Blind and Anonymous Identity-Based Encryption and Authorised Private Searches on Public Key Encrypted Data

Jan Camenisch[1], Markulf Kohlweiss[2], Alfredo Rial[2], and Caroline Sheedy[3]

[1] Zurich Research Lab
IBM Research
jca@zurich.ibm.com
[2] ESAT-COSIC / IBBT
Katholieke Universiteit Leuven
{markulf.kohlweiss,alfredo.rialduran}@esat.kuleuven.be
[3] School of Computing
Dublin City University
csheedy@computing.dcu.ie

Abstract. Searchable encryption schemes provide an important mechanism to cryptographically protect data while keeping it available to be searched and accessed. In a common approach for their construction, the encrypting entity chooses one or several keywords that describe the content of each encrypted record of data. To perform a search, a user obtains a trapdoor for a keyword of her interest and uses this trapdoor to find all the data described by this keyword.

We present a searchable encryption scheme that allows users to privately search by keywords on encrypted data in a public key setting and decrypt the search results. To this end, we define and implement two primitives: public key encryption with *oblivious* keyword search (PEOKS) and *committed blind anonymous* identity-based encryption (IBE). PEOKS is an extension of public key encryption with keyword search (PEKS) in which users can obtain trapdoors from the secret key holder without revealing the keywords. Furthermore, we define committed blind trapdoor extraction, which facilitates the definition of authorisation policies to describe which trapdoor a particular user can request. We construct a PEOKS scheme by using our other primitive, which we believe to be the first blind and anonymous IBE scheme.

We apply our PEOKS scheme to build a public key encrypted database that permits authorised private searches, i.e., neither the keywords nor the search results are revealed.

Keywords: Blind identity-based encryption, searchable encryption, public key encryption with keyword search.

1 Introduction

Vast quantities of sensitive personal data are retained for the purpose of network forensics and cyber investigations [1]. The advantages of the availability of such data for the investigation of serious crimes and the protection of national security are considerable.

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-642-00468-1_29

S. Jarecki and G. Tsudik (Eds.): PKC 2009, LNCS 5443, pp. 196–214, 2009.

However, these advantages must be counterpoised by the dangers that such data could fall into the wrong hands.

The encryption of retained data is a desirable counter measure against data theft. But how, then, can the investigator, such as the police or a secret service, search the data without having to decrypt the whole database? What if the investigator should only be given access to data that fulfills certain criteria? This seems to be a hard problem, as the criteria themselves may be sensitive and thus requiring protective measures, such as encryption. Moreover, a secret service is often reluctant to reveal the type of queries it wants to run on the encrypted database.

We consider a scenario in which an investigator searches for data described by multiple keywords without revealing the keywords or the search results to the database server. This scenario is akin to the private searching of streaming data presented in [2]. While in [2] the data is searched as it is generated (and can thereafter be discarded), in our scenario data is first stored in encrypted form and can be searched at a later stage. To provide a high level of security we make use of asymmetric cryptography. The database server only possesses the public encryption key (and cannot decrypt the retained data itself). In this way, data that is already encrypted remains secure even against a strong adversary that breaks into the database server. The decryption key is stored by a security server, which will only be involved when executing search queries.

As the details of queries made are to be obscured even from the security server, it is necessary to impose some restrictions on the investigator. Thus we introduce some checks and balances to avoid abuse by overzealous or malicious investigators. One obvious restriction is in the number of queries that the investigator can make. An unreasonable number of requests may be an indication of abuse. Another restriction that we consider is to involve a judge in granting search warrants to the investigator. The keyword is still hidden, but the security server is guaranteed that a judge (or another authority figure) has approved the search for a specific keyword.

In [3] the authors build an encrypted and searchable audit log. They propose two schemes, one based on symmetric encryption and one based on asymmetric encryption. They conclude that asymmetric encryption provides better security, as it reduces the trust in the encrypting entity. Our work can be seen as an extension of their asymmetric scheme with the possibility to obliviously search the encrypted database. For the symmetric case, in which the audit log server knows all the information needed for decrypting the database, the problem of performing oblivious searches is covered by [4,5]. The problem of oblivious searching on public key encrypted data is more difficult.

Outline of our solution. In [3], the asymmetric searchable encryption scheme is based on identity-based encryption (IBE) [6]. The keywords themselves are used to encrypt the database, i.e., they are the identity strings of the IBE scheme. The anonymity property of Boneh-Franklin IBE scheme [6] ensures that a ciphertext does not leak the identity string used to generate the encryption. The security server holds the master secret key that is used to derive the secret keys corresponding to the keywords that are needed for searching. A similar technique for searchable encryption was formalized as public key encryption with keyword search (PEKS) by [7]. In PEKS, the derived keys are referred to as search trapdoors, which can be given to third parties to grant them search rights.

When trying to build an oblivious search mechanism for such a database we have to address two difficulties: hiding the keywords from the security server and hiding the search results from the database. For the former, we present two new cryptographic primitives. The first one is *committed blind anonymous* IBE. In this context, *anonymous* means that the ciphertext does not leak the key (identity) under which it was encrypted [8,9] and *blind* means that a user can request the decryption key for a given identity without the key generation entity learning the identity [10]. The work of [10] describes how to construct blind key derivation protocols for [11] and [12,13], but these schemes are not anonymous. Moreover, it is much harder to derive a blind key derivation protocol for the Boneh-Franklin IBE scheme [6] used in [3], and we are interested in IBE schemes that do not require random oracles for their security proofs. (As shown by [14,15,16], a scheme may be insecure even if proven secure in the random oracle model.) As a corollary to our results, we obtain the first instantiation of [3] secure without random oracles.

We design a committed blind anonymous IBE scheme based on the anonymous IBE scheme due to [9]. As the scheme in [9] is only selective ID secure [11], we extend it with adaptive ID security [17] and prove the modified scheme secure. For the modified scheme we design a blind key extraction protocol. This leads to the first blind anonymous IBE scheme we are aware of. We extend the definition of blind IBE to allow for the derivation of a secret key for a committed identity. This allows the key generation entity to enforce authorisation policies on the identities for which a secret key is requested, as described in [18].

The second primitive we present is public key encryption with oblivious keyword search (PEOKS), which we implement using our committed blind anonymous IBE scheme. First, we extend the definition of PEKS to incorporate the encryption of a secret message when computing a searchable encryption. This secret message can contain a symmetric key, which allows PEKS to be used directly in settings such as [3]. Then we define blind key extraction with committed keywords, which facilitates the use of a policy that states for which keywords a trapdoor can be extracted while still keeping them hidden from the trapdoor generation entity.

In order to hide the search results from the database one could in theory download the whole database and then use PEOKS to do the search. This is inefficient. We describe a data structure that allows to use private information retrieval (PIR) [19] to improve the communication efficiency of the search.

Our contribution. We define and construct the first blind anonymous IBE scheme. We generalize PEKS to be usable in settings such as [3], and we extend it to incorporate the facility to perform oblivious keywords searches (PEOKS). Both our blind anonymous IBE scheme and our PEOKS scheme support committed blind key extraction and thus allow for complex policies. Finally, we describe the first public key encrypted database that allows for oblivious searches, i.e., both the keywords and the search results remain hidden.

Outline of the paper. In Sect. 2 we introduce basic concepts and security assumptions and in Sect. 3 we define committed blind anonymous IBE and PEOKS. We construct a committed blind anonymous IBE scheme and we show how to apply it to build a PEOKS scheme in Sect. 4. In Sect. 5, we describe the use of PEOKS to construct a

privacy-preserving searchable encrypted database. Finally, Sect. 6 draws a conclusion and discusses future work.

2 Technical Preliminaries

A function ν is *negligible* if, for every integer c, there exists an integer K such that for all $k > K$, $|\nu(k)| < 1/k^c$. A problem is said to be *hard* (or *intractable)* if there exists no probabilistic polynomial time (p.p.t.) algorithm on the size of the input to solve it. ϵ denotes the empty string.

Bilinear Maps. Let G_1, G_2 and G_T be groups of prime order p. A map $e : G_1 \times G_2 \rightarrow G_T$ must satisfy the following properties:

(a) *Bilinearity*. A map $e : G_1 \times G_2 \rightarrow G_T$ is bilinear if $e(a^x, b^y) = e(a, b)^{xy}$;
(b) *Non-degeneracy*. For all generators $g \in G_1$ and $h \in G_2$, $e(g, h)$ generates G_T;
(c) *Efficiency*. There exists an efficient algorithm $\mathsf{BMGen}(1^k)$ that outputs $(p, G_1, G_2, G_T, e, g, h)$ to generate the bilinear map and an efficient algorithm to compute $e(a, b)$ for any $a \in G_1, b \in G_2$.

The security of our scheme is based on the following number-theoretic assumptions:

Definition 1 (Decision BDH). *Given $g, g^a, g^b, g^c \in G_1$, $h, h^a, h^b \in G_2$, and $Z \in G_T$ for random exponents $a, b, c \in Z_p$, decide whether $Z = e(g, h)^{abc}$ or a random element from G_T. The Decision BDH assumption holds if all p.p.t algorithms have negligible advantage in solving the above problem.*

Definition 2 (Decision Linear). *Given $g, g^a, g^b, g^{ac}, g^{bd}, Z \in G_1$, $h, h^a, h^b \in G_2$ for random exponents $a, b, c, d \in Z_p$, decide whether $Z = g^{c+d}$ or a random element in G_1. The Decision Linear assumption holds if all p.p.t algorithms have negligible advantage in solving the above problem.*

Commitment Schemes. A *commitment scheme* is a two phase scheme that allows a user to *commit* to a hidden value, while preserving the ability of the user to *reveal* the committed value at a later stage. The properties of a commitment scheme are *hiding*: the value committed to must remain undiscovered until the reveal stage, and *binding*: the only value which may be revealed is the one that was chosen in the commit stage.

We use the perfectly hiding commitment scheme proposed by Pedersen [20]: Given a group G of prime order p with generators g and h, generate a commitment C to $x \in Z_p$ by choosing at random $open_x \leftarrow Z_p$ and computing $C = g^x h^{open_x}$. The commitment is opened by revealing x and $open_x$.

Proofs of Knowledge. We use several existing results to prove statements about discrete logarithms; (1) proof of knowledge of a discrete logarithm modulo a prime [21], (2) proof of knowledge of the equality of some element in different representations [22], (3) proof that a commitment opens to the product of two other committed values [23,24,25], and (4) proof of the disjunction or conjunction of any two of the previous [26]. These results are often given in the form of Σ-protocols but they can be turned into zero-knowledge protocols using efficient zero-knowledge compilers [27,28].

When referring to the proofs above, we follow the notation introduced by Camenisch and Stadler [29] for various proofs of knowledge of discrete logarithms and proofs of the validity of statements about discrete logarithms.

$$PK\{(\alpha, \beta, \delta) : y = g^{\alpha} h^{\beta} \wedge \tilde{y} = \tilde{g}^{\alpha} \tilde{h}^{\delta}\}$$

denotes a "*zero-knowledge Proof of Knowledge of integers* α, β, *and* δ *such that* $y = g^{\alpha} h^{\beta}$ *and* $\tilde{y} = \tilde{g}^{\alpha} \tilde{h}^{\delta}$ *holds*", where $y, g, h, \tilde{y}, \tilde{g}$, and $\tilde{h}$ are elements of some groups $G = \langle g \rangle = \langle h \rangle$ and $\tilde{G} = \langle \tilde{g} \rangle = \langle \tilde{h} \rangle$ that have the same order. (Note that some elements in the representation of y and $\tilde{y}$ are equal.) The convention is that letters in the parenthesis, in this example α, β, and δ, denote quantities whose knowledge is being proven, while all other values are known to the verifier. There exists a knowledge extractor which can extract these quantities from a successful prover.

3 Definitions of Committed Blind Anonymous IBE and PEOKS

3.1 Anonymous Identity-Based Encryption

We recall the definition of identity-based encryption [6]. An IBE scheme Π consists of the algorithms $(\mathsf{IBESetup}, \mathsf{IBEExtract}, \mathsf{IBEEnc}, \mathsf{IBEDec})$:

$\mathsf{IBESetup}(1^k)$ outputs parameters $params$ and master secret key msk.
$\mathsf{IBEExtract}(params, msk, id)$ outputs the secret key sk_{id} for identity id.
$\mathsf{IBEEnc}(params, id, m)$ outputs ct encrypting m under id.
$\mathsf{IBEDec}(params, sk_{id}, ct)$ outputs message m encrypted in ct.

An IBE scheme is *anonymous* [30], if it is not possible to associate the identity id used to encrypt a message m with the resulting ciphertext ct (in the context of public key encryption this is also known as key privacy [31]).

Abdalla et al. [30] define anonymity through a security game in which the adversary receives a ciphertext encrypted with an identity that is randomly picked from two identities of his choosing. The adversary has to guess the identity used to encrypt the ciphertext. As in [8], we combine this game with the standard chosen plaintext security game for IBE in which the adversary needs to guess which message out of two possible messages was encrypted.[1]

Definition 3 (Secure Anonymous IBE [30]). *Let* k *be a security parameter. An anonymous IBE scheme* Π *is secure if every p.p.t. adversary* $\mathcal{A}$ *has an advantage negligible in* k *in the following game:*

Setup. *The game runs* $\mathsf{IBESetup}(1^k)$ *to generate* $(params, msk)$.

Phase 1. $\mathcal{A}$ *may query an oracle* $\mathcal{O}_{\mathsf{IBEExtract}}(params, msk, id)$ *polynomially many times with input* id. *The oracle then runs* $\mathsf{IBEExtract}(params, msk, id)$ *and returns the associated* sk_{id}.

Challenge. $\mathcal{A}$ *presents the simulator two target identities* id_0, id_1, *which have not*

[1] We define an adaptive identity security game as this is required by our IBE to PEOKS transformation.

been queried in Phase 1, and two challenge messages m_0, m_1. *The simulator selects two random bits* b_1 *and* b_2, *and returns to* $\mathcal{A}$ *the challenge ciphertext* $ct = \mathsf{IBEEnc}(params, id_{b_1}, m_{b_2})$.

Phase 2. $\mathcal{A}$ *may again query oracle* $\mathcal{O}_{\mathsf{IBEExtract}}(params, msk, id)$ *polynomially many times on* id *provided* id *is not* id_0 *or* id_1.

Guess. $\mathcal{A}$ *outputs* b'_1, b'_2. *We define the advantage of* $\mathcal{A}$ *as* $|Pr[b'_1 = b_1 \wedge b'_2 = b_2] - 1/4|$.

3.2 Committed Blind Anonymous IBE

In standard IBE schemes, a key generation centre $\mathcal{KGC}$ executes the *key extraction* algorithm IBEExtract that returns the secret key sk_{id} corresponding to input identity id. Green and Hohenberger [10] propose extracting the secret key in a *blinded* manner. The blinding action obscures the identity from the $\mathcal{KGC}$. We extend this concept by proposing a *committed blind anonymous* IBE scheme, where the $\mathcal{KGC}$ is given a commitment to the requested identity. A user can reveal partial information about the identity or prove statements about it using efficient zero-knowledge proofs about commitments [32,25].[2]

A committed blind anonymous IBE scheme consists of the algorithms Π of an IBE scheme, a secure commitment scheme Commit, and the protocol IBEBlindExtract:

$\mathsf{IBEBlindExtract}(\mathcal{U}(params, id, open_{id}), \mathcal{KGC}(params, msk, C))$ generates the secret decryption key sk_{id} for $\mathcal{U}$'s identity id in an interactive key issuing protocol between $\mathcal{U}$ and the $\mathcal{KGC}$. If $C = \mathsf{Commit}(id, open_{id})$, $\mathcal{U}$'s output is a decryption key sk_{id} and the output of the $\mathcal{KGC}$ is empty. Otherwise both parties output $\perp$.

Green and Hohenberger [10] construct a security argument for blind-IBE by defining two properties for the IBEBlindExtract protocol: leak freeness and selective-failure blindness. Leak freeness requires that IBEBlindExtract is a secure two-party computation that does not leak any more information than IBEExtract.[3] Selective-failure blindness requires that a potentially malicious authority does not learn anything about the user's identity during the IBEBlindExtract protocol. Additionally, it cannot cause the IBEBlindExtract protocol to selectively fail depending on the user's choice of identity. We provide adapted versions of these properties for committed blind anonymous IBE.

Definition 4 (Leak Freeness [10]). *An* IBEBlindExtract *protocol of an* IBE *scheme is leak free if, for all efficient adversaries* $\mathcal{A}$, *there exists an efficient simulator* $\mathcal{S}$ *such that for every value* k, *no efficient distinguisher* $\mathcal{D}$ *can determine whether it is playing Game Real or Game Ideal with non-negligible advantage, where*

Game Real: *Run* $\mathsf{IBESetup}(1^k)$. *As many times as* $\mathcal{D}$ *wants, he picks a commitment* C *and* $\mathcal{A}$*'s input* $state$. $\mathcal{A}$ *runs* $\mathsf{IBEBlindExtract}(\mathcal{A}(params, state), \mathcal{KGC}(params, msk, C))$ *with the* $\mathcal{KGC}$. $\mathcal{A}$ *returns the resulting* $view$ *to* D.

[2] Technically this can be seen as restricting the blind key derivation queries to a certain language, membership of which is proven in zero-knowledge.

[3] It also implies that the user is required 'to know' the id for which she needs a key to be extracted. We also require that she knows the opening to the commitment.

Game Ideal: *Run* $\mathsf{IBESetup}(1^k)$. *As many times as* $\mathcal{D}$ *wants, he picks a commitment* C *and initial input* $state$. $\mathcal{S}$ *obtains* $(params, state)$ *and may choose values* id *and* $open_{id}$ *to query an oracle* $\mathcal{O}_{\mathsf{IBEExtract}}$ *that knows* msk *and is parameterized with* C. *If* $C = \mathsf{Commit}(id, open_{id})$, *the oracle returns key* $sk_{id} \leftarrow \mathsf{IBEExtract}(params, msk, id)$, *otherwise* $\perp$. $\mathcal{S}$ *returns a simulated view to* $\mathcal{D}$.

Definition 5 (Selective-Failure Blindness [33]). *An* IBEBlindExtract *protocol is said to be selective-failure blind if every adversary* $\mathcal{A}$ *has a negligible advantage in the following game:* $\mathcal{A}$ *outputs* $params$ *and a pair of identities* id_0, id_1. *A random bit* $b \in \{0,1\}$ *is chosen, and* $\mathcal{A}$ *is given two fresh commitments* C_b, C_{1-b} *and black-box access to two oracles:* $\mathcal{U}(params, id_b, open_{id_b})$ *and* $\mathcal{U}(params, id_{1-b}, open_{id_{1-b}})$. *The* $\mathcal{U}$ *algorithms produce* sk_b, sk_{1-b} *respectively. If* $sk_b \neq \perp$ *and* $sk_{1-b} \neq \perp$, $\mathcal{A}$ *receives* (sk_0, sk_1); *if only* $sk_{1-b} = \perp$, $(\epsilon, \perp)$; *if only* $sk_b = \perp$, $(\perp, \epsilon)$; *and if* $sk_b = sk_{1-b} = \perp$, $\mathcal{A}$ *receives* $(\perp, \perp)$. *Finally,* $\mathcal{A}$ *outputs his guess* b'. *The advantage of* $\mathcal{A}$ *in this game is* $|Pr[b' = b] - 1/2|$.

Following [10], we define a secure committed blind anonymous IBE as follows.

Definition 6 (Secure Committed Blind Anonymous IBE). *A committed blind anonymous IBE scheme* $(\Pi, \mathsf{IBEBlindExtract}, \mathsf{Commit})$ *is secure if and only if: (1) The underlying* Π *is a secure anonymous IBE scheme, (2)* Commit *is a secure commitment scheme, and (3)* IBEBlindExtract *is leak free and selective-failure blind.*

3.3 Public Key Encryption with Oblivious Keyword Search

We recall and extend the definition of PEKS [7]. A PEKS scheme $\Upsilon = (\mathsf{KeyGen}, \mathsf{PEKS}, \mathsf{Trapdoor}, \mathsf{Test})$ consists of the algorithms:

$\mathsf{KeyGen}(1^k)$ outputs a public key A_{pub} and secret key A_{priv}.
$\mathsf{PEKS}(A_{pub}, W, m)$ outputs a searchable encryption S_W of m under keyword W.
$\mathsf{Trapdoor}(A_{pub}, A_{priv}, W)$ outputs a trapdoor T_W that allows to search for the keyword W.
$\mathsf{Test}(A_{pub}, S_W, T_{W'})$ outputs the message m encoded in S_W, if $W = W'$; otherwise it outputs $\perp$.

This definition of PEKS extends the standard definition [7] by encoding a secret m into the PEKS element S_W generated by the PEKS algorithm. Test outputs this secret when a match occurs.

A secure PEKS scheme must be *chosen plaintext attack* (CPA) secure and *consistent* [30]. CPA security requires that an attacker cannot distinguish two PEKS elements generated for keywords and messages of his choice, even if given oracle access to Trapdoor for other keywords. Consistency requires that if the searchable encryption and the trapdoor are computed using different keywords, then algorithm Test should output $\perp$ upon such input.

In PEKS, the party holding the secret key A_{priv} runs the Trapdoor algorithm to obtain the trapdoor T_W for a keyword W. Public key encryption with oblivious keyword search (PEOKS) is an extension of PEKS in which a user $\mathcal{U}$ performing a search can obtain in a *committed* and *blinded* manner the trapdoor T_W from the trapdoor generation entity $\mathcal{TGE}$. The $\mathcal{TGE}$ only learns a commitment to the search term.

A PEOKS scheme consists of the algorithms Υ of a PEKS scheme, a secure commitment scheme Commit used to commit to keywords, and the following BlindTrapdoor protocol:

BlindTrapdoor$(\mathcal{U}(A_{pub}, W, open_W), \mathcal{TGE}(A_{pub}, A_{priv}, C))$ generates a trapdoor T_W for a keyword W in an interactive protocol between $\mathcal{U}$ and $\mathcal{TGE}$. If C=Commit$(W, open_W)$, $\mathcal{U}$'s output is the trapdoor T_W and the output of $\mathcal{TGE}$ is empty. Otherwise both parties output $\perp$.

Leak freeness and *selective-failure blindness* can be defined for BlindTrapdoor following the definition for IBEBlindExtract.[4] We define the security of PEOKS similarly to that of a committed blind anonymous IBE scheme, assuming a secure underlying PEKS scheme.

Definition 7 (Secure PEOKS). *A* PEOKS *scheme (*Υ*,* BlindTrapdoor*,* Commit*) is secure if and only if: (1) The underlying* Υ *is a secure PEKS scheme, (2)* Commit *is a secure commitment scheme, and (3)* BlindTrapdoor *is leak free and selective-failure blind.*

4 Construction of a Committed Blind Anonymous IBE Scheme and a Transformation to PEOKS

4.1 The Underlying Anonymous IBE Scheme

We present an anonymous IBE scheme that is adaptive identity secure in the standard model, based on the anonymous IBE scheme proposed by Boyen-Waters [9]. The Boyen-Waters scheme is selective identity secure. We use a transformation due to Naccache [12], a variant of that of Waters [17], to achieve the required adaptive identity security. The use of such a transformation was proposed by Boyen-Waters [9]. We provide what we believe to be the first proof of security for this variant. Our scheme supports asymmetric bilinear maps, allowing the use of a wider range of potentially more efficient implementations using different pairing types [34]. Let identity $id \in \{0,1\}^{\ell \times n}$ and let $id_1 \| \ldots \| id_n = id$ be the separation of id into ℓ bit integers id_i. Let $H_1(id) = g_0 \prod_{i=1}^{n} g_i^{id_i}$ and $H_2(id) = h_0 \prod_{i=1}^{n} h_i^{id_i}$. Our anonymous IBE scheme Π = (IBESetup, IBEExtract, IBEEnc, IBEDec) consists of the following algorithms :

IBESetup(1^k). Run BMGen(1^k) to obtain a bilinear map setup $(p, G_1, G_2, G_T, e, g, h)$. Choose values $\alpha, z_0, z_1, \ldots, z_n, t_1, t_2, t_3, t_4 \leftarrow Z_p^*$ and keep $msk = (\alpha, t_1, t_2, t_3, t_4)$ as the master key. Compute the system parameters as

$$params = \Big(\Omega = e(g,h)^{t_1 t_2 \alpha}, g, h, g_0 = g^{z_0}, \ldots, g_n = g^{z_n}, v_1 = g^{t_1}, \ldots, v_4 = g^{t_4}, h_0 = h^{z_0}, \ldots, h_n = h^{z_n}\Big).$$

[4] The inputs are mapped 1-to-1. KeyGen is used instead of IBESetup and Trapdoor instead of IBEExtract.

$\mathsf{IBEExtract}(params, msk, id)$. Choose two random values $\tilde{r}_1, \tilde{r}_2 \leftarrow Z_p^*$ and compute the key

$$sk_{id} = \Big(h^{\tilde{r}_1 t_1 t_2 + \tilde{r}_2 t_3 t_4}, h^{-\alpha t_2} H_2(id)^{-\tilde{r}_1 t_2}, h^{-\alpha t_1} H_2(id)^{-\tilde{r}_1 t_1}, H_2(id)^{-\tilde{r}_2 t_4}, H_2(id)^{-\tilde{r}_2 t_3}\Big) .$$

$\mathsf{IBEEnc}(params, id, msg)$. To encrypt a message $msg \in G_T$, choose $s, s_1, s_2 \leftarrow Z_p$, and generate the ciphertext

$$ct = \Big(\Omega^s \cdot msg, H_1(id)^s, v_1^{s-s_1}, v_2^{s_1}, v_3^{s-s_2}, v_4^{s_2}\Big).$$

$\mathsf{IBEDec}(params, sk_{id}, ct)$. Parse sk_{id} as $(d_0, d_1, d_2, d_3, d_4)$ and ct as $(c', c_0, c_1, c_2, c_3, c_4)$ and return

$$msg = c' \cdot e(c_0, d_0) \cdot e(c_1, d_1) \cdot e(c_2, d_2) \cdot e(c_3, d_3) \cdot e(c_4, d_4).$$

Theorem 1. *The scheme Π is an adaptive identity secure anonymous IBE scheme under the DBDH and DLIN assumptions.* Please see the full version for the proof.

4.2 Blind Extraction Protocol

We introduce an interactive blind key extraction protocol IBEBlindExtract, which extends algorithm IBEExtract.

Intuition behind our construction. Generating a randomly distributed secret key by means of the IBEBlindExtract protocol requires the values $\tilde{r}_1, \tilde{r}_2$ to be jointly chosen by the user and the key issuer in a manner which prevents either party from learning anything about the other's randomness. This prevents a user that learns the issuer's randomness from potentially decrypting messages of other users and an issuer that learns a user's randomness from potentially breaking the blindness of the key issued.

The key issuer, $\mathcal{KGC}$, chooses random values $\hat{r}_1, \hat{r}_2 \leftarrow Z_p^*$, and the user $\mathcal{U}$ picks random values $r_1', r_2' \leftarrow Z_p^*$. The key generation protocol may be implemented using standard secure two-party computation techniques [35], as a protocol in which the user inputs r_1', r_2' and the $\mathcal{KGC}$ inputs $\alpha, t_1, t_2, t_3, t_4, \hat{r}_1, \hat{r}_2$. The user's output in the protocol is a secret key

$$sk_{id} = (h^{\tilde{r}_1 t_1 t_2 + \tilde{r}_2 t_3 t_4}, h^{-\alpha t_2} H_2(id)^{-\tilde{r}_1 t_2}, h^{-\alpha t_1} H_2(id)^{-\tilde{r}_1 t_1}, H_2(id)^{-\tilde{r}_2 t_4}, H_2(id)^{-\tilde{r}_2 t_3}) ,$$

with $\tilde{r}_1 = \hat{r}_1 r_1'$ and $\tilde{r}_2 = \hat{r}_2 r_2'$. The $\mathcal{KGC}$ learns nothing further, and outputs nothing. By decomposing this protocol into sub-protocols, whose results only require simple arithmetic operations (addition and multiplication), we obtain an efficient protocol.

Construction. Our committed blind anonymous IBE scheme consists of the algorithms Π of the underlying IBE scheme, the Pedersen commitment scheme Commit, and the following IBEBlindExtract protocol:

$\mathsf{IBEBlindExtract}(\mathcal{U}(params, id, open_{id}) \leftrightarrow \mathcal{KGC}(params, msk, C))$.

1. $\mathcal{KGC}$ chooses at random $\hat{r}_1, \hat{r}_2 \leftarrow Z_p^*$, and the user $\mathcal{U}$ chooses at random $u_0, u_1, u_2 \leftarrow Z_p$ and $u_3, r_1', r_2' \leftarrow Z_p^*$. Implicitly, $\tilde{r}_1 = \hat{r}_1 r_1'$ and $\tilde{r}_2 = \hat{r}_2 r_2'$. $\mathcal{U}$ computes $C_{u_3} = \mathsf{Commit}(u_3, open_{u_3})$, and $\mathcal{KGC}$ computes $C_{\hat{r}_1} = \mathsf{Commit}(\hat{r}_1, open_{\hat{r}_1})$ and $C_{\hat{r}_2} = \mathsf{Commit}(\hat{r}_2, open_{\hat{r}_2})$. $\mathcal{KGC}$ and $\mathcal{U}$ make use of a two-party protocol for simple arithmetics modulo p (parameterized by C_{u_3}, $C_{\hat{r}_1}$, and $C_{\hat{r}_2}$). $\mathcal{U}$ inputs $u_0, u_1, u_2, u_3, open_{u_3}, r_1', r_2'$ and $\mathcal{KGC}$ inputs $\alpha, t_1, t_2, t_3, t_4, \hat{r}_1, open_{\hat{r}_1}, \hat{r}_2, open_{\hat{r}_2}, open_{x_0}, open_{x_1}, open_{x_2}$. If $C_{u_3} = \mathsf{Commit}(u_3, open_{u_3})$, $C_{\hat{r}_1} = \mathsf{Commit}(\hat{r}_1, open_{\hat{r}_1})$, and $C_{\hat{r}_2} = \mathsf{Commit}(\hat{r}_2, open_{\hat{r}_2})$ the output of $\mathcal{KGC}$ is

$$\begin{aligned} x_0 &= (\hat{r}_1 r_1' t_1 t_2 + \hat{r}_2 r_2' t_3 t_4) + u_0 \pmod p\,, \\ x_1 &= -(u_3 / r_1' \cdot \alpha t_2) + u_1 \pmod p\,, \\ x_2 &= -(u_3 / r_1' \cdot \alpha t_1) + u_2 \pmod p. \end{aligned}$$

   Provided that $\mathcal{KGC}$ does not abort at that moment, $\mathcal{U}$ obtains $C_{x_0} = \mathsf{Commit}(x_0, open_{x_0})$, $C_{x_1} = \mathsf{Commit}(x_1, open_{x_1})$ and $C_{x_2} = \mathsf{Commit}(x_2, open_{x_2})$ as output. Otherwise, both parties output $\perp$. In Sect. 4.3 we show how to efficiently realise such a protocol.
2. $\mathcal{U}$ computes $ID' = H_2(id)^{u_3}$, where u_3 is a blinding value, and sends ID' to $\mathcal{KGC}$. $\mathcal{U}$ proves that the identity in ID' corresponds to C_{id} and that ID' is well-formed using C_{u_3}. $\mathcal{KGC}$ returns $\perp$ if the proof fails. Details about this proof of knowledge can be found in Appendix A.
3. $\mathcal{KGC}$ computes

$$sk_{id}{}' = (h^{x_0}, h^{x_1} ID'^{-\hat{r}_1 t_2}, h^{x_2} ID'^{-\hat{r}_1 t_1}, ID'^{-\hat{r}_2 t_4}, ID'^{-\hat{r}_2 t_3}).$$

4. $\mathcal{KGC}$ sends the blinded key $sk_{id}{}' = (d_0', d_1', d_2', d_3', d_4')$ to $\mathcal{U}$, and engages in a proof of knowledge that it is correctly constructed. The proof assures $\mathcal{U}$ that $\mathcal{KGC}$'s chosen values $\hat{r}_1, \hat{r}_2, open_{\hat{r}_1}, open_{\hat{r}_2}, t_1, t_2, t_3, t_4, x_0, x_1, x_2, open_{x_0}, open_{x_1}, open_{x_2}$ correspond to $sk_{id}{}'$ and to the commitments $C_{\hat{r}_1}$, $C_{\hat{r}_2}$, C_{x_0}, C_{x_1} and C_{x_2} (see Appendix A). If the proof fails, $\mathcal{U}$ returns $\perp$. Otherwise, she computes

$$\begin{aligned} sk_{id} = (d_0, d_1, d_2, d_3, d_4) = (d_0' h^{-u_0}, (d_1' h^{-u_1})^{r_1'/u_3}, (d_2' h^{-u_2})^{r_1'/u_3}, d_3'^{\,r_2'/u_3}, \\ d_4'^{\,r_2'/u_3})\,. \end{aligned}$$

Theorem 2. *The* IBEBlindExtract *protocol provides a* leak-free *and* selective-failure blind *committed blind extraction protocol for the adaptive identity secure anonymous IBE scheme.*

Proof. Leak freeness: Note that the simulator $\mathcal{S}$ can rewind an instance of the adversary $\mathcal{A}$ that he runs internally. He simulates the communication between the distinguisher $\mathcal{D}$ and $\mathcal{A}$ by passing $\mathcal{D}$'s input to $\mathcal{A}$ and $\mathcal{A}$'s output to $\mathcal{D}$.

In the two party protocol $\mathcal{S}$ can provide random input. Using rewinding techniques, $\mathcal{S}$ extracts $\mathcal{A}$'s input r_1', r_2', and u_0, u_1, u_2, u_3 to the two party computation protocol. In the next step of the blind issuing protocol $\mathcal{A}$ must send $ID' = H_2(id)^{u_3}$ together with a proof of knowledge of a correct representation of ID' and C_{id}. $\mathcal{S}$ uses its rewinding access to $\mathcal{A}$ in order to also extract id, and $open_{id}$.

Next $\mathcal{S}$ submits id, $open_{id}$ to $\mathcal{O}_{\mathsf{IBEExtract}}$ to obtain a valid secret key $sk_{id} = (d_0, d_1, d_2, d_3, d_4)$. $\mathcal{S}$ returns $(d_0 \cdot h^{u_0}, d_1^{u_3/r'_1} h^{u_1}, d_2^{u_3/r'_1} h^{u_2}, d_3^{u_3/r'_2}, d_4^{u_3/r'_2})$ to $\mathcal{A}$. These values are distributed in the same way as in IBEBlindExtract.

Selective-failure blindness: $\mathcal{A}$ provides $params$, and two identities id_0, id_1. The game chooses a random bit b. $\mathcal{A}$ is given commitments $C_b = \mathsf{Commit}(id_b, open_b)$ and $C_{1-b} = \mathsf{Commit}(id_{1-b}, open_{1-b})$. $\mathcal{A}$ has blackbox access to two oracles $\mathcal{U}(params, id_{1-b}, open_{1-b})$ and $\mathcal{U}(params, id_b, open_b)$.

Note that once an oracle $\mathcal{U}$ is activated, $\mathcal{A}$ can run a two-party protocol with the oracle, the result of which are three randomly distributed values in Z_p (x_0, x_1, x_2). In the next step, the oracle provides a randomly distributed value in G_2 (ID'), to $\mathcal{A}$. Then the oracle performs a zero-knowledge proof with $\mathcal{A}$.

Suppose that $\mathcal{A}$ runs one or both of the oracles up to this point. Up to now the distributions of the two oracles are computationally indistinguishable. (Otherwise we could break the security of the two party computation, the hiding property of the commitment scheme or the witness indistinguishability of the zero-knowledge proof. The latter is implied by the zero-knowledge property of the proof system.)

$\mathcal{A}$ must provide values $(d'_0, d'_1, d'_2, d'_3, d'_4)$ and a proof that these values were correctly computed. We can assume that $\mathcal{A}$ chooses these values using an arbitrary complex strategy. We show that any adversary $\mathcal{A}$ can predict the output sk_i of $\mathcal{U}$ without further interaction with the oracles:

1. $\mathcal{A}$ does the proof of Step 4 internally with itself. If the proof fails, it records $sk_0 = \perp$. Otherwise, the adversary temporarily records sk_0=$\mathsf{IBEExtract}(params, msk, id_0)$.
2. In turn, $\mathcal{A}$ generates different $(d'_0, d'_1, d'_2, d'_3, d'_4)$ and executes a second proof of knowledge (again internally), now for the second oracle. It performs the same checks and recordings for sk_1 and id_1.
3. Finally the adversary predicts (sk_0, sk_1), if both $sk_0 \neq \perp$ and $sk_1 \neq \perp$; $(\epsilon, \perp)$, if only $sk_1 = \perp$; $(\perp, \epsilon)$, if only $sk_0 = \perp$; and $(\perp, \perp)$, if $sk_0 = sk_1 = \perp$.

These predictions result in the same distributions as that returned by the oracle, as the same checks are performed. Moreover, note that for the case that keys are returned by the game they are in both cases equally distributed random keys because of the random values r'_1 and r'_2 contributed by the oracles.

4.3 Two-Party Protocol for Modulo Arithmetics

The protocol uses a public key additive homomorphic encryption scheme with encryption and decryption functions HEnc and HDec, such that the following hold: $\mathsf{HEnc}(x) \otimes y = \mathsf{HEnc}(xy)$ and $\mathsf{HEnc}(x) \oplus \mathsf{HEnc}(y) = \mathsf{HEnc}(x + y)$. In addition, the encryption should be verifiable [36], meaning it should allow for efficient proofs of knowledge about the encrypted content. The key pair is generated by the $\mathcal{KGC}$ and is made available to $\mathcal{U}$.

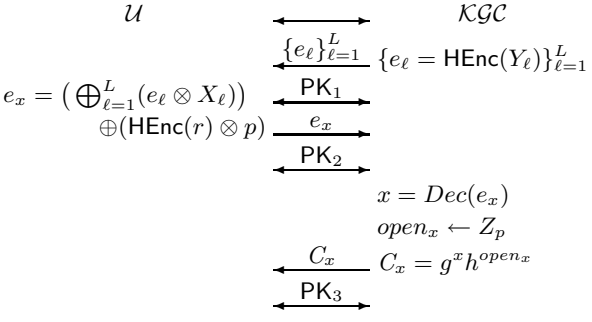
We describe an efficient committed two-party computation protocol for computing algebraic terms with addition and multiplication modulo a prime p that generalises ideas presented in [37]. The round complexity of the protocol is 3 if non-interactive proofs of knowledge are used and 12 if interactive proofs of knowledge are used.[5]

[5] The round complexity can be reduced by interleaving the proofs and piggybacking some of the messages.

Let $x_1, \ldots, x_N, open_{x_1}, \ldots, open_{x_N} \in Z_p$ and $y_1, \ldots, y_M, open_{y_1}, \ldots, open_{y_M} \in Z_p$ be the secret input variables and openings of $\mathcal{U}$ and $\mathcal{KGC}$ respectively and let $C_{x_1}, \ldots, C_{x_N}$ and $C_{y_1}, \ldots, C_{y_M}$ be public commitments to the x_i and y_i. We provide a protocol for computing the multivariate polynomial $\sum_{\ell=1}^{L} a_\ell \prod_{n=1}^{N} x_n^{u_{\ell n}} \prod_{m=1}^{M} y_m^{v_{\ell m}}$ where $u_{11}, \ldots, u_{LN}, v_{11}, \ldots, v_{LM} \in \{0, 1\}$ and $a_\ell \in Z_p$ are publicly known values.

The parties can do parts of the computation locally: $\mathcal{U}$ sets $X_\ell = a_\ell \prod_{n=1}^{N} x_n^{u_{\ell n}} \mod p$ and $\mathcal{KGC}$ sets $Y_\ell = \prod_{m=1}^{M} y_m^{v_{\ell m}} \mod p$. To prove that the computation was done correctly $\mathcal{U}$ computes commitment $C_{X_\ell} = \mathsf{Commit}(X_\ell, open_{X_\ell})$ and $\mathcal{KGC}$ computes $C_{Y_\ell} = \mathsf{Commit}(Y_\ell, open_{Y_\ell})$.

The parties can complete the computation using homomorphic encryption as described in the following protocol (The message space of the homomorphic encryption needs to be at least $2^k \ell p^2$). The $\bigoplus$ operator denotes the homomorphic addition of multiple ciphertexts.



The $\mathcal{KGC}$ encrypts each Y_ℓ and sends it to the user. The proof PK_1 assures $\mathcal{U}$ that C_{Y_ℓ} was computed correctly using the values in commitments C_{y_i} and that the e_ℓ are encryptions of the values committed to in the C_{Y_ℓ}.

Next, $\mathcal{U}$ computes the encrypted result. The term $r \cdot p, 0 < r < (2^k - 1)\ell p$ is added to avoid possible modulo overflows from revealing any statistically significant information about $\mathcal{U}$'s input. $\mathcal{U}$ proves to $\mathcal{KGC}$ in PK_2 that C_{X_ℓ} was computed correctly using the values in commitments C_{x_i} and that e_x was computed correctly using the values committed to in the C_{Y_ℓ}.

As a last step, $\mathcal{KGC}$ decrypts e_x, does a single modulo p reduction to obtain the result of the computation, and commits to the result in commitment C_x. In PK_3 $\mathcal{KGC}$ proves to the user that C_x contains the same value modulo p as encrypted in e_x. For details on how to do the proofs $\mathsf{PK}_1, \ldots, \mathsf{PK}_3$ we refer to [29,38]. An efficient implementation of such a protocol is presented in [37] using the Paillier homomorphic encryption scheme [39].

4.4 Transformation to PEOKS

We construct a suitable PEKS scheme for our application scenario using the anonymous IBE scheme presented in Sect. 4.1. We follow a generic transformation by Abdalla et al. [30] from IBE to PEKS. The transformation takes as input the algorithms Π of a secure IBE scheme and returns a PEKS scheme $\Upsilon = (\mathsf{KeyGen}, \mathsf{PEKS}, \mathsf{Trapdoor}, \mathsf{Test})$. Note that our scheme differs from preexisting schemes as Test returns a secret message in case of a match:

$\mathsf{KeyGen}(1^k)$ runs algorithm $\mathsf{IBESetup}(1^k)$ and returns the key pair (A_{pub}, A_{priv}), the $(params, msk)$ of the IBE scheme.

$\mathsf{PEKS}(A_{pub}, W, msg)$ takes as input public key A_{pub}, keyword W and message msg. It outputs a searchable encryption S_W of message msg under keyword W as follows:

1. Generate a random value $C_2 \in \{0,1\}^k$.
2. Compute $C_1 = \mathsf{IBEEnc}(A_{pub}, W, msg \| C_2)$.
3. Output the tuple $S_W = (C_1, C_2)$.

$\mathsf{Trapdoor}(A_{pub}, A_{priv}, W)$ outputs a trapdoor $T_W = \mathsf{IBEExtract}(A_{pub}, A_{priv}, W)$ that enables a search for the keyword W.

$\mathsf{Test}(A_{pub}, S_W, T_{W'})$ parses S_W as (C_1, C_2) and computes $M = \mathsf{IBEDec}(A_{pub}, T_{W'}, C_1)$. If $M = msg \| C_2$, it outputs the message msg encoded in S_W; if there is no match, it outputs $\perp$.

In order to achieve the oblivious property in our PEOKS scheme, we extend algorithm Trapdoor to a BlindTrapdoor protocol. Our PEOKS scheme is thus composed of the algorithms Υ of the PEKS scheme, a secure commitment scheme Commit, and a BlindTrapdoor protocol where

$\mathsf{BlindTrapdoor}(\mathcal{U}(A_{pub}, W, open_W), \mathcal{TGE}(A_{pub}, A_{priv}, C))$ generates a trapdoor T_W for a keyword W by running protocol $\mathsf{IBEBlindExtract}(\mathcal{U}(A_{pub}, W, open_W), \mathcal{KGC}(A_{pub}, A_{priv}, C))$.

5 Authorised Private Searches on Public Key Encrypted Data

We describe a public key encrypted database that enables oblivious searches. Our construction is similar to the audit log presented in [3]. Each data record is encrypted using a fresh random symmetric key and associated with several searchable encryptions. Each searchable encryption is generated using input of a keyword that describes the content of the record, and a secret message that contains the symmetric key. Once an investigator obtains a trapdoor that matches a searchable encryption (i.e., both were computed on input the same keyword), she is returned the symmetric key that allows her to decrypt the record.

In constructing authorised private searches, we ensure that neither the keywords of interest for the investigator nor the search results are revealed. For the first property, we employ the PEOKS scheme. The investigator runs protocol BlindTrapdoor with the trapdoor generation entity ($\mathcal{TGE}$) in order to retrieve a trapdoor for a committed keyword in a blind manner. The committed blind extraction allows the $\mathcal{TGE}$ to construct policies detailing the data that a particular investigator can obtain. To enforce these restrictions, the $\mathcal{TGE}$ requires the investigator to prove in zero-knowledge that the keyword used to compute the commitment belongs to a certain language. We also consider a party (such as a judge) in charge of deciding which keywords can be utilized by the investigator, and describe how the investigator obtains a search warrant from the judge and shows it to the $\mathcal{TGE}$. The judge and the $\mathcal{TGE}$ are only involved in providing search warrants and trapdoors respectively, and can remain off-line when not required to perform these tasks.

To obscure the search results, we describe a data structure that allows the use of a PIR scheme and that integrates concepts from [40] to improve the efficiency of the searches[6]. Since the PIR queries are made over encrypted data, we also ensure that the investigator does not obtain any information about data described by keywords for which she was not authorised to retrieve a trapdoor. It should also be noted that, due to the public key setting, the database only stores the public key of the PEOKS scheme. Thus, in the event that it gets corrupted, records encrypted prior to corruption remain secure (forward secrecy).

Details on data storage. We describe a data structure in which only one searchable encryption per keyword is computed, while still allowing each data record to be described by several keywords. Once the investigator finds the searchable encryption that matches her trapdoor, she receives the information needed to decrypt all the data records described by the corresponding keyword. This mechanism of data storage allows for an efficient search (not all the searchable encryptions need to be tested) and is privacy enhancing in so far as it hides the number of keywords that describe a record from the investigator.

We use encrypted linked lists and store the encrypted nodes at random positions in the PIR database to hide which node belongs to which linked list, as introduced in [40]. We construct one linked list per keyword. Each node in the linked list contains the information required to retrieve and decrypt one record associated with the keyword. A node contains a PIR query index P_R for the data record and the key K_R used to encrypt the record. It also stores a PIR query index to the next node on the list, and the key used to encrypt it. To encrypt the nodes and the records of data, we employ a symmetric encryption algorithm Enc.

When the data holder adds a keyword W for which no searchable encryption has previously been computed, he chooses a symmetric key K_{N_1} and runs algorithm $\mathsf{PEKS}(A_{pub}, W, K_{N_1}||P_{N_1})$ to compute the searchable encryption. P_{N_1} is the PIR query index to the first node of the list and K_{N_1} is the symmetric key used to encrypt this node. He then builds the node $N_1 = (P_R, K_R, P_{N_2}, K_{N_2})$, computes $\mathsf{Enc}(K_{N_1}, N_1)$, and stores the node in the position given by P_{N_1}. Finally, he deletes P_{N_1} and K_{N_1} from his memory but keeps values P_{N_2} and K_{N_2}. P_{N_2} and K_{N_2} are the PIR query index and the key for the next node in the list. In position P_{N_2} a flag is stored to indicate the end of the list.

When the data holder chooses this keyword to describe another record R', it builds the second node $N_2 = (P_{R'}, K_{R'}, P_{N_3}, K_{N_3})$, runs $\mathsf{Enc}(K_{N_2}, N_2)$, and stores the encrypted node in the position given by P_{N_2}. It deletes P_{N_2} and K_{N_2} from his memory but keeps P_{N_3} and K_{N_3} to facilitate adding another node to the list. He also stores the flag in P_{N_3}. This iterative procedure is applied as many times as required.

If a data record is described by several keywords, one node per keyword is generated and stored in its corresponding linked list. All these nodes contain the same PIR query index to the data record and the same key used to encrypt the record.

[6] The amount of PIR queries may give some indication about the number of records retrieved. This information can be hidden through dummy transactions up to an upper limit on the number of matching records.

Authorizing and performing private searches. An investigator that wants to search on the encrypted database follows the procedure:

1. The investigator requests authorisation from the judge to perform a search on a given database for a particular keyword W. Assuming the investigator holds the relevant credentials, the judge grants a warrant. In practice, this means that the investigator runs a protocol GetCredential with the judge, which returns to the investigator a credential $cred$ with attribute W from the judge.
2. The investigator requests a trapdoor from the $\mathcal{TGE}$. This is a three step process:
   (a) The investigator has a commitment $C = Commit(W, open_W)$ to the keyword W for which she wants to receive a trapdoor, and sends C to the $\mathcal{TGE}$.
   (b) The investigator and the $\mathcal{TGE}$ run an interactive protocol, ShowCredential. This verifies the validity of the credential presented by the investigator and the claim that the keyword used to compute the commitment is the same as the keyword contained in the credential's attributes.
   (c) The investigator and the $\mathcal{TGE}$ execute the BlindTrapdoor protocol, with investigator input $A_{pub}, W, open_W$ and $\mathcal{TGE}$ input A_{pub}, A_{priv}, C. The protocol returns no output to the $\mathcal{TGE}$, and a trapdoor T_W to the investigator.
3. The investigator downloads the list of PEKS elements for all the keywords.
4. If an investigator performs a successful Test for a PEKS element (using the correct trapdoor), the algorithm returns the key and PIR query index pair that correspond to the first node of the list. The investigator uses the PIR scheme to retrieve the node and the first record. As above, each node returns sufficient information to link to the next node, until all data related to the keyword have been returned.

Remark. GetCredential and ShowCredential can be implemented using conventional signatures: during GetCredential the judge signs $C = Commit(W, open_W)$ to create the credential $cred$ (a signature on C); in the ShowCredential protocol the investigator sends $cred$ together with C and the $\mathcal{TGE}$ verifies the signature. More sophisticated credential protocols [41,42,43,44,45] allow the implementation of more complex policies, such as, e.g., the *time restricted searches* described below.

Time restricted searches. In PEKS, the notion of temporary keyword search [30] implies that searchable encryptions and trapdoors are related to a specific time period in such a way that, even if the keyword used to compute them is the same, they do not match if the time period is different. The simplest way to build PEKS with temporary keyword search is to concatenate keywords W and time periods t when computing searchable encryptions and trapdoors. When applying this solution to our database, multiple linked lists are generated for the same keyword concatenated, each corresponding to a different time frame.

This function is useful to provide searches in which the investigator is allowed to obtain all records described by a specific keyword that were stored within a restricted period of time. In this case, the credential issued by the judge is extended to contain two additional attributes t_1 and t_2 corresponding to a start time-stamp and end time-stamp which limit the period of an investigation. When showing the credential to the $\mathcal{TGE}$, the investigator computes a commitment to $W||t$ and also proves that $t_1 \leq t \leq t_2$. This can for instance be done using the techniques described in [32].

6 Conclusion and Future Work

We have defined and implemented a searchable encryption scheme, PEOKS, that allows for oblivious searches on public key encrypted data. For this purpose, we have extended the PEKS primitive by adding blind trapdoor extraction with committed keywords. In order to implement PEOKS, we have defined committed blind anonymous IBE and we have provided a construction of such a scheme. Finally, we applied PEOKS to build a public key encrypted database that permits authorised private searches.

As future work we leave the design of a blind key extraction protocol secure under concurrent execution. Furthermore, more efficient anonymous identity-based encryption schemes with more light weight key derivation protocols would translate directly into highly efficient PEOKS. Unfortunately, the scheme in [8] does not seem fit for our purposes as it uses stateful randomness in the secret key generation phase.

We observe that in a practical application it is likely that an investigator would want to search for data described by a predicate formed by conjunctions and disjunctions of keywords. Future work would focus on using attribute-hiding predicate encryption [46] to build a scheme that permits oblivious searches on encrypted data by specifying predicates of keywords.

Acknowledgements

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no 216483. It has also been funded by a Science Foundation of Ireland Basic Research Grant, project number 04/BR/CS0692.

References

1. Directive 2006/24/ec of the european parliament and of the council. Official Journal of the European Union (April 2006)
2. Ostrovsky, R., Skeith III, W.E.: Private searching on streaming data. J. Cryptology 20(4), 397–430 (2007)
3. Waters, B.R., Balfanz, D., Durfee, G., Smetters, D.K.: Building an encrypted and searchable audit log. In: NDSS, The Internet Society (2004)
4. Chor, B., Gilboa, N., Naor, M.: Private information retrieval by keywords (1998)
5. Ogata, W., Kurosawa, K.: Oblivious keyword search. J. Complexity 20(2-3), 356–371 (2004)
6. Boneh, D., Franklin, M.K.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
7. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: Proceedings of Eurocrypt, vol. 4 (2004)
8. Gentry, C.: Practical identity-based encryption without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 445–464. Springer, Heidelberg (2006)
9. Boyen, X., Waters, B.: Anonymous hierarchical identity-based encryption (without random oracles). In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 290–307. Springer, Heidelberg (2006)

10. Green, M., Hohenberger, S.: Blind identity-based encryption and simulatable oblivious transfer. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 265–282. Springer, Heidelberg (2007)
11. Boneh, D., Boyen, X.: Efficient selective-id secure identity-based encryption without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)
12. Naccache, D.: Secure and practical identity-based encryption. Information Security, IET 1(2), 59–64 (2007)
13. Chatterjee, S., Sarkar, P.: Trading time for space: Towards an efficient ibe scheme with short(er) public parameters in the standard model. In: Won, D.H., Kim, S. (eds.) ICISC 2005. LNCS, vol. 3935, pp. 424–440. Springer, Heidelberg (2006)
14. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. J. ACM 51(4), 557–594 (2004)
15. Dwork, C., Naor, M., Reingold, O., Stockmeyer, L.J.: Magic functions. J. ACM 50(6), 852–921 (2003)
16. Goldwasser, S., Kalai, Y.T.: On the (in)security of the fiat-shamir paradigm. In: FOCS, p. 102. IEEE Computer Society, Los Alamitos (2003)
17. Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)
18. Coull, S., Green, M., Hohenberger, S.: Controlling access to an oblivious database using stateful anonymous credentials. Cryptology ePrint Archive, Report 2008/474 (2008), http://eprint.iacr.org/
19. Chor, B., Goldreich, O., Kushilevitz, E., Sudan, M.: Private information retrieval. In: FOCS, pp. 41–50 (1995)
20. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)
21. Schnorr, C.P.: Efficient signature generation for smart cards. Journal of Cryptology 4(3), 239–252 (1991)
22. Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 89–105. Springer, Heidelberg (1993)
23. Camenisch, J., Michels, M.: Proving in zero-knowledge that a number n is the product of two safe primes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 107–122. Springer, Heidelberg (1999)
24. Camenisch, J.L.: Group Signature Schemes and Payment Systems Based on the Discrete Logarithm Problem. PhD thesis, ETH Zürich (1998)
25. Brands, S.: Rapid demonstration of linear relations connected by boolean operators. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 318–333. Springer, Heidelberg (1997)
26. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 174–187. Springer, Heidelberg (1994)
27. Damgård, I.: Concurrent zero-knowledge is easy in practice. Available online at Theory of Cryptography Library (June 1999)
28. Damgård, I.: On σ-protocols (2002), http://www.daimi.au.dk/~ivan/Sigma.ps
29. Camenisch, J., Stadler, M.: Proof systems for general statements about discrete logarithms. Technical Report TR 260, Institute for Theoretical Computer Science, ETH Zürich (March 1997)

30. Abdalla, M., Bellare, M., Catalano, D., Kiltz, E., Kohno, T., Lange, T., Malone-Lee, J., Neven, G., Paillier, P., Shi, H.: Searchable Encryption Revisited: Consistency Properties, Relation to Anonymous IBE, and Extensions. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 205–222. Springer, Heidelberg (2005)
31. Bellare, M., Boldyreva, A., Desai, A., Pointcheval, D.: Key-privacy in public-key encryption. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 566–582. Springer, Heidelberg (2001)
32. Boudot, F.: Efficient proofs that a committed number lies in an interval. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 431–444. Springer, Heidelberg (2000)
33. Camenisch, J., Neven, G., Shelat, A.: Simulatable adaptive oblivious transfer. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 573–590. Springer, Heidelberg (2007)
34. Galbraith, S., Paterson, K., Smart, N.: Pairings for cryptographers. Cryptology ePrint Archive, Report 2006/165 (2006), http://eprint.iacr.org/
35. Yao, A.C.: Protocols for secure computations. In: Proc. 23rd IEEE Symposium on Foundations of Computer Science (FOCS), pp. 160–164 (1982)
36. Camenisch, J., Damgård, I.: Verifiable encryption, group encryption, and their applications to group signatures and signature sharing schemes. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 331–345. Springer, Heidelberg (2000)
37. Camenisch, J., Koprowski, M., Warinschi, B.: Efficient blind signatures without random oracles. In: Blundo, C., Cimato, S. (eds.) SCN 2004. LNCS, vol. 3352, pp. 134–148. Springer, Heidelberg (2005)
38. Camenisch, J., Shoup, V.: Practical verifiable encryption and decryption of discrete logarithms. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 126–144. Springer, Heidelberg (2003)
39. Paillier, P.: Public-key cryptosystems based on composite residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–239. Springer, Heidelberg (1999)
40. Curtmola, R., Garay, J.A., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: improved definitions and efficient constructions. In: Juels, A., Wright, R.N., di Vimercati, S.D.C. (eds.) ACM Conference on Computer and Communications Security, pp. 79–88. ACM, New York (2006)
41. Brands, S.: Rethinking Public Key Infrastructure and Digital Certificates— Building in Privacy. PhD thesis, Eindhoven Inst. of Tech. The Netherlands (1999)
42. Camenisch, J., Lysyanskaya, A.: Efficient non-transferable anonymous multi-show credential system with optional anonymity revocation. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 93–118. Springer, Heidelberg (2001)
43. Camenisch, J., Lysyanskaya, A.: A signature scheme with efficient protocols. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 2002. LNCS, vol. 2576, pp. 268–289. Springer, Heidelberg (2003)
44. Camenisch, J., Lysyanskaya, A.: Signature schemes and anonymous credentials from bilinear maps. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 56–72. Springer, Heidelberg (2004)
45. Bangerter, E., Camenisch, J., Lysyanskaya, A.: A Cryptographic Framework for the Controlled Release Of Certified Data. In: 12th International Workshop on Security Protocols 2004, Cambridge, England, April 26, 2004, pp. 20–42. Springer, Heidelberg (2004)
46. Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. Cryptology ePrint Archive, Report 2007/404 (2007), http://eprint.iacr.org/

A Proofs of Knowledge of Correct Key Derivation

Proof for Step 2. The $\mathcal{KGC}$ has commitment C_{u_3} to u_3, and C_{id} to the user's choice of id. In Step 2 of our IBEBlindExtract protocol the user does the following proof of knowledge to convince the $\mathcal{KGC}$ that her message ID' is well formed:

$$
\begin{aligned}
PK\{&(id_1, \ldots, id_n, u_3, id_1 \cdot u_3, \ldots, id_n \cdot u_3, open_{id}, open_{u_3}, \\
&open_{id} \cdot u_3) : C_{id} = (\prod_{i=1}^{n} (h_0^{2^{l(i-1)}})^{id_i}) h_1^{open_{id}} \wedge && (1)\\
&\bigwedge_{i=1}^{n} 0 \leq id_i < 2^l \wedge C_{u_3} = h_0^{u_3} h_1^{open_{u_3}} \wedge && (2)\\
&1 = C_{id}^{u_3} (\prod_{i=1}^{n} ((1/h_0)^{2^{l(i-1)}})^{id_i \cdot u_3}) (1/h_1)^{open_{id} \cdot u_3} \wedge && (3)\\
&ID' = h_0^{u_3} \prod_{i=1}^{n} h_i^{id_i \cdot u_3} \} \, . && (4)
\end{aligned}
$$

The user proves that id is correctly encoded in ID'. This is done in two steps: (1) and (2) prove that id is correctly split up into its n components id_i; (3) and (4) prove that ID' contains a blinded version of $H_2(id)$. This requires to prove multiplicative relations between u_3 and the id_i in (3).

Proof for Step 4. The user has commitments $C_{\hat{r}_1}$, $C_{\hat{r}_2}$ and C_{x_0}, C_{x_1}, and C_{x_2}. In Step 4 of our IBEBlindExtract protocol the $\mathcal{KGC}$ does the following proof of knowledge to convince the user that the blinded key $(d'_0, d'_1, d'_2, d'_3, d'_4)$ it returns is well formed:

$$
\begin{aligned}
PK\{&(\hat{r}_1, \hat{r}_2, open_{\hat{r}_1}, open_{\hat{r}_2}, t_1, t_2, t_3, t_4, x_0, x_1, x_2, open_{x_0}, \\
&open_{x_1}, open_{x_2}, -\hat{r}_1 t_1, -\hat{r}_1 t_2, -\hat{r}_2 t_3, -\hat{r}_2 t_4) : \\
&C_{\hat{r}_1} = h_0^{\hat{r}_1} h_1^{open_{\hat{r}_1}} \wedge C_{\hat{r}_2} = h_0^{\hat{r}_2} h_1^{open_{\hat{r}_2}} \wedge v_1 = g^{t_1} \wedge v_2 = g^{t_2} \wedge v_3 = g^{t_3} \wedge \\
&v_4 = g^{t_4} \wedge C_{x_0} = h_0^{x_0} h_1^{open_{x_0}} \wedge C_{x_1} = h_0^{x_1} h_1^{open_{x_1}} \wedge C_{x_2} = h_0^{x_2} h_1^{open_{x_2}} \wedge \\
&1 = (1/v_1)^{\hat{r}_1} (1/g)^{-\hat{r}_1 t_1} \wedge 1 = (1/v_2)^{\hat{r}_1} (1/g)^{-\hat{r}_1 t_2} \wedge 1 = (1/v_3)^{\hat{r}_2} (1/g)^{-\hat{r}_2 t_3} \wedge \\
&1 = (1/v_4)^{\hat{r}_2} (1/g)^{-\hat{r}_2 t_4} \wedge d'_0 = h^{x_0} \wedge d'_1 = h^{x_1} ID'^{-\hat{r}_1 t_2} \wedge d'_2 = h^{x_2} ID'^{-\hat{r}_1 t_1} \wedge \\
&d'_3 = ID'^{-\hat{r}_2 t_4} \wedge d'_4 = ID'^{-\hat{r}_2 t_3} \} \, .
\end{aligned}
$$

By means of this proof the $\mathcal{KGC}$ demonstrates to the user that it uses the correct values for $x_0, x_1, x_2, t_1, t_2, t_3, t_4, \hat{r}_1, \hat{r}_2$ when it computes $(d'_0, d'_1, d'_2, d'_3, d'_4)$. The proof involves proving the multiplicative relations $-\hat{r}_1 t_1, -\hat{r}_1 t_2, -\hat{r}_2 t_3, -\hat{r}_2 t_4$ between $t_1, t_2, t_3, t_4, \hat{r}_1, \hat{r}_2$.