Public Key Encryption with Searchable Keywords Based on Jacobi Symbols

Giovanni Di Crescenzo[1] and Vishal Saraswat[2]

[1] Telcordia Technologies, Piscataway-NJ, USA
giovanni@research.telcordia.com
[2] University of Minnesota, Minneapolis-MN, USA
vishal@math.umn.edu

Abstract. Public-key encryption schemes with searchable keywords are useful to delegate searching capabilities on encrypted data to a third party, who does not hold the entire secret key, but only an appropriate token which allows searching operations but preserves data privacy. Such notion was previously proved to imply identity-based public-key encryption [5] and to be equivalent to anonymous (or key-private) identity-based encryption which are useful for fully-private communication.

So far all presented public-key encryption with keyword search (PEKS) schemes were based on bilinear forms and finding a PEKS that is not based on bilinear forms has been an open problem since the notion of PEKS was first introduced in [5]. We construct a public-key encryption scheme with keyword search based on a variant of the *quadratic residuosity problem*. We obtain our scheme using a non-trivial transformation of Cocks' identity-based encryption scheme [9]. Thus we show that the primitive of PEKS can be based on additional intractability assumptions which is a conventional desiderata about all cryptographic primitives.

Keywords: Public-Key Encryption, Searchable Public-Key Encryption, Quadratic Residuosity, Jacobi Symbol.

1 Introduction

A classical research area in Cryptography is that of designing candidates for cryptographic primitives under different intractability assumptions, so to guarantee that the cryptographic primitive does not depend on the supposed hardness of a single computational problem and its fortune against cryptanalytic research. In this paper we concentrate on a recently introduced primitive, *public-key encryption with keyword search* (PEKS) [5], for which all constructions in the literature were based on assumptions related to bilinear forms. We present a PEKS scheme based on a new assumption that can be seen as a variant of the classical assumption on the hardness of deciding quadratic residuosity modulo composite integers.

Motivation. PEKS allows a sender to compute an encrypted message, so that the receiver can allow a third party to search keywords in the encrypted message without (additional) loss of privacy on the content of the message. The following motivating example for PEKS is taken almost verbatim from [5]. Suppose user Alice wishes to read her email on a number of devices: laptop, desktop, pager, etc. Alice's mail gateway is supposed to route email to the appropriate device based on the keywords in the email. For example, when Bob sends email with the keyword "urgent" the mail is routed to Alice's pager. When Bob sends email with the keyword "lunch" the mail is routed to Alice's desktop for reading later. One expects each email to contain a small number of keywords. For example, all words on the subject line as well as the sender's email address could be used as keywords. Now, suppose Bob sends encrypted email to Alice using Alice's public key. Both the contents of the email and the keywords are encrypted. In this case the mail gateway cannot see the keywords and hence cannot make routing decisions. With public-key encryption with keyword search one can enable Alice to give the gateway the ability to test whether "urgent" is a keyword in the email, but the gateway should learn nothing else about the email. More generally, Alice should be able to specify a few keywords that the mail gateway can search for, but learn nothing else about incoming mail.

Previous work. In its non-interactive variant, constructions for this primitive were showed to be at least as hard to obtain as constructions for identity-based encryption (as proved in [5]). Moreover, the existence of PEKS was proved to follow from the existence of "anonymous" or "key-private" identity-based encryption (this was noted in [5] and formally proved in [1]); namely, encryption where the identity of the recipient remains unknown. Anonymous encryption is well-known to be an attractive solution to the problem of fully-private communication (i.e., sender-anonymous and receiver-anonymous ciphertexts, as well as protection against traffic analysis, by using bulletin boards); see, e.g., discussions in [2,8]). It is a natural goal then to try to convert the existing identity-based public-key cryptosystems into their anonymous variant, so that a PEKS is automatically obtained. In fact, the anonymity or key-privacy property for a public-key encryption scheme (whether it is identity-based or not), is itself a property of independent interest, as already discussed in [2], where this property was defined and investigated for conventional (i.e., not identity-based) public-key encryption schemes. So far, however, all presented public-key encryption schemes with keyword search were transformations of identity-based cryptosystems based on bilinear forms. Even the authors of [5] noted the difficulty of coming up with other examples of public-key encryption schemes with keyword search, by observing that the only identity-based cryptosystem not based on bilinear forms (namely, Cocks' scheme [9]) does not seem to have a direct transformation into an anonymous variant and thus into a public-key encryption scheme with keyword search. Further work on PEKS (e.g., [17,15,13,1,8]) did not contribute towards this goal, but further studied schemes and variations based on bilinear forms.

Our results. In this paper we construct the first public-key encryption scheme with keyword search which is not based on bilinear forms but is based on a new assumption that can be seen as a variant of the well-known hardness of deciding quadratic residues modulo a large composite integer. Our scheme is obtained as a non-trivial transformation of Cocks' identity-based encryption scheme [9]. By the known equivalence of public-key encryption scheme with keyword search and anonymous identity-based encryption, our scheme immediately gives the first anonymous identity-based encryption scheme which is not based on bilinear forms, a problem left open in [5]. Our scheme essentially preserves the time efficiency of the (not anonymous) identity-based encryption of Cocks' scheme, which was claimed in the original paper [9] to be satisfactory in a hybrid encryption mode (that is, when used to encrypt first a short session key and then using this key to produce a symmetric encryption of a large message). We do note however that the decryption time of Cocks' scheme (and thus of our scheme too) is less efficient than the known schemes based on bilinear forms.

The construction of a new identity-based encryption scheme based on quadratic residuosity [7] and having short ciphertexts was claimed very recently, and after seeing our present work [3]. This scheme is also anonymous, like ours, but is based on very different techniques. Although their scheme is quite elegant, encryption and decryption operations are estimated [4] to be significantly less efficient than in Cocks' scheme. Instead, when used as an anonymous identity-based encryption scheme, our scheme is only less efficient than the original (and not anonymous) Cocks' scheme [9] by a small constant factor.

Organization of the paper. In what follows, we start by reviewing in Section 2 the formal definitions related to the notion of interest in this paper: public-key encryption with keyword search. In Section 3 we present our public-key cryptosystem with keyword search and in Section 4 we prove its properties.

2 Definitions and Preliminaries

We recall the known notion and formal definition of public-key encryption with keyword search (as defined in [5,1]). We assume familiarity with the notion of identity-based public-key cryptosystems (as defined, for instance, in [6,9]).

An identity-based public-key cryptosystem can be defined as a 4-tuple of algorithms (Setup, KeyGen, Encrypt, Decrypt), with the following semantics: Setup is used by the trusted authority TA to generate public parameters PK and a master secret key SK; KeyGen is used by the trusted authority TA to generate a trapdoor key t_{ID} given a party's ID; Encrypt is used by a sender who wants to encrypt a message to a receiving party and only uses the receiver's ID and the public parameters PK; Decrypt is used by a receiver to decrypt a ciphertext and only uses the trapdoor t_{ID} and the public parameters PK. We denote the identity-based cryptosystem in [9] as CC-IBE = (CC-Setup, CC-KeyGen, CC-Encrypt, CC-Decrypt):

2.1 Public-Key Encryption with Keyword Search

Informally speaking, in a public-key encryption scheme with keyword search, a sender would like to send a message in encrypted form to a receiver, so that the receiver can allow a third party to search keywords in the encrypted message without losing (additional) privacy on the message's content. According to [5], a non-interactive implementation of this task can be performed as follows. The sender encrypts her message using a conventional public key cryptosystem, and then appends to the resulting ciphertext a *Public-key Encryption with Keyword Search* (PEKS) of each keyword. Specifically, to encrypt a message M with searchable keywords $W_1, \ldots, W_m$, the sender computes and sends to the receiver

$$E_{A_{pub}}(M) \parallel \mathsf{ksEnc}(A_{pub}, W_1) \parallel \cdots \parallel \mathsf{ksEnc}(A_{pub}, W_m), \qquad (1)$$

where A_{pub} is the receiver's public key and E is the encryption algorithm of the conventional public-key cryptosystem. Based on this encryption, the receiver can give the third party a certain trapdoor T_W which enables the third party to test whether one of the keywords associated with the message is equal to the word W of the receiver's choice. Specifically, given $\mathsf{ksEnc}(A_{pub}, W')$ and T_W, the third party can test whether $W = W'$; if $W \neq W'$ the third party learns nothing more about W'. Note that sender and receiver do not communicate in this entire process, as the sender generates the searchable ciphertext for W' just given the receiver's public key (thus, the term "*public-key* encryption with keyword search" is used here).

More formally, we consider a setting with three parties: a sender, a receiver, and a third party (representing the e-mail gateway in the application example given in the introduction). In this setting, a public-key encryption with keyword search is defined as follows.

Definition 1. A (non-interactive) public-key encryption scheme with keyword search (PEKS) consists of the following polynomial time randomized algorithms:

1. $\mathsf{KeyGen}(1^m)$: on input security parameter 1^m in unary, it returns a pair (A_{pub}, A_{priv}) of public and private keys.
2. $\mathsf{ksEnc}(A_{pub}, W)$: on input a public key A_{pub} and a keyword W, it returns a ciphertext, also called the *searchable encryption* of W.
3. $\mathsf{Trapdoor}(A_{priv}, W)$: on input Alice's private key and a keyword W, it returns a trapdoor T_W.
4. $\mathsf{Test}(A_{pub}, S, T_W)$: on input Alice's public key, a searchable encryption $S = \mathsf{ksEnc}(A_{pub}, W')$, and a trapdoor $T_W = \mathsf{Trapdoor}(A_{priv}, W)$, it returns 'yes' or 'no'.

Given the above definition, an execution of a public-key encryption scheme with keyword search goes as follows. First, the receiver runs the KeyGen algorithm to generate her public/private key pair. Then, she uses the Trapdoor algorithm to generate trapdoors T_W for any keywords W which she wants the third party to search for. The third party uses the given trapdoors as input to the Test algorithm to determine whether a given message encrypted by any sender using algorithm ksEnc contains one of the keywords W specified by the receiver.

We now define three main properties which public-key encryption schemes with keyword search may satisfy: (two variants of) consistency and security. The following basic definition will be useful towards that: we say that a given function $f : N \rightarrow [0, 1]$ is *negligible in* n if $f(n) < 1/p(n)$ for any polynomial p and sufficiently large n.

Consistency. Next, we consider definitions of consistency for a PEKS (following definitions in [5,1]). We consider two variants: right-keyword consistency and adversary-based consistency for a PEKS in the random oracle model.

Informally, in right-keyword consistency, we require the success of the search of any word W for which the encryption algorithm had computed a searchable encryption.

Definition 2. We say that a PEKS is *right-keyword consistent* if it holds that for any word W, the probability that $\mathsf{Test}(A_{pub}, C, T_W) \neq$ 'yes' is negligible in m, where A_{pub} was generated using the KeyGen algorithm and input 1^m, C was computed as $\mathsf{ksEnc}(A_{pub}, W)$ and T_W was computed as $\mathsf{Trapdoor}(A_{priv}, W)$.

Informally, in adversary-based consistency, one would like to ensure that even an adversary that has access to the public parameters PK and to a (uniformly distributed) random oracle cannot come up with two different keywords such that the testing algorithm returns 'yes' on input a trapdoor for one word and a public-key encryption with keyword search of the other. Formally, we define consistency against an attacker $\mathcal{A}$ using the following game between a challenger and an attacker. Here, we denote by m the security parameter, given in unary as input to both players, and by k the length of the keywords, where we assume that $k = \Theta(m^c)$, for some constant $c > 0$ (this assumption is seen to be wlog using simple padding).

PEKS Adversary-Based Consistency Game

1. The challenger runs the $\mathsf{KeyGen}(1^m)$ algorithm to generate A_{pub} and A_{priv}. It gives A_{pub} to the attacker.
2. The attacker returns two keywords $W_0, W_1 \in \{0, 1\}^k$.
3. Encryption $C = \mathsf{ksEnc}(A_{pub}, W_0)$ and trapdoor $T_{W_1} = \mathsf{Trapdoor}(A_{priv}, W_1)$ are computed.
4. The attacker wins the game if $W_0 \neq W_1$ and $\mathsf{Test}(A_{pub}, C, T_{W_1})$ returns 'yes'.

We define $\mathcal{A}$'s advantage $\mathrm{Adv}_{\mathcal{A}}(m, k)$ in breaking the consistency of PEKS as the probability that the attacker wins the above game.

Definition 3. We say that a PEKS satisfies (computational) *adversary-based consistency* if for any attacker $\mathcal{A}$ running in time polynomial in m, we have that the function $\mathrm{Adv}_{\mathcal{A}}(m)$ is negligible in m.

Security. Finally, we recall the definition of security for a PEKS (in the sense of semantic-security). Here, one would like to ensure that an $\mathsf{ksEnc}(A_{pub}, W)$ does not reveal any information about W unless T_W is available. This is done by considering an attacker who is able to obtain trapdoors T_W for any W of his choice, and require that, even under such attack, the attacker should not be able to distinguish an encryption of a keyword W_0 from an encryption of a keyword W_1 for which he did not obtain the trapdoor. Formally, we define security against an active attacker $\mathcal{A}$ using the following game between a challenger and the attacker. Here, we denote by m the security parameter, given in unary as input to both players, and by k the length of the keywords, where we assume that $k = \Theta(m^c)$, for some constant $c > 0$ (this assumption is seen to be wlog using simple padding).

PEKS Security Game

1. The challenger runs the $\mathsf{KeyGen}(1^m)$ algorithm to generate A_{pub} and A_{priv}. It gives A_{pub} to the attacker.
2. The attacker can adaptively ask the challenger for the trapdoor T_W for any keyword $W \in \{0,1\}^k$ of his choice.
3. At some point, the attacker $\mathcal{A}$ sends the challenger two keywords W_0, W_1 on which it wishes to be challenged. The only restriction is that the attacker did not previously ask for the trapdoors T_{W_0} or T_{W_1}. The challenger picks a random $b \in \{0,1\}$ and gives the attacker $C = \mathsf{ksEnc}(A_{pub}, W_b)$. We refer to C as the challenge ciphertext.
4. The attacker can continue to ask for trapdoors T_W for any keyword W of his choice as long as $W \neq W_0, W_1$.
5. Eventually, the attacker $\mathcal{A}$ outputs $d \in \{0,1\}$.

Here, the attacker wins the game if its output differs significantly depending on whether he was given the challenge ciphertext corresponding to W_0 or W_1. This is formalized as follows. First, for $b = 0, 1$, let $\mathcal{A}^b = 1$ denote the event that $\mathcal{A}$ returns 1 given that $C = \mathsf{ksEnc}(A_{pub}, W_b)$. Then, define $\mathcal{A}$'s advantage in breaking the PEKS scheme as

$$\mathrm{Adv}_{\mathcal{A}}(m) = \left| \mathrm{Prob}[\, \mathcal{A}^0 = 1\,] - \mathrm{Prob}[\, \mathcal{A}^1 = 1\,] \right|$$

Definition 4. We say that a PEKS is semantically secure against an adaptive chosen-keyword attack if for any attacker $\mathcal{A}$ running in time polynomial in m, we have that the function $\mathrm{Adv}_{\mathcal{A}}(m)$ is negligible in m.

Remarks. We defined right-keyword consistency as done in [5,1] (although the name was first used in [1]). The (computational version of the) adversary-based consistency was defined as a relaxed version of what is called just consistency in [1]; the relaxation consisting in only restricting the adversary to return keywords which have a known upper-bounded length. Although guaranteeing a slightly weaker property, this is essentially not a limitation in practical scenarios where a (small) upper bound on the length of keywords is known to all parties. We also note that such a relaxation is always done, for instance, in the definition of conventional public-key cryptosystems.

3 Our Construction

In this section we present our main construction: a public-key encryption with keyword search under an intractability assumption related to quadratic residuosity modulo Blum-Williams integers. We first define the intractability assumption which we use and formally state our main result; then, in Subsection 3.1 we give an informal discussion where we sketch a preliminary (but flawed) construction, explain why it does not work, and how we fix it; finally, in Subsection 3.2, we formally describe our public-key cryptosystem with keyword search.

An intractability assumption. Our cryptosystem is based on the following assumption which is a variation of the well-known quadratic residuosity problem.

*Quadratic Indistinguishability Problem (*QIP*).* Let m be a security parameter. Let $(p, q) \leftarrow \mathrm{BW}(1^m)$ denote the random process of uniformly and independently choosing two m-bit primes p, q such that $p = q = 3 \bmod 4$. Let $QR(n)$ denote the set of quadratic residues modulo n and let $\mathbb{Z}_n^{+1}$ (resp. $\mathbb{Z}_n^{-1}$) be the set of positive integers which are $< n$, coprime with n and have Jacobi symbol equal to $+1$ (resp. -1). Also, let $s \leftarrow \mathrm{CS}(n, \alpha)$ denote the random process of randomly and independently choosing an integer s in $\mathbb{Z}_n^{+1}$ such that the condition α holds. The QIP *problem* consists of efficiently distinguishing the following two distributions:

$$\begin{aligned} D_0(1^m) = \{&(p, q) \leftarrow \mathrm{BW}(1^m); n \leftarrow p \cdot q; h \leftarrow \mathbb{Z}_n^{+1}; \\ &\qquad s \leftarrow \mathrm{CS}(n, s^2 - 4h \in \mathbb{Z}_n^{-1} \cup QR(n)) \,:\, (n, h, s)\} \\ D_1(1^m) = \{&(p, q) \leftarrow \mathrm{BW}(1^m); n \leftarrow p \cdot q; h \leftarrow \mathbb{Z}_n^{+1}; s \leftarrow \mathbb{Z}_n^* \,:\, (n, h, s)\}. \end{aligned}$$

We say that algorithm $\mathcal{A}$ has *advantage* ϵ in solving QIP if we have that:

$$\Big| \Pr[(n, h, s) \leftarrow D_0(1^m) : \mathcal{A}(n, h, s) = 1] - \Pr[(n, h, s) \leftarrow D_1(1^m) : \mathcal{A}(n, h, s) = 1] \Big| = \epsilon. \quad (2)$$

We say that QIP is *intractable* if all polynomial time (in m) algorithms have a negligible (in m) advantage in solving QIP.

Our result. In the rest of the paper we prove the following.

Theorem 1. Assume that the QIP problem is intractable. Then there exists (constructively) a public-key encryption scheme with keyword search.

3.1 An Informal Discussion

A first (not yet anonymous) construction. The first approach is a very natural one — we simply apply the Cocks' IBE scheme in place of Boneh-Franklin scheme as done in the public-key encryption scheme with keyword search presented in [5]. Given a security parameter 1^m, for $m \in \mathbb{Z}^+$, the user Alice uses algorithm CC-Setup to generate two sufficiently large primes p, q such that both p and q are congruent to $3 \mod 4$ and a cryptographic hash function H (assumed to behave like a random oracle in the analysis); then she outputs the public parameters $A_{pub} = (n, H)$ and keeps secret the master secret key $A_{priv} = (p, q)$. Alice treats each keyword W as an identity and, using algorithm CC-KeyGen, computes a square root g of $h = H(W)$ or $-h$ depending on which one is a square modulo n, and supplies the mail server with $T_W = g$ as the trapdoor for W. A user Bob wishing to send an encrypted email to Alice with the keyword W uses algorithm CC-Encrypt to encrypt the string 1^k, where $k = |W|$, using W as the necessary input identity, thus obtaining $\mathsf{ksEnc}(A_{pub}, W) = s = (s_1, \ldots, s_k, s_{k+1}, \ldots, s_{2k})$. The server then decrypts s as in algorithm CC-Decrypt and outputs 'yes' if the decryption returns 1^k, or outputs 'no' otherwise.

The problem with the first approach. The above scheme does not satisfy anonymity because Cocks' IBE scheme is not *public-key private* or *anonymous*, in the sense of Bellare et al. [2], and thus the ciphertext returned by algorithm PEKS may reveal more information than desired about the searchable keyword. This fact was already briefly mentioned in [5], but it is useful to analyze it in greater detail here to understand how we will obtain our main construction. Specifically, to clarify this fact, we note that for $i = 1, \ldots, k$, it holds that

$$s_i^2 - 4h = (t_i + h/t_i)^2 - 4h = (t_i - h/t_i)^2 \mod n$$

and therefore, except with negligible probability, $\left(\frac{s_i^2 - 4h}{n}\right) = +1$ and, analogously, for $i = k+1, \ldots, 2k$, it holds that

$$s_i^2 + 4h = (t_i - h/t_i)^2 + 4h = (t_i + h/t_i)^2 \mod n$$

and therefore, except with negligible probability, $\left(\frac{s_i^2 + 4h}{n}\right) = +1$.

On the other hand, for any other keyword $W' \neq W$, if $h' = H(W')$, the quantities $s_i^2 - 4h'$ and $s_i^2 + 4h'$ are not necessarily squares and their Jacobi symbols $\left(\frac{s_i^2 \pm 4h'}{n}\right)$ may be -1. In fact, when h' is randomly chosen in $\mathbb{Z}_n^*$, for each $i \in \{1, \ldots, 2k\}$, it holds that $\left(\frac{s_i^2 \pm 4h'}{n}\right) = -1$ exactly half the time $s_i^2 \pm 4h'$ is in $\mathbb{Z}_n^*$. Thus, an outsider can easily find out whether a keyword W is in the message or not with some non-negligible probability, which is not desirable.

Fixing the problems and ideas behind our construction. At a very high level, we still would like to use the approach in [5]; which, very roughly speaking, might be abstracted as follows: a searchable ciphertext is 'carefully computed' as the output of an identity-based encryption algorithm on input a plaintext sent in the clear and a function of the keyword W as the identity; the computation of the searchable ciphertext is such that (with high probability) the plaintext sent in the clear is the actual decryption of this ciphertext if and only if the trapdoor associated to the same keyword W is used to decrypt. However, the main difficulty in implementing this approach with (a modification of) the Cocks' scheme CC-IBE is in obtaining a modification which additionally satisfies the 'public-key privacy' or 'anonymity' property. We solve this problem by modifying the distribution of the ciphertext in CC-IBE, so that its modified distribution is 'properly randomized', and, when used in the context of a ciphertext associated with our public-key encryption scheme with keyword search, does not reveal which keyword is being used. The randomization of the ciphertext has to guarantee not only that the ciphertext does not reveal the identity used (or, in other words, the integer $h = H(ID)$), but also has to guarantee that the distribution remains the same when it is matched with another identity (e.g., another integer $h' = H(ID')$). In this randomization process, we have to take care of two main technical obstacles, one related to the distribution of the integers s_i with respect to the Jacobi symbols $\left(\frac{s_i^2 \pm 4h}{n}\right)$; and another one, related to efficiently guaranteeing that all values s_i are constructed using uniformly and independently distributed hashes (or, functions of the keyword) playing as the identity. We achieve this through two levels of randomization. First, the ciphertext contains $4k$ integers s_i in $\mathbb{Z}_n^*$ such that the Jacobi symbols of the related expressions $s_i^2 - 4h$ are uniformly distributed in $\{-1, +1\}$ whenever $s_i^2 \pm 4h$ is in $\mathbb{Z}_n^*$. Second, to make sure that these Jacobi symbols are also independently distributed, we do not use a single value h, but use a uniformly and independently distributed h_i for each index i.

3.2 Formal Description

We denote our public-key encryption with keyword search scheme as MainScheme $=$ (M-KeyGen, M-ksEnc, M-Trapdoor, M-Test). MainScheme uses a cryptographic hash function $H : \{0,1\}^k \to \mathbb{Z}_n^{+1}$ (which is assumed in the analysis to behave as a random oracle). We denote by m the security parameter and by k the length of keywords. We assume wlog that $k = \Theta(m^c)$ for some constant $c > 0$ (concrete values for m, k can be $m = 1024$ and $k = 160$). MainScheme can be described as follows:

```
M-KeyGen($1^m$): On input security parameter $1^m$ in unary, for $m \in \mathbb{Z}^+$, do the
    following:
    1. randomly choose two primes $p, q$ of length $m/2$, and such that both $p$
       and $q$ are congruent to 3 mod 4 and set $n = pq$;
    2. Set $A_{pub} = (n, 1^k)$ and $A_{priv} = (p, q)$, and output: $(A_{pub}, A_{priv})$.
M-ksEnc($A_{pub}, W$): Let $A_{pub} = (n, 1^k)$, $W \in \{0,1\}^k$, and do the following:
    1. For each $i = 1, \ldots, 4k$,
          compute $h_i = H(W|i)$;
          randomly and independently choose $u_i \in \mathbb{Z}_n^*$;
          if $\left(\frac{u_i^2 - 4h_i}{n}\right) = +1$ then
             randomly and independently choose $t_i \in \mathbb{Z}_n^{+1}$;
             set $s_i = (t_i + h_i/t_i) \mod n$.
          if $\left(\frac{u_i^2 - 4h_i}{n}\right) \in \{-1, 0\}$ then set $s_i = u_i$.
    2. Output $\boldsymbol{s} = (s_1, \ldots, s_{4k})$.
M-Trapdoor($A_{priv}, W$): Let $A_{priv} = (p, q)$, $W \in \{0,1\}^k$, and do the following:
    1. For $i = 1, \ldots, 4k$;
          compute $h_i = H(W|i)$;
          use $p, q$ to randomly choose $g_i \in \mathbb{Z}_n^*$ (if any) such that $g_i^2 = h_i \mod n$;
          if $h_i$ has no square root modulo $n$, then set $g_i = \perp$;
    2. return: $(g_1, \ldots, g_{4k})$.
```

```
M-Test($A_{pub}, s, T_W$): Let $T_W = (g_1, \ldots, g_{4k})$ and $\boldsymbol{s} = (s_1, \ldots, s_{4k})$, and do the following:
  1. For $i = 1, \ldots, 4k$,
       if $g_i = \perp$ then set $\bar{t}_i = \perp$;
       if $g_i^2 = h_i \mod n$ then
         if $\left(\frac{s_i^2-4h_i}{n}\right) = +1$ then set $\bar{t}_i = \left(\frac{s_i+2g_i}{n}\right)$;
           otherwise set $\bar{t}_i = \perp$;
  2. output 'yes' if $\bar{t}_i \in \{+1, \perp\}$ for all $i = 1, \ldots, 4k$; otherwise output 'no'.
```

Remarks: ciphertext distribution and scheme parameters. We note the distribution of the ciphertext $\boldsymbol{s} = (s_1, \ldots, s_{4k})$ returned by algorithm M-ksEnc has only negligible statistical distance from the distribution where each element s_i is uniformly distributed among the integers such that $s_i^2 - 4h_i \in QR(n)$ with probability 1/2, or $s_i^2 - 4h_i \in \mathbb{Z}_n^{-1}$ with probability 1/2.

We note that it is essential to choose our scheme's parameter $k = \Theta(m^c)$, for some $c > 0$, to guarantee that the consistency properties of MainScheme are satisfied in an asymptotic sense. Good practical choices for parameters m, k include setting $m = 1024$ and $k = 160$.

4 Properties of Our Construction

In Subsections 4.1 and 4.2 we prove the consistency and security properties of our public-key encryption scheme with keyword search.

4.1 Proof of Consistency

We prove the right-keyword consistency here and omit the proof of adversary-based consistency of MainScheme to meet space constraints.

Right-keyword consistency. For $i = 1, \ldots, 4k$, whenever $\left(\frac{s_i^2-4h_i}{n}\right) = +1$, it always holds that $h_i = g_i^2 \mod n$ and it never holds that $\bar{t}_i = \left(\frac{s_i+2g_i}{n}\right) = -1$. The latter fact is proved by observing that, for $i = 1, \ldots, 4k$, it holds that

$$s_i + 2g_i = t_i + h_i/t_i + 2g_i = t_i \cdot (1 + g_i^2/t_i{}^2 + 2g_i/t_i) = t_i \cdot (1 + g_i/t_i)^2 \mod n,$$

and thus, except with negligible probability,

$$\bar{t}_i = \left(\frac{s_i + 2g_i}{n}\right) = \left(\frac{t_i}{n}\right) = +1.$$

Now, the above equalities do not hold only when $s_i + 2g_i \mod n$ is not in $\mathbb{Z}_n^*$, in which case it still holds that $\left(\frac{s_i+2g_i}{n}\right) = 0 \neq -1$. As a consequence of these two facts, the right-keyword consistency property holds with probability 1.

4.2 Proof of Security

Let $\mathcal{A}$ be a polynomial-time algorithm that attacks MainScheme and succeeds in breaking with advantage ϵ, and while doing that, it makes at most $q_H > 0$ queries to the random oracle H and at most $q_T > 0$ trapdoor queries. We would like to show that ϵ is negligible in m or otherwise $\mathcal{A}$ can be used to construct an algorithm $\mathcal{B}$ that violates the intractability of the QIP problem. More precisely, we will attempt to violate the intractability of one among two problems that we call QIP$_1$ and QIP$_2$, and that are easily seen to be computationally equivalent to QIP.

We prove this by defining a sequence of games, which we call 'MainScheme Security Game t', for $t = 0, \ldots, 4k$, which are all variations of the PEKS Security Game defined in Section 2.

MainScheme **Security Game** t

1. Algorithm $\mathcal{B}$ takes as input (n, h_0, h_1, s), where n is a Blum-Williams integer, and $h_0, h_1 \in \mathbb{Z}_n^{+1}$ and $s \in \mathbb{Z}_n^*$.
2. First of all $\mathcal{B}$ runs the M-KeyGen(1^m) algorithm to generate $A_{pub} = (n, 1^k)$ and $A_{priv} = (p, q)$; afterwards, it gives A_{pub} to the attacker $\mathcal{A}$.
3. $\mathcal{A}$ can adaptively ask for outputs from the random oracle H to any inputs of its choice. To respond to H-queries, algorithm $\mathcal{B}$ maintains a list of tuples $\langle W_i, j, h_{i,j}, g_{i,j}, d(i,j), c(i,j)\rangle$ called the H-list. The list is initially empty. When $\mathcal{A}$ queries the random oracle H at a point $(W_i|j)$, for $W_i \in \{0,1\}^k$ and $j \in \{1, \ldots, 4k\}$, algorithm $\mathcal{B}$ responds as follows.

   If tuple $\langle W_i, j, h_{i,j}, g_{i,j}, d(i,j), c(i,j)\rangle$ appears on the H-list then algorithm $\mathcal{B}$ responds with $H(W_i|j) = h_{i,j} \in \mathbb{Z}_n^{+1}$.

   Otherwise, $\mathcal{B}$ uniformly chooses $d(i,j) \in \{0,1\}$, $r_{i,j} \in \mathbb{Z}_n^*$, and randomly choose $c(i,j) \in \{0,1\}$ such that $c(i,j) = 0$ with probability $1/(q_T + 1)$ and $c(i,j) = 1$ with probability $1 - 1/(q_T + 1)$.

   If $c(i,j) = 1$ then $\mathcal{B}$ computes $h_{i,j} = (-1)^{d(i,j)} \cdot r_{i,j}^2 \mod n$; sets $g_{i,j} = \perp$ if $d(i,j) = 1$, or $g_{i,j} = r_{i,j}$ if $d(i,j) = 0$; adds $\langle W_i, j, h_{i,j}, g_{i,j}, d(i,j), c(i,j)\rangle$ to the H-list and responds with $h_{i,j}$ to the H-query $(W_i|j)$.

   If $c(i,j) = 0$, then $\mathcal{B}$ sets $d = d(i,j)$, computes $h_{i,j} = h_d \cdot r_{i,j}^2 \mod n$, sets $g_{i,j} = r_{i,j}$, adds $\langle W_i, j, h_{i,j}, g_{i,j}, d(i,j), c(i,j)\rangle$ to the H-list and responds with $h_{i,j}$ to the H-query $(W_i|j)$.
4. $\mathcal{A}$ can adaptively ask for the trapdoor T_W for any keyword $W \in \{0,1\}^k$ of his choice, to which $\mathcal{B}$ responds as follows.

   If tuple $\langle W_i, j, h_{i,j}, g_{i,j}, d(i,j), c(i,j)\rangle$ already appears on the H-list, for some $j \in \{1, \ldots, 4k\}$ and $W_i = W$, then $\mathcal{B}$ responds with $(g_{i,1}, \ldots, g_{i,4k})$ to the trapdoor query W if $c(i,j) = 1$ or reports failures and halts if $c(i,j) = 0$.

   Otherwise $\mathcal{B}$ randomly chooses $d(i,j) \in \{0,1\}$ and $r_{i,j} \in \mathbb{Z}_n^*$; computes $h_{i,j} = (-1)^{d(i,j)} \cdot r_{i,j}^2 \mod n$; sets $g_{i,j} = \perp$ if $d(i,j) = 1$ or $g_{i,j} = r_{i,j}$ otherwise, responds with $(g_{i,1}, \ldots, g_{i,4k})$ to the trapdoor query W and inserts $\langle W_i, j, h_{i,j}, g_{i,j}, d(i,j), c(i,j)\rangle$ in H-list.
5. The attacker $\mathcal{A}$ sends the two keywords W_0, W_1 on which it wishes to be challenged (for which it did not previously ask for trapdoors T_{W_0} or T_{W_1}).

If the two tuples

$$\langle W_u, j, h_{u,j}, g_{u,j}, d(u,j), c(u,j)\rangle \text{ and } \langle W_v, j, h_{v,j}, g_{v,j}, d(v,j), c(v,j)\rangle$$

satisfying $W_u = W_0$, $W_v = W_1$, $j = t$, and $((c(u,j) = 0) \vee (c(v,j) = 0))$, are not in H-list, then $\mathcal{B}$ reports failures and halts.

Otherwise $\mathcal{B}$ computes $(s_1, \ldots, s_{4k})$ as follows:
- $s_1, \ldots, s_{t-1}$ are computed as from algorithm M-ksEnc on input A_{pub}, W_0;
- s_t is set equal to $s \cdot r_{i,t} \mod n$;
- $s_{t+1}, \ldots, s_{4k}$ are computed as from algorithm M-ksEnc on input A_{pub}, W_1.

6. Given challenge $(s_1, \ldots, s_{4k})$, $\mathcal{A}$ can continue to ask for random oracle H's outputs for any input of its choice, and for trapdoors T_W for any keyword W of his choice as long as $W \neq W_0, W_1$; these are answered as in items 3 and 4, respectively.
7. $\mathcal{A}$ outputs $out \in \{0,1\}$.

By using a standard hybrid argument on our assumption that $\mathcal{A}$ breaks the security of MainScheme with probability ϵ, we obtain that there exists $t \in \{1, \ldots, 4k\}$ such that

$$|\operatorname{Prob}[\mathcal{A}_t = 1] - \operatorname{Prob}[\mathcal{A}_{t+1} = 1]| \geq \epsilon/4k, \tag{3}$$

where by $\mathcal{A}_t = 1$ we denote the event that $\mathcal{A}$ returns 1 in the real attack game given that the challenge ciphertext $\boldsymbol{s}$ had been computed as follows: $s_1, \ldots, s_t$ are computed as in algorithm M-ksEnc on input A_{pub}, W_0; and $s_{t+1}, \ldots, s_{4k}$ are computed as in algorithm M-ksEnc on input A_{pub}, W_1. Similarly, as in [5], we can obtain that with probability at least $\epsilon/4k$, $\mathcal{A}$ queries at least one of the two H-queries $(W_0|t)$, $(W_1|t)$. (The proof is omitted for lack of space.)

The proof continues by considering two cases according to whether only one of the two queries are made or both of them are made. In the first case, we show that $\mathcal{B}$ violates the intractability of the QIP$_1$ problem, and in the second case we show that it violates the intractability of the QIP$_2$ problem. Both the QIP$_1$ and QIP$_2$ are minor variants of the QIP problem and easily seen to be computationally equivalent to it.

Case (a). We now consider the case when only one of the two H-queries; say, $(W_0|t)$, is made by algorithm $\mathcal{A}$. We define the QIP$_1$ *problem* as the problem of efficiently distinguishing the following two distributions:

$$\begin{aligned} D_{1,0}(1^m) = \{&(p,q) \leftarrow \mathrm{BW}(1^m); n \leftarrow p \cdot q; d \leftarrow \{0,1\}; h_0, h_1 \leftarrow \mathbb{Z}_n^{+1}; \\ &s \leftarrow \mathrm{CS}(n, s^2 - 4h_d \in \mathbb{Z}_n^{-1} \cup QR(n)) \,:\, (n, h_0, h_1, s)\} \\ D_{1,1}(1^m) = \{&(p,q) \leftarrow \mathrm{BW}(1^m); n \leftarrow p \cdot q; h_0, h_1 \leftarrow \mathbb{Z}_n^{+1}; s \leftarrow \mathbb{Z}_n^* \,:\, (n, h_0, h_1, s)\} \end{aligned}$$

We say that algorithm $\mathcal{A}$ has *advantage* ϵ in solving QIP$_1$ if we have that:

$$\begin{aligned} \big| &\Pr[(n, h_0, h_1, s) \leftarrow D_{1,0}(1^m) : \mathcal{A}(n, h_0, h_1, s) = 1] \\ &- \Pr[(n, h_0, h_1, s) \leftarrow D_{1,1}(1^m) : \mathcal{A}(n, h_0, h_1, s) = 1]\big| = \epsilon. \end{aligned} \tag{4}$$

We say that QIP$_1$ is *intractable* if all polynomial time (in m) algorithms have a negligible (in m) advantage in solving QIP$_1$.

By a simple simulation argument, we can prove the following theorem:

Theorem 2. The QIP_1 problem is intractable if and only if the QIP problem is so.

We continue the proof by noting that bit $c(i,t)$ associated to the query $(W_0|t)$, where the i-th queried keyword is W_0, satisfies $c(i,t) = 0$ with probability $1/(q_T + 1)$. Assuming that $c(i,t) = 0$, we evaluate the distribution of ciphertext $\boldsymbol{s}$ in MainScheme Security Game t, for $t = 1, \ldots, 4k$.

First, we let $d = d(i,t)$ and observe that when $(n, h_0, h_1, s) \in D_{1,0}(1^m)$, the ciphertext $\boldsymbol{s}$ in MainScheme Security Game t appears to $\mathcal{A}$ to be distributed exactly as if $s_1, \ldots, s_t$ were computed as in algorithm M-ksEnc on input A_{pub}, W_0, and $s_{t+1}, \ldots, s_{4k}$ were computed as in algorithm M-ksEnc on input A_{pub}, W_1. This can be seen by observing that we assumed that $c(i,t) = 0$ and thus $H(W_0|t) = h_d \cdot r_{i,t}^2$; then, it holds that s_t is randomly distributed among the integers such that $s_t^2 - 4H(W_0|t) \in \mathbb{Z}_n^{-1} \cup QR(n))$ as it satisfies $s_t^2 - 4H(W_0|t) = (sr_{i,t})^2 - 4h_d r_{i,t}^2 = r_{i,t}^2(s^2 - 4h_d)$, where $s^2 - 4h_d$ is also randomly distributed among the integers in $\mathbb{Z}_n^{-1} \cup QR(n))$ as $(n, h_0, h_1, s) \in D_{1,0}(1^m)$. Therefore, the probability that $\mathcal{A}$ returns 1 in MainScheme Security Game t when $(n, h_0, h_1, s) \in D_{1,0}(1^m)$ is the same as the probability that $\mathcal{A}_t = 1$.

We now consider the case when $(n, h_0, h_1, s) \in D_{1,1}(1^m)$, the ciphertext $\boldsymbol{s}$ in MainScheme Security Game t appears to $\mathcal{A}$ to be distributed exactly as if $s_1, \ldots, s_{t-1}$ were computed as in algorithm M-ksEnc on input A_{pub}, W_0, and $s_t, \ldots, s_{4k}$ were computed as in algorithm M-ksEnc on input A_{pub}, W_1. This can be seen by observing that s_t is uniformly distributed in $\mathbb{Z}_n^*$ by definition of $D_{1,1}$, and that if s_t were computed as in algorithm M-ksEnc on input A_{pub}, W_1, it would appear to $\mathcal{A}$ to have the same distribution, as we assumed that $(W_1|t)$ was not queried by $\mathcal{A}$. Therefore, the probability that $\mathcal{A}$ returns 1 in MainScheme Security Game t when $(n, h_0, h_1, s) \in D_{1,1}(1^m)$ is the same as the probability that $\mathcal{A}_{t-1} = 1$.

This implies that the probability that $\mathcal{B}$ distinguishes $D_{1,0}(1^m)$ from $D_{1,1}(1^m)$ is the probability $1/(e \cdot q_T)$ that $\mathcal{B}$ does not halt in MainScheme Security Game t, times the probability $\epsilon/(4k \cdot (q_T + 1))$ that $\mathcal{A}$ makes only one H-queries among $(H_0|t), (H_1|t)$ and it holds that the associated bit $c_{\cdot,t} = 0$.

Since ϵ is assumed to be not negligible, then so is the quantity $\epsilon/(e \cdot 4k \cdot (q_T + 1))$, and therefore $\mathcal{B}$ violates the intractability of the QIP_1 problem.

Case (b). We now consider the case when both H-queries $(W_0|t)$, $(W_1|t)$ are made by algorithm $\mathcal{A}$. We define the QIP_2 *problem* as the problem of efficiently distinguishing the following two distributions:

$$\begin{aligned} D_{2,0}(1^m) = \{&(p,q) \leftarrow \text{BW}(1^m); n \leftarrow p \cdot q; h_0, h_1 \leftarrow \mathbb{Z}_n^{+1}; \\ &s \leftarrow \text{CS}(n, s^2 - 4h_0 \in \mathbb{Z}_n^{-1} \cup QR(n)) \,:\, (n, h_0, h_1, s)\} \\ D_{2,1}(1^m) = \{&(p,q) \leftarrow \text{BW}(1^m); n \leftarrow p \cdot q; h_0, h_1 \leftarrow \mathbb{Z}_n^{+1}; \\ &s \leftarrow \text{CS}(n, s^2 - 4h_1 \in \mathbb{Z}_n^{-1} \cup QR(n)) \,:\, (n, h_0, h_1, s)\} \end{aligned}$$

We say that algorithm $\mathcal{A}$ has *advantage* ϵ in solving QIP$_2$ if we have that:

$$\Big| \Pr[(n,h_0,h_1,s) \leftarrow D_{2,0}(1^m) : \mathcal{A}(n,h_0,h_1,s)=1] \\ - \Pr[(n,h_0,h_1,s) \leftarrow D_{2,1}(1^m) : \mathcal{A}(n,h_0,h_1,s)=1]\Big| = \epsilon. \quad (5)$$

We say that QIP$_2$ is *intractable* if all polynomial time (in m) algorithms have a negligible (in m) advantage in solving QIP$_2$.

By a simple hybrid argument, we can prove the following theorem:

Theorem 3. The QIP$_2$ problem is intractable if and only if the QIP problem is so.

We continue the proof by noting that bits $c(i,t), c(j,t)$ associated to the two queries, where the i-th queried keyword is W_i and the j-th queried keyword is W_1, satisfy $c(i,t) = c(j,t) = 0$ with probability at least $1/(q_T+1)^2$. Under this setting, we evaluate the distribution of ciphertext $\boldsymbol{s}$ in MainScheme Security Game t, for $t = 1, \ldots, 4k$.

First, we observe that when $(n,h_0,h_1,s) \in D_{2,0}(1^m)$, the ciphertext $\boldsymbol{s}$ in MainScheme Security Game t appears to $\mathcal{A}$ to be distributed exactly as if $s_1, \ldots, s_t$ were computed as in algorithm M-ksEnc on input A_{pub}, W_0, and $s_{t+1}, \ldots, s_{4k}$ were computed as in algorithm M-ksEnc on input A_{pub}, W_1. This can be seen by observing that we assumed that $c(i,t) = 0$ and thus $H(W_0|t) = h_0 \cdot r_{i,t}^2$; then, it holds that s_t is randomly distributed among the integers such that $s_t^2 - 4H(W_0|t) \in \mathbb{Z}_n^{-1} \cup QR(n))$ as it satisfies $s_t^2 - 4H(W_0|t) = (sr_{i,t})^2 - 4h_0 r_{i,t}^2 = r_{i,t}^2(s^2 - 4h_0)$, where $s^2 - 4h_0$ is also randomly distributed among the integers in $\mathbb{Z}_n^{-1} \cup QR(n))$ as $s \in D_{2,0}(1^m)$. Therefore, the probability that $\mathcal{A}$ returns 1 in MainScheme Security Game t when $(n,h_0,h_1,s) \in D_{2,0}(1^m)$ is the same as the probability that $\mathcal{A}_t = 1$.

Analogously, when $(n,h_0,h_1,s) \in D_{2,1}(1^m)$, the ciphertext $\boldsymbol{s}$ in MainScheme Security Game t appears to $\mathcal{A}$ to be distributed exactly as if $s_1, \ldots, s_{t-1}$ were computed as in algorithm M-ksEnc on input A_{pub}, W_0, and $s_t, \ldots, s_{4k}$ were computed as in algorithm M-ksEnc on input A_{pub}, W_1. This can be seen as before by again observing that we assumed that $c(j,t) = 0$ and thus $H(W_1|t) = h_1 \cdot r_{j,t}^2$; then, it holds that s_t is randomly distributed among the integers such that $s_t^2 - 4H(W_1|t) \in \mathbb{Z}_n^{-1} \cup QR(n))$ as it satisfies $s_t^2 - 4H(W_1|t) = (sr_{j,t})^2 - 4h_1 r_{j,t}^2 = r_{j,t}^2(s^2 - 4h_1)$, where $s^2 - 4h_1$ is also randomly distributed among the integers in $\mathbb{Z}_n^{-1} \cup QR(n))$ as $s \in D_{2,1}(1^m)$. Therefore, the probability that $\mathcal{A}$ returns 1 in MainScheme Security Game t when $(n,h_0,h_1,s) \in D_{2,1}(1^m)$ is the same as the probability that $\mathcal{A}_{t-1} = 1$.

This implies that $\mathcal{B}$ distinguishes $D_{2,0}(1^m)$ from $D_{2,1}(1^m)$ is the probability $1/(e \cdot q_T)$ that $\mathcal{B}$ does not halt in MainScheme Security Game t, times the probability $\epsilon/(4k \cdot (q_T+1)^2)$ that $\mathcal{A}$ makes both H-queries $(H_0|t), (H_1|t)$ and it holds that $c_{i,t} = c_{j,t} = 0$.

Since ϵ is assumed to be not negligible, then so is the quantity $\epsilon/(e \cdot 4k \cdot q_T (q_T+1)^2)$, and therefore $\mathcal{B}$ violates the intractability of the QIP$_2$ problem.

Acknowledgements

We thank Fadil Santosa, Minnesota Center for Industrial Mathematics, University of Minnesota, and Andrew Odlyzko, Digital Technology Center, University of Minnesota, for support and interesting discussions.

References

1. 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, Springer, Heidelberg (2005)
2. Bellare, M., Boldyreva, A., Desai, A., Pointcheval, D.: Key-Privacy in Public-Key Encryption. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, Springer, Heidelberg (2001)
3. Boneh, D.: Private communication (February 2007)
4. Boneh, D.: Private communication (August 2007)
5. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public Key Encryption with Keyword Search. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (2004)
6. Boneh, D., Franklin, M.: Identity-based Encryption from the Weil Pairing. SIAM J. of Computing 32(3), 586–615 (2003) (Extended abstract in Crypto 2001)
7. Boneh, D., Gentry, C., Hamburg, M.: Space-Efficient Identity Based Encryption Without Pairings (in submission)
8. Boyen, X., Waters, B.: Anonymous Hierarchical Identity-Based Encryption (without Random Oracles). In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, Springer, Heidelberg (2006)
9. Cocks, C.: An Identity Based Encryption Echeme based on Quadratic Residues. In: Eighth IMA International Conference on Cryptography and Coding, Royal Agricultural College, Cirencester, UK (December 2001)
10. Cohen, H.: A Course in Computational Algebraic Number Theory. In: Graduate Texts in Mathematics, vol. 138, Springer, Heidelberg (1993)
11. Coron, J.: On the Exact Security of Full-Domain-Hash. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 229–235. Springer, Heidelberg (2000)
12. Dolev, D., Dwork, C., Naor, M.: Non-Malleable Cryptography. SIAM Journal on Computing (2000) Early version in Proc. of STOC 1991
13. Golle, P., Staddon, J., Waters, B.R.: Secure Conjunctive Keyword Search over Encrypted Data. In: Jakobsson, M., Yung, M., Zhou, J. (eds.) ACNS 2004. LNCS, vol. 3089, Springer, Heidelberg (2004)
14. Maniatis, P., Roussopoulos, M., Swierk, E., Lai, K., Appenzeller, G., Zhao, X., Bake, M.: The Mobile People Architecture. ACM Mobile Computing and Communications Review (MC2R) 3(3) (July 1999)
15. Park, D.J., Kim, K., Lee, P.J.: Public Key Encryption with Conjunctive Keyword Search. In: Lim, C.H., Yung, M. (eds.) WISA 2004. LNCS, vol. 3325, Springer, Heidelberg (2005)
16. Shamir, A.: Identity-based Cryptosystems and Signature Schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, Springer, Heidelberg (1985)
17. Waters, B., Balfanz, D., Durfee, G., Smetters, D.: Building an Encrypted and Searchable Audit Log. In: Proc. of NDSS 2004 (2004)