply state our definitions, referring the reader to [1] for more information.

Definition 2. *Let $\mathcal{PEKS} = (\mathsf{KG}, \mathsf{PEKS}, \mathsf{Td}, \mathsf{Test})$ be a PEKS scheme. Let $\mathcal{U}$ be an adversary and let k be the security parameter. Consider 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, and* computationally consistent *if it is negligible for all polynomial-time adversaries.*

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.

CONSISTENCY OF $\mathcal{BDOP}$-$\mathcal{PEKS}$, REVISITED. 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 1 extends to show:

Proposition 3. *The $\mathcal{BDOP}$-$\mathcal{PEKS}$ scheme is not statistically consistent.*

The proof is in [1]. On the positive side, the following, proved in [1], means that $\mathcal{BDOP}$-$\mathcal{PEKS}$ is probably just fine in practice:

Theorem 4. *The $\mathcal{BDOP}$-$\mathcal{PEKS}$ scheme is computationally consistent in the random oracle model.*

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 $\mathcal{PEKS}$-$\mathcal{STAT}$ are then depicted in Fig. 2. We are denoting by $|x|$ the length of a string x. The scheme uses ideas from the $\mathcal{BDOP}$-$\mathcal{PEKS}$ scheme [7] as well as from the $\mathcal{BF}$-$\mathcal{IBE}$ scheme [8], but adds some new elements. 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 polynomial-time. This is because polynomial time 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 formal statement of our result is the following:

Theorem 5. *The $\mathcal{PEKS}$-$\mathcal{STAT}$ scheme is statistically consistent in the random oracle model. Further, $\mathcal{PEKS}$-$\mathcal{STAT}$ is* PEKS-IND-CPA*-secure in the random oracle model assuming that the BDH problem is hard relative to generator $\mathcal{G}$.*

We refer to [1] for the proof, and provide a little intuition here. Privacy when the adversary is restricted to attacking the scheme only on keywords of size at

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

Fig. 2. Algorithms constituting the PEKS scheme $\mathcal{PEKS\text{-}STAT}$. Here $f(k) = k^{\lg(k)}$, $\mathcal{G}$ is a pairing parameter generator and $H_1 : \{0,1\}^* \to \mathbb{G}_1$, $H_2 : \mathbb{G}_2 \to \{0,1\}^{3k}$, $H_3 : \{0,1\}^* \to \{0,1\}^k$, and $H_4 : \{0,1\}^* \to \{0,1\}^k$ are random oracles. In implementations we require that the lengths of the encodings of (rP, c, t, K_2) be polynomial in k.

most $f(k)$ can be shown based on techniques used to prove IBE-IND-CPA of the $\mathcal{BF}$-$\mathcal{IBE}$ scheme [8] and to prove anonymity of the same scheme (cf. Theorem 8). When the keyword has length at least $f(k)$ it is sent in the clear, but intuitively the reason this does not violate privacy is that the adversary is poly(k) time and thus cannot even write down such a keyword in order to query it to its challenge oracle. More interesting is the proof of statistical consistency. 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. The scheme is designed so that the adversary effectively has to find a large number of collisions to win. It uses the fact that H_2 is with high probability injective, and then uses a counting argument based on an occupancy problem bound.

3 PEKS and Anonymous IBE

IBE. An *identity-based encryption* (IBE) scheme [17,8] $\mathcal{IBE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$ consists of four polynomial-time algorithms. 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. (In constructs we might use multiple random oracles, but since one can always obtain these from a single one [5], definitions will assume just one.) Associated to the scheme is a message space MsgSp where for $\mathsf{MsgSp}(k) \subseteq \{0,1\}^*$ for every $k \in \mathbb{N}$. 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 $(pk, msk) \stackrel{\$}{\leftarrow} \mathsf{Setup}(1^k)$, the random choice of H, and the coins of all the algorithms in the expression above. Unless otherwise stated, it is assumed that $\{0,1\}^k \subseteq \mathsf{MsgSp}(k)$ for all $k \in \mathbb{N}$.

Privacy and anonymity. Privacy (IBE-IND-CPA) follows [8] while anonymity (IBE-ANO-CPA) is a straightforward adaptation of [3] to IBE schemes. Let $\mathcal{IBE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$ be an IBE scheme. Let $\mathcal{A}$ be an adversary and let k be the security parameter. Now consider the following experiments, where $b \in \{0,1\}$ is a bit:

Experiment $\mathbf{Exp}^{\text{ibe-ind-cpa-b}}_{\mathcal{IBE},\mathcal{A}}(k)$	**Experiment** $\mathbf{Exp}^{\text{ibe-ano-cpa-b}}_{\mathcal{IBE},\mathcal{A}}(k)$
$IDSet \leftarrow \emptyset$; $(pk, msk) \stackrel{\$}{\leftarrow} \mathsf{Setup}(1^k)$	$IDSet \leftarrow \emptyset$; $(pk, msk) \stackrel{\$}{\leftarrow} \mathsf{Setup}(1^k)$
Pick random oracle H	Pick random oracle H
$(id, M_0, M_1, st) \stackrel{\$}{\leftarrow} \mathcal{A}^{\text{KeyDer},H}(\mathtt{find}, pk)$	$(id_0, id_1, M, st) \stackrel{\$}{\leftarrow} \mathcal{A}^{\text{KeyDer},H}(\mathtt{find}, pk)$
$C \stackrel{\$}{\leftarrow} \mathsf{Enc}^H(pk, id, M_b)$	$C \stackrel{\$}{\leftarrow} \mathsf{Enc}^H(pk, id_b, M)$
$b' \stackrel{\$}{\leftarrow} \mathcal{A}^{\text{KeyDer},H}(\mathtt{guess}, C, st)$	$b' \stackrel{\$}{\leftarrow} \mathcal{A}^{\text{KeyDer},H}(\mathtt{guess}, C, st)$
If $\{M_0, M_1\} \not\subseteq \mathsf{MsgSp}(k)$ then return 0	If $M \notin \mathsf{MsgSp}(k)$ then return 0
If $id \notin IDSet$ and $\lvert M_0 \rvert = \lvert M_1 \rvert$	If $\{id_0, id_1\} \cap IDSet = \emptyset$
Then return b' else return 0	Then return b' else return 0

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

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

For prop $\in \{\text{ind}, \text{ano}\}$, we define the advantage $\mathbf{Adv}^{\text{ibe-prop-cpa}}_{\mathcal{IBE},\mathcal{A}}(k)$ of $\mathcal{A}$ in the corresponding experiment as

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

IBE scheme $\mathcal{IBE}$ is said to be IBE-IND-CPA-*secure* (resp., IBE-ANO-CPA-*secure*) if $\mathbf{Adv}^{\text{ibe-ind-cpa}}_{\mathcal{IBE},\mathcal{A}}$ (resp., $\mathbf{Adv}^{\text{ibe-ano-cpa}}_{\mathcal{IBE},\mathcal{A}}$) is a negligible function in k for all polynomial-time adversaries $\mathcal{A}$.

bdop-ibe-2-peks. The bdop-ibe-2-peks transform [7] takes input an IBE scheme $\mathcal{IBE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$ and returns the PEKS scheme $\mathcal{PEKS} = (\mathsf{KG}, \mathsf{Td}, \mathsf{PEKS}, \mathsf{Test})$ where $\mathsf{KG}(1^k) = \mathsf{Setup}(1^k)$, $\mathsf{Td}(sk, w) = \mathsf{KeyDer}(sk, w)$, $\mathsf{PEKS}(pk, w) = \mathsf{Enc}(pk, w, 0^k)$, and $\mathsf{Test}(t_w, C)$ returns 1 iff $\mathsf{Dec}(t_w, C) = 0^k$. Since $\mathcal{BF}\text{-}\mathcal{IBE}$ is anonymous (Theorem 8), since $\mathcal{BDOP}\text{-}\mathcal{PEKS}$ is exactly $\mathcal{BF}\text{-}\mathcal{IBE}$ transformed via bdop-ibe-2-peks, and since $\mathcal{BDOP}\text{-}\mathcal{PEKS}$ is not statistically consistent (Proposition 3), we can conclude that the bdop-ibe-2-peks transformation does not necessarily yield a statistically consistent PEKS scheme. The following theorem strengthens this result, showing that, under the minimal assumption of the existence of an IBE-IND-CPA- and IBE-ANO-CPA-secure IBE scheme, there exists an IBE scheme such that the resulting PEKS scheme via bdop-ibe-2-peks fails to be even computationally consistent.

Theorem 6. *For any* IBE-ANO-CPA*-secure and* IBE-IND-CPA*-secure IBE scheme $\mathcal{IBE}$, there exists another* 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.*

The full proof can be found in the full version [1]; here we sketch the main idea. Given IBE scheme $\mathcal{IBE}$, consider the following IBE scheme $\overline{\mathcal{IBE}}$. The public key includes a normal public key for $\mathcal{IBE}$ and a random string R of length k. A message M is encrypted by encrypting $M\|R$ under $\mathcal{IBE}$. When decrypting a ciphertext C, the result is parsed as $M\|R'$. If $R' = R$, then M is returned as the plaintext, otherwise the decryption algorithm returns 0^k. The resulting Test algorithm will return 1, except with negligible probability, regardless of what trapdoor is being used. This is because the decryption algorithm returns 0^k whenever the last k bits of the plaintext are not equal to R. Intuitively, this should happen with all but negligible probability since, if $\mathcal{IBE}$ is IBE-IND-CPA-secure, a portion of a string encrypted to one identity should not correctly decrypt with the secret key for a different identity.

FIXING THE bdop-ibe-2-peks TRANSFORMATION. The negative result in Theorem 6 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. Our new new-ibe-2-peks transform, instead of always encrypting the same message 0^k, chooses and encrypts a random message R and appends R in the clear to the ciphertext. Thus, given IBE scheme $\mathcal{IBE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$, the PEKS scheme $\mathcal{PEKS} = \text{new-ibe-2-peks}(\mathcal{IBE}) = (\mathsf{KG}, \mathsf{Td}, \mathsf{PEKS}, \mathsf{Test})$ is such that $\mathsf{KG}(1^k) = \mathsf{Setup}(1^k)$, $\mathsf{Td}(sk, w) = \mathsf{KeyDer}(sk, w)$, $\mathsf{PEKS}(pk, w) = (\mathsf{Enc}(pk, w, R), R)$ where $R \stackrel{\$}{\leftarrow} \{0,1\}^k$, and $\mathsf{Test}(t_w, C = (C_1, C_2))$ returns 1 iff $\mathsf{Dec}(t_w, C_1)$ returns 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. We prove the theorem statement below in the full version [1].

Theorem 7. *Let $\mathcal{IBE}$ be an IBE scheme and let $\mathcal{PEKS}$ be the PEKS scheme built 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.*

ANONYMOUS IBE SCHEMES. Theorem 7 motivates a search for IBE-ANO-CPA-secure IBE schemes. The following shows that the Boneh-Franklin BasicIdent IBE scheme, $\mathcal{BF}$-$\mathcal{IBE}$, is anonymous. The proof is simple due to our use of an extension to Halevi's technique for proving the anonymity of public key encryption schemes [15]. This extension and the proof can be found in [1].

Theorem 8. *The $\mathcal{BF}$-$\mathcal{IBE}$ scheme is* IBE-ANO-CPA*-secure in the random oracle model assuming that BDH is hard relative to the underlying pairing generator.*

4 Extensions

We propose three extensions of concepts seen above, namely anonymous HIBEs, public-key encryption with temporary keyword search, and identity-based PEKS.

4.1 Anonymous HIBE

HIBEs. A *hierarchical identity-based encryption* (HIBE) scheme [16,11,6] 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 string ε. 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 (this is ε if $i = 0$). 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 polynomial-time algorithms. Via $(pk, msk = usk[\varepsilon]) \stackrel{\$}{\leftarrow} \mathsf{Setup}(1^k)$, 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 $usk[id] \stackrel{\$}{\leftarrow} \mathsf{KeyDer}^H(usk[\mathrm{par}(id)], id)$ the parent of an identity id with $|id| \geq 1$ can compute a secret key for id; via $C \stackrel{\$}{\leftarrow} \mathsf{Enc}^H(pk, id, M)$ 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. (In constructs we might use multiple random oracles, but since one can always obtain these from a single one [5], definitions will assume just one.) Associated to the scheme is a message space MsgSp where for $\mathsf{MsgSp}(k) \subseteq \{0,1\}^*$ for every $k \in \mathbb{N}$. For consistency, we require that for all $k \in \mathbb{N}$, all identities id with $|id| \geq 1$ and all messages $M \in \mathsf{MsgSp}(k)$, $\Pr[\mathsf{Dec}^H(\mathsf{KeyDer}^H(usk[\mathrm{par}(id)], id), \mathsf{Enc}^H(pk, id, M)) = M] = 1$, where the probability is taken over the choice of $(pk, usk[\varepsilon]) \stackrel{\$}{\leftarrow} \mathsf{Setup}(1^k)$, the

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

$\mathsf{KeyDer}^{H_{1,1},\ldots,H_{1,l}}(usk, id)$
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 $(pk, id, S_{l+1}, Q_1, \ldots, Q_l, s_{l+1})$

$\mathsf{Enc}^{H_{1,1},\ldots,H_{1,l},H_2}(pk, id, m)$
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 $(C_1, \ldots, C_{l+1})$

$\mathsf{Dec}^{H_2}(usk, C)$
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 $C_{l+1} \oplus H_2(\kappa)$

Fig. 3. Algorithms of the $m\mathcal{GS}$-$\mathcal{HIBE}$ scheme $m\mathcal{GS}$-$\mathcal{HIBE}$. $\mathcal{G}$ 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.

random choice of H, and the coins of all the algorithms in the expression above. Unless otherwise stated, it is assumed that $\{0,1\}^k \subseteq \mathsf{MsgSp}(k)$ for all $k \in \mathbb{N}$.

Privacy and anonymity. Let d: $\mathbb{N} \to \mathbb{N}$ be a maximum depth parameter. The notion of privacy, denoted HIBE-IND-CPA[$d(k)$], 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, and all identities id (in queries or challenges) must have $|id| \leq d(k)$. For anonymity, we define the notion of the scheme being anonymous at level $l \geq 1$, denoted HIBE-ANO-CPA[$l, d(k)$]. (Stronger notions are possible, but not needed here.) It is analogous to IBE-ANO-CPA except that the identities returned by the adversary must differ *only* in the l-th component. The adversary can ask the key derivation oracle KeyDer for the secret keys of all identities except for those of the challenge identities or any of their ancestors, and all identities id (in queries or challenges) must have $|id| \leq d(k)$. The definitions are provided in full in [1].

Construction. The HIBE scheme of [16] 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 [11] (here denoted $\mathcal{GS}$-$\mathcal{HIBE}$) is equivalent to the (provably anonymous as per Theorem 8) Boneh-Franklin IBE scheme [8] when restricted to the first level, 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. We suggest a modified version of $\mathcal{GS}$-$\mathcal{HIBE}$ called $m\mathcal{GS}$-$\mathcal{HIBE}$. It is depicted in Fig. 3. The following, proved in [1], implies in particular that $m\mathcal{GS}$-$\mathcal{HIBE}$ is the first full HIBE scheme providing any anonymity at all. The restriction on d is inherited from [11].

Theorem 9. *The $m\mathcal{GS}$-$\mathcal{HIBE}$ scheme is* HIBE-ANO-CPA$[1, d(k)]$*-secure and* HIBE-IND-CPA$[d(k)]$*-secure for any $d(k) = O(\log(k))$ in the random oracle model assuming the BDH problem is hard relative to the generator $\mathcal{G}$.*

4.2 Temporarily Searchable Encryption

PETKS. *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 polynomial-time algorithms 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 $0 \leq s \leq e \leq N(k) - 1$, 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. (In constructs we might use multiple random oracles, but since one can always obtain these from a single one [5], definitions will assume just one.) We require that for all $k \in \mathbb{N}$, all s, e, i with $0 \leq s \leq i \leq e \leq N(k) - 1$, and all $w \in \{0, 1\}^*$, we have $\Pr[\mathsf{Test}^H(\mathsf{Td}^H(sk, w, s, e), \mathsf{PEKS}^H(pk, w, i)) = 1] = 1$, 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. Computational, statistical and perfect consistency can be defined analogously to the way they were defined for PEKS. For details, see [1].

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. A formal definition of the notion of privacy, which we denote PETKS-IND-CPA, is in [1].

Constructions with linear complexity. PETKS is reminiscent of forward-security [4, 9], and, as in these works, there are straightforward solutions with keys of length linear in N. 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 by encrypting under pk_i using the PEKS scheme. The trapdoor for interval $[s..e]$ consists of all PEKS trapdoors 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 $O(N)$ public and secret key length, but still has linear trapdoor length, because the PETKS trapdoor still contains PEKS trapdoors for all time periods $s, \ldots, e$.

A CONSTRUCTION WITH $O(\log(N))$ COMPLEXITY. We now present a transformation hibe-2-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 $\mathcal{PETKS} = \mathsf{hibe\text{-}2\text{-}petks}(\mathcal{HIBE}, t(k)) = (\mathsf{KG}, \mathsf{Td}, \mathsf{PETKS}, \mathsf{Test}, N)$ 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 $C_1 \xleftarrow{\$} \{0,1\}^k$ 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, derives $usk[(w, i_1, \ldots, i_{t(k)})]$ using repetitive calls to the KeyDer algorithm, and returns 1 iff $\mathsf{Dec}(usk[(w, i_1, \ldots, i_{t(k)})], C_2) = C_1$. The proof of the following is in [1]:

Theorem 10. *Let $\mathcal{HIBE}$ be a HIBE scheme, and let $\mathcal{PETKS} = \mathsf{hibe\text{-}2\text{-}petks}(\mathcal{HIBE}, t(k))$ where $t(k) = O(\log(k))$. If $\mathcal{HIBE}$ is* HIBE-ANO-CPA$[1, t(k)+1]$*-secure, then $\mathcal{PETKS}$ is* PETKS-IND-CPA*-secure. Furthermore, if $\mathcal{HIBE}$ is* HIBE-IND-CPA$[t(k)+1]$*-secure, then $\mathcal{PETKS}$ is computationally consistent.*

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 $O(\log N)$ and trapdoors of size $O(\log^2 N)$. 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.

4.3 ID-based Searchable Encryption

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).

Like in IBE schemes, this allows to use any string as a recipient's public key for the PEKS scheme.

IBEKS. An identity-based encryption with keyword search scheme $\mathcal{IBEKS}$ = $(\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Td}, \mathsf{Enc}, \mathsf{Test})$ is made up of five algorithms. Via $(pk, msk) \stackrel{\$}{\leftarrow} \mathsf{Setup}(1^k)$, where $k \in \mathbb{N}$ is the security parameter, the randomized setup algorithm produces master keys; via $usk[id] \stackrel{\$}{\leftarrow} \mathsf{KeyDer}^H(msk, id)$, the master computes the secret key for identity id; via $C \stackrel{\$}{\leftarrow} \mathsf{Enc}^H(pk, id, w)$, a sender encrypts a keyword w to identity id to get a ciphertext; via $t_w \stackrel{\$}{\leftarrow} \mathsf{Td}^H(usk[id], w)$, 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 $w \in \{0,1\}^*$, we have $\Pr[\mathsf{Test}^H(\mathsf{Td}^H(\mathsf{KeyDer}^H(msk, id), w), \mathsf{Enc}^H(pk, id, w)) = 1] = 1$, where the probability is taken over the choice of $(pk, msk) \stackrel{\$}{\leftarrow} \mathsf{Setup}(1^k)$, the random choice of H, and the coins of all algorithms in the expression above.

Consistency and privacy. Computational, statistical and perfect consistency can be defined analogously to the way they were defined for PEKS. A privacy notion (denoted IBEKS-IND-CPA) can be obtained by appropriately combining ideas of the definitions of privacy for PEKS and IBE. For details, see [1].

Construction. We now propose a generic transformation, called $\mathsf{hibe\text{-}2\text{-}ibeks}$, to convert any HIBE scheme with two levels into an IBEKS scheme. Given a HIBE scheme $\mathcal{HIBE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$ with two levels, $\mathsf{hibe\text{-}2\text{-}ibeks}$ returns the IBEKS scheme $\mathcal{IBEKS} = (\mathsf{Setup}, \overline{\mathsf{KeyDer}}, \overline{\mathsf{Enc}}, \mathsf{Td}, \mathsf{Test})$ such that $\overline{\mathsf{KeyDer}}(msk, id) = (usk, id)$ where $usk \stackrel{\$}{\leftarrow} \mathsf{KeyDer}(msk, id)$, $\overline{\mathsf{Enc}}(pk, id, w) = (C_1, C_2)$ where $C_1 \stackrel{\$}{\leftarrow} \{0,1\}^k$ and $C_2 = \mathsf{Enc}(pk, (id, w), C_1)$, $\mathsf{Td}(\overline{usk} = (usk, id), w) = \mathsf{KeyDer}(usk, (id, w))$ and $\mathsf{Test}(t_w, (C_1, C_2))$ returns 1 iff $\mathsf{Dec}(t_w, C_2) = C_1$. The proof of the following is in [1]:

Theorem 11. *Let $\mathcal{HIBE}$ be a HIBE scheme and $\mathcal{IBEKS} = \mathsf{hibe\text{-}2\text{-}ibeks}(\mathcal{HIBE})$. 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.*

We know of no HIBE scheme that is anonymous at level 2, and thus we have no concrete instantiations of the above. (We exclude the scheme of [16] because it is not secure against a polynomial number of level-2 key extractions, as required for HIBE-ANO-CPA[2,2]-security and in particular for our construction.)

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. M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi. Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions. Full version of current paper. Available at IACR Cryptology ePrint Archive, `http://eprint.iacr.org`.
2. M. Ajtai and C. Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *29th ACM STOC*. ACM Press, 1997.
3. M. Bellare, A. Boldyreva, A. Desai, and D. Pointcheval. Key-privacy in public-key encryption. In *ASIACRYPT 2001, LNCS* 2248. Springer, 2001.
4. M. Bellare and S. K. Miner. A forward-secure digital signature scheme. In *CRYPTO'99, LNCS* 1666. Springer, 1999.
5. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS 93*. ACM Press, 1993.
6. D. Boneh, X. Boyen, and E.-J. Goh. Hierarchical identity based encryption with constant size ciphertext. In *EUROCRYPT 2005, LNCS* 3494. Springer, 2005.
7. D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. In *EUROCRYPT 2004, LNCS* 3027. Springer, 2004.
8. D. Boneh and M. K. Franklin. Identity based encryption from the Weil pairing. *SIAM Journal on Computing*, 32(3), 2003.
9. R. Canetti, S. Halevi, and J. Katz. A forward-secure public-key encryption scheme. In *EUROCRYPT 2003, LNCS* 2656. Springer, 2003.
10. C. Dwork, M. Naor, and O. Reingold. Immunizing encryption schemes from decryption errors. In *EUROCRYPT 2004, LNCS* 3027. Springer, 2004.
11. C. Gentry and A. Silverberg. Hierarchical ID-based cryptography. In *ASIACRYPT 2002, LNCS* 2501. Springer, 2002.
12. E.-J. Goh. Secure indexes. Cryptology ePrint Archive, Report 2003/216, 2003.
13. O. Goldreich. *Foundations of Cryptography: Basic Applications*, volume 2. Cambridge University Press, Cambridge, UK, 2004.
14. P. Golle, J. Staddon, and B. R. Waters. Secure conjunctive keyword search over encrypted data. In *ACNS 2004, LNCS* 3089. Springer, 2004.
15. S. Halevi. A sufficient condition for key-privacy. Cryptology ePrint Archive, Report 2005/005, 2005.
16. J. Horwitz and B. Lynn. Toward hierarchical identity-based encryption. In *EUROCRYPT 2002, LNCS* 2332. Springer, 2002.
17. A. Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO'84, LNCS* 196. Springer, 1985.
18. D. X. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In *2000 IEEE Symposium on Security and Privacy*, 2000.
19. B. R. Waters. Efficient identity-based encryption without random oracles. In *EUROCRYPT 2005, LNCS* 3494. Springer, 2005.
20. B. R. Waters, D. Balfanz, G. Durfee, and D. K. Smetters. Building an encrypted and searchable audit log. In *ISOC NDSS 2004*, 2004.