Searchable Public Key Encryption

Dan Boneh[*]
dabo@cs.stanford.edu

Giovanni Di Crescenzo
giovanni@research.telcordia.com

Rafail Ostrovsky
rafail@research.telcordia.com

Giuseppe Persiano
pino.persiano@unisa.it

Abstract

We study the problem of searching on data that is encrypted using a public key system. Consider user Bob who sends email to user Alice encrypted under Alice's public key. An email gateway wants to test whether the email contains the keyword "urgent" so that it could route the email to Alice's pager. We construct a mechanism that enables Alice to provide a key that enables the gateway to test whether the word "urgent" is a keyword in the email without learning anything else about the email. We refer to this mechanism as *Searchable Public Key Encryption*. Similarly, consider a mail server that stores various messages publicly encrypted for Alice by others. Using our mechanism Alice can send the mail server a key that will enable the server to identify all messages containing some specific keyword, but learn nothing else. We precisely define the concept of searchable public key encryption and give three constructions.

1 Introduction

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. The mobile people project [13] provides this email processing capability.

Now, suppose Bob sends encrypted email to Alice. As usual, Bob encrypts the email and all keywords using Alice's public key. In this case the mail gateway cannot see the keywords and hence cannot make routing decisions. As a result, the mobile people project is unable to process secure email without violating user privacy. Our goal is to enable the gateway 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. We give precise definitions in section 2.

To do so, Bob encrypts his email using a standard public key system. He then appends to the resulting ciphertext a *searchable public-key encryption* (SPKE) of each keyword. To send a message M with keywords $W_1, \ldots, W_m$ Bob sends

$$E_{A_{pub}}(M) \;\|\; \mathsf{SPKE}(A_{pub}, W_1) \;\|\; \cdots \;\|\; \mathsf{SPKE}(A_{pub}, W_m)$$

[*]Supported by NSF and the Packard foundation.

Where A_{pub} is Alice's public key. The point of searchable encryption is that Alice can give the gateway a certain trapdoor T_W that enables the gateway to test whether one of the keywords associated with the message is equal to the word W of Alice's choice. Given $\mathsf{SPKE}(A_{pub}, W')$ and T_W the gateway can test whether $W = W'$. If $W \neq W'$ the gateway learns nothing more about W'. Note that Alice and Bob do not communicate in this entire process. Bob generates the searchable encryption for W' just given Alice's public key.

In some cases, it is instructive to view the email gateway as an IMAP or POP email server. The server stores many emails and each email contains a small number of keywords. As before, all these emails are created by various people sending mail to Alice encrypted using her public key. We want to enable Alice to ask queries of the form: do any of the messages on the server contain the keyword "urgent"? Alice would do this by giving the server a trapdoor T_W, thus enabling the server to retrieve emails containing the keyword W. The server learns nothing else about the emails.

Related work. Song et al. [15] study the problem of searching on data encrypted using a secret *symmetric* key. This comes up in the context of an encrypted file server where a user wants to search her own encrypted files for a specific word. The encrypted files were initially encrypted by the user and the user is the one issuing the search queries. We study the public-key version of this problem where data on a mail server is encrypted by various people using the user's public key, while enabling the user to issue search queries for specific keywords. Due to the computation cost of public key encryption, our constructions only enable searching on a small number of file keywords, where as in the symmetric key settings one can afford to search on an entire encrypted file. We note that searching on encrypted data (both in the symmetric and public key settings) can potentially be solved using general two-party computation techniques [18, 8]. However, the resulting protocols would require many rounds of communication and would most likely be impractical. In this paper we construct searchable public-key encryption schemes that are both efficient and non-interactive. Recently, Waters et al. [17] showed that searchable public key encryption can be used to build an encrypted and searchable audit log.

Throughout the paper we use the term *negligible function* to refer to a function $f : \mathbb{R} \to [0, 1]$ such that $f(s) < 1/g(s)$ for any polynomial g and sufficiently large s.

2 Searchable public key encryption: definitions

We start by precisely defining what is a secure Searchable Public Key Encryption (SPKE) scheme. Here "public-key" refers to the fact that ciphertexts are created by various people using Alice's public key. Suppose user Bob is about to send an encrypted email to Alice with keywords $W_1, \ldots, W_k$ (words in the subject line and the sender's address could be used as keywords, so that k is relatively small). Bob sends the following message:

$$\left[E_{A_{pub}}[msg],\ \mathsf{SPKE}(A_{pub}, W_1), \ldots, \mathsf{SPKE}(A_{pub}, W_k)\right]$$

where A_{pub} is Alice's public key, msg is the email body, and SPKE is an algorithm with properties discussed below. The SPKE values do not reveal any information about the message, but enable searching for specific keywords. For the rest of the paper, we use as our sample application a mail server that stores all incoming email.

Our goal is to enable Alice to send a short secret key T_W to the mail server that will enable the server to locate all messages containing the keyword W, but learn nothing else. Alice produces this trapdoor T_W using her private key. The server simply sends the relevant emails back to Alice. We call such a system *non-interactive public key searchable encryption.*

Definition 2.1. A non-interactive public key searchable encryption scheme consists of the following polynomial time randomized algorithms:

1. $\mathsf{KeyGen}(s)$: Takes a security parameter, s, and generates a public/private key pair A_{pub}, A_{priv}.
2. $\mathsf{SPKE}(A_{pub}, W)$: for a public key A_{pub} and a word W, produces a searchable encryption of W.
3. $\mathsf{Trapdoor}(A_{priv}, W)$: given Alice's private key and a word W produces a trapdoor T_W.
4. $\mathsf{Test}(A_{pub}, S, T_W)$: given Alice's public key, a searchable encryption $S = \mathsf{SPKE}(A_{pub}, W')$, and a trapdoor $T_W = \mathsf{Trapdoor}(A_{priv}, W)$, outputs 'yes' if $W = W'$ and 'no' otherwise.

Alice runs the KeyGen algorithm to generate her public/private key pair. She uses Trapdoor to generate trapdoors T_W for any keywords W that she wants the mail server or mail gateway to search for. The mail server uses the given trapdoors as input to the $\mathsf{Test}()$ algorithm to determine whether a given email contains one of the keywords W specified by Alice.

Next, we define security for an SPKE. We need to ensure that an $\mathsf{SPKE}(A_{pub}, W)$ does not reveal any information about W, unless T_W is available. We define security against an active attacker who is able to obtain trapdoors T_W for any W of his choice. Even under such attack the attacker should not be able to search for a keyword W' 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 (the security parameter s is given to both players as input).
SPKE Security game:

1. The challenger runs the $\mathsf{KeyGen}(s)$ 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 $W \in \{0,1\}^*$ of his choice.
3. At some point, the attacker $\mathcal{A}$ sends the challenger two words 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 $C = \mathsf{SPKE}(A_{pub}, W_b)$. We refer to C as the challenge SPKE.
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 $b' \in \{0,1\}$ and wins the game if $b = b'$.

In other words, the attacker wins the game if he can correctly guess whether he was given the SPKE for W_0 or W_1. We define $\mathcal{A}$'s advantage in breaking the SPKE as

$$\mathrm{Adv}_{\mathcal{A}}(s) = |\Pr[b = b'] - \frac{1}{2}|$$

Definition 2.2. We say that the non-interactive searchable public-key encryption scheme SPKE is semantically secure against an adaptive chosen keyword attack if for any polynomial time attacker $\mathcal{A}$ we have that $\mathrm{Adv}_{\mathcal{A}}(s)$ is a negligible function.

2.1 SPKE implies Identity Based Encryption

Searchable public key encryption is related to Identity Based Encryption (IBE) [16, 1]. Constructing a secure SPKE appears to be a harder problem than constructing an IBE. Indeed, the following lemma shows that SPKE implies Identity Based Encryption. The converse is probably false, as discussed below. Security notions for IBE, and in particular chosen ciphertext secure IBE (IND-ID-CCA), are defined in [1].

Lemma 2.3. *A non-interactive searchable encryption scheme (*SPKE*) that is semantically secure against an adaptive chosen keyword attack gives rise to a chosen ciphertext secure IBE system (IND-ID-CCA).*

Proof sketch: Given an SPKE $(\mathsf{KeyGen}, \mathsf{SPKE}, \mathsf{Trapdoor}, \mathsf{Test})$ the IBE system is as follows:

1. Setup: Run the SPKE KeyGen algorithm to generate A_{pub}/A_{priv}. The IBE system parameters are A_{pub}. The master-key is A_{priv}.
2. KeyGen: The IBE private key associated with a public key $X \in \{0,1\}^*$ is
$$d_X = [\mathsf{Trapdoor}(A_{priv},\ X\|0),\ \ \mathsf{Trapdoor}(A_{priv},\ X\|1)]\,,$$
where $\|$ denotes concatenation.
3. $\mathsf{Encrypt}$: Encrypt a bit $b \in \{0,1\}$ using a public key $X \in \{0,1\}^*$ as: $CT = \mathsf{SPKE}(A_{pub},\ X\|b)$.
4. $\mathsf{Decrypt}$: To decrypt $CT = \mathsf{SPKE}(A_{pub}, X\|b)$ using the private key $d_X = (d_0, d_1)$ output '0' if $\mathsf{Test}(A_{pub}, CT, d_0) =$ 'yes'. Output '1' if $\mathsf{Test}(A_{pub}, CT, d_1) =$ 'yes'.

One can show that the resulting system is IND-ID-CCA assuming the SPKE is semantically secure against an adaptive chosen message attack. □

This shows that building non-interactive public-key searchable encryption is at least as hard as building an IBE system. One might be tempted to prove the converse (i.e., IBE implies SPKE) by defining

$$\mathsf{SPKE}(A_{pub}, W) = E_W[0^k] \tag{1}$$

i.e. encrypt a string of k zeros with the IBE public key $W \in \{0,1\}^*$. The Test algorithm attempts to decrypt $E_W[0]$ and checks that the resulting plaintext is 0^k. Unfortunately, this does not necessarily give a secure searchable encryption scheme. The problem is that the ciphertext CT could expose the public key (W) used to create CT. Generally, an encryption scheme need not hide the public key that was used to create a given ciphertext. But this property is essential for the SPKE construction given in (1).

Generally, it appears that constructing a searchable public-key encryption is a harder problem than constructing an IBE scheme. Nevertheless, our first two SPKE constructions are based on recent constructions for IBE systems. We are able to prove security by exploiting extra properties of these systems. We note that for some of these constructions security is based on stronger complexity assumptions than the corresponding IBE systems. This is consistent with our claim that building an SPKE is harder than building an IBE.

3 Constructions

We give three constructions for public-key searchable encryption: (1) an efficient system based on a variant of the Decision Diffie-Hellman assumption, (2) a somewhat less efficient system using elements modulo a composite, and (3) a system based on general trapdoor permutations.

3.1 Construction using bilinear maps

Our first construction is based on a variant of the Diffie-Hellman problem. Boneh and Franklin [1] recently used bilinear maps on elliptic curves to build an efficient IBE system. Abstractly, they use two groups G_1, G_2 of prime order p and a bilinear map $e : G_1 \times G_1 \to G_2$ between them. The map satisfies the following properties:

1. Computable: given $g, h \in G_1$ there is a polynomial time algorithms to compute $e(g, h) \in G_2$.
2. Bilinear: for any integers $x, y \in [1, p]$ we have $e(g^x, g^y) = e(g, g)^{xy}$
3. Non-degenerate: if g is a generator of G_1 then $e(g, g)$ is a generator of G_2.

The size of G_1, G_2 is determined by the security parameter.

We build a non-interactive searchable encryption scheme from such a bilinear map. The construction is based on [1]. We will need hash functions $H_1 : \{0,1\}^* \to G_1$ and $H_2 : G_2 \to \{0,1\}^{\log p}$. Our SPKE works as follows:

- KeyGen: The input security parameter determines the size, p, of the groups G_1 and G_2. The algorithm picks a random $\alpha \in \mathbb{Z}_p^*$ and a generator g of G_1. It outputs $A_{pub} = [g, h = g^\alpha]$ and $A_{priv} = \alpha$.

- $\mathsf{SPKE}(A_{pub}, W)$: First compute $t = e(H_1(W), h^r) \in G_2$ for a random $r \in \mathbb{Z}_p^*$.
Output $\mathsf{SPKE}(A_{pub}, W) = [g^r,\ H_2(t)]$.

- $\mathsf{Trapdoor}(A_{priv}, W)$: output $T_W = H_1(W)^\alpha \in G_1$.

- $\mathsf{Test}(A_{pub}, S, T_W)$: let $S = [A, B]$. Test if $H_2(e(T_W, A)) = B$.
If so, output 'yes'; if not, output 'no'.

We prove that this system is a non-interactive searchable encryption scheme semantically secure against a chosen keyword attack in the random oracle model. The proof of security relies on the difficulty of the Bilinear Diffie-Hellman problem (BDH) [1, 11].

Bilinear Diffie-Hellman Problem (BDH): Fix a generator g of G_1. The BDH problem is as follows: given $g, g^a, g^b, g^c \in G_1$ as input, compute $e(g, g)^{abc} \in G_2$. We say that BDH is intractable if all polynomial time algorithms have a negligible advantage in solving BDH.

We note that the Boneh-Franklin IBE system [1] relies on the same intractability assumption for security. The security of our SPKE is proved in the following theorem. The proof is set in the random oracle model. Indeed, it is currently an open problem to build a secure IBE, and hence an SPKE, without the random oracle model.

Theorem 3.1. *The non-interactive searchable encryption scheme (SPKE) above is semantically secure against a chosen keyword attack in the random oracle model assuming BDH is intractable.*

Proof. Suppose $\mathcal{A}$ is an attack algorithm that has advantage ϵ in breaking the SPKE. Suppose $\mathcal{A}$ makes at most q_{H_2} hash function queries to H_2 and at most q_T trapdoor queries (we assume q_T and q_{H_2} are positive). We construct an algorithm $\mathcal{B}$ that solves the BDH problem with probability at least $\epsilon' = \epsilon/(e q_T q_{H_2})$, where e is the base of the natural logarithm. Algorithm $\mathcal{B}$'s running time is approximately the same as $\mathcal{A}$'s. Hence, if the BDH assumption holds in G_1 then ϵ' is a negligible function and consequently ϵ must be a negligible function in the security parameter.

Let g be a generator of G_1. Algorithm $\mathcal{B}$ is given $u_1 = g^\alpha, u_2 = g^\beta, u_3 = g^\gamma \in G_1$. Its goal is to output $v = e(g, g)^{\alpha\beta\gamma} \in G_2$. Algorithm $\mathcal{B}$ simulates the challenger and interacts with forger $\mathcal{A}$ as follows:

KeyGen. Algorithm $\mathcal{B}$ starts by giving $\mathcal{A}$ the public key $A_{pub} = [g, u_1]$.

H_1, H_2-queries. At any time algorithm $\mathcal{A}$ can query the random oracles H_1 or H_2. To respond to H_1 queries algorithm $\mathcal{B}$ maintains a list of tuples $\langle W_j, h_j, a_j, c_j \rangle$ called the H_1-list. The list is initially empty. When $\mathcal{A}$ queries the random oracle H_1 at a point $W_i \in \{0,1\}^*$, algorithm $\mathcal{B}$ responds as follows:

1. If the query W_i already appears on the H_1-list in a tuple $\langle W_i, h_i, a_i, c_i \rangle$ then algorithm $\mathcal{B}$ responds with $H_1(W_i) = h_i \in G_1$.
2. Otherwise, $\mathcal{B}$ generates a random coin $c_i \in \{0,1\}$ so that $\Pr[c_i = 0] = 1/(q_T + 1)$.
3. Algorithm $\mathcal{B}$ picks a random $a_i \in \mathbb{Z}_p$.
   If $c_i = 0$, $\mathcal{B}$ computes $h_i \leftarrow u_2 \cdot g^{a_i} \in G_1$.
   If $c_i = 1$, $\mathcal{B}$ computes $h_i \leftarrow g^{a_i} \in G_1$.
4. Algorithm $\mathcal{B}$ adds the tuple $\langle W_i, h_i, a_i, c_i \rangle$ to the H_1-list and responds to $\mathcal{A}$ by setting $H_1(W_i) = h_i$. Note that either way h_i is uniform in G_1 and is independent of $\mathcal{A}$'s current view as required.

Similarly, at any time $\mathcal{A}$ can issue a query to H_2. Algorithm $\mathcal{B}$ responds to a query for $H_2(t)$ by picking a new random value $V \in \{0,1\}^{\log p}$ for each new t and setting $H_2(t) = V$. In addition, $\mathcal{B}$ keeps track of all H_2 queries by adding the pair (t, V) to an H_2-list. The H_2-list is initially empty.

Trapdoor queries. When $\mathcal{A}$ issues a query for the trapdoor corresponding to the word W_i algorithm $\mathcal{B}$ responds as follows:

1. Algorithm $\mathcal{B}$ runs the above algorithm for responding to H_1-queries to obtain an $h_i \in G_1$ such that $H_1(W_i) = h_i$. Let $\langle W_i, h_i, a_i, c_i \rangle$ be the corresponding tuple on the H_1-list. If $c_i = 0$ then $\mathcal{B}$ reports failure and terminates.
2. Otherwise, we know $c_i = 1$ and hence $h_i = g^{a_i} \in G_1$. Define $T_i = u_1^{a_i}$. Observe that $T_i = H(W_i)^\alpha$ and therefore T_i is the correct trapdoor for the keyword W_i under the public key $A_{pub} = [g, u_1]$. Algorithm $\mathcal{B}$ gives T_i to algorithm $\mathcal{A}$.

Challenge. Eventually algorithm $\mathcal{A}$ produces a pair of keywords W_0 and W_1 that it wishes to be challenged on. Algorithm $\mathcal{B}$ generates the challenge SPKE as follows:

1. Algorithm $\mathcal{B}$ runs the above algorithm for responding to H_1-queries twice to obtain a $h_0, h_1 \in G_1$ such that $H_1(W_0) = h_0$ and $H_1(W_1) = h_1$. For $i = 0, 1$ let $\langle W_i, h_i, a_i, c_i \rangle$ be the corresponding tuples on the H_1-list. If both $c_0 = 1$ and $c_1 = 1$ then $\mathcal{B}$ reports failure and terminates.
2. We know that at least one of c_0, c_1 is equal to 0. Algorithm $\mathcal{B}$ randomly picks a $b \in \{0,1\}$ such that $c_b = 0$ (if only one c_b is equal to 0 then no randomness is needed since there is only one choice).
3. Algorithm $\mathcal{B}$ responds with the challenge SPKE $C = [u_3,\ J]$ for a random $J \in \{0,1\}^{\log p}$.

Note that this challenge implicitly defines $H_2(e(H_1(W_b), u_1^\gamma)) = J$. In other words,

$$J = H_2(e(H_1(W_b), u_1^\gamma)) = H_2(e(u_2 g^{a_b}, g^{\alpha\gamma})) = H_2(e(g,g)^{\alpha\gamma(\beta + a_b)})$$

With this definition, C is a valid SPKE for W_b as required.

More trapdoor queries. $\mathcal{A}$ can continue to issue trapdoor queries for keywords W_i where the only restriction is that $W_i \neq W_0, W_1$. Algorithm $\mathcal{B}$ responds to these queries as before.

Output. Eventually, $\mathcal{A}$ outputs its guess $b' \in \{0,1\}$ indicating whether the challenge C is the result of $\mathsf{SPKE}(A_{pub}, W_0)$ or $\mathsf{SPKE}(A_{pub}, W_1)$. At this point, algorithm $\mathcal{B}$ picks a random pair (t, V) from the H_2-list and outputs $t/e(u_1, u_3)^{a_b}$ as its guess for $e(g,g)^{\alpha\beta\gamma}$, where a_b is the value used in the Challenge step. The reason this works is that, as we will show, $\mathcal{A}$ must have issued a query for either $H_2(e(H_1(W_0), u_1^\gamma))$ or $H_2(e(H_1(W_1), u_1^\gamma))$. Therefore, with probability 1/2 the H_2-list contains a pair whose left hand side is $t = e(H_1(W_b), u_1^\gamma) = e(g,g)^{\alpha\gamma(\beta+a_b)}$. If $\mathcal{B}$ picks this pair (t, V) from the H_2-list then $t/e(u_1, u_3)^{a_b} = e(g,g)^{\alpha\beta\gamma}$ as required.

This completes the description of algorithm $\mathcal{B}$. It remains to show that $\mathcal{B}$ correctly outputs $e(g,g)^{\alpha\beta\gamma}$ with probability at least ϵ'. To do so, we first analyze the probability that $\mathcal{B}$ does not abort during the simulation. We define two events:

$\mathcal{E}_1$**:** $\mathcal{B}$ does not abort as a result of any of $\mathcal{A}$'s trapdoor queries.
$\mathcal{E}_2$**:** $\mathcal{B}$ does not abort during the challenge phase.

We first argue as in [5] that both events $\mathcal{E}_1$ and $\mathcal{E}_2$ occur with sufficiently high probability.

Claim 1: The probability that algorithm $\mathcal{B}$ does not abort as a result of $\mathcal{A}$'s trapdoor queries is at least $1/e$. Hence, $\Pr[\mathcal{E}_1] \geq 1/e$.

Proof. Without loss of generality we assume that $\mathcal{A}$ does not ask for the trapdoor of the same keyword twice. The probability that a trapdoor query causes $\mathcal{B}$ to abort is $1/(q_T+1)$. To see this, let W_i be $\mathcal{A}$'s i'th trapdoor query and let $\langle W_i, h_i, a_i, c_i \rangle$ be the corresponding tuple on the H_1-list. Prior to issuing the query, the bit c_i is independent of $\mathcal{A}$'s view — the only value that could be given to $\mathcal{A}$ that depends on c_i is $H(W_i)$, but the distribution on $H(W_i)$ is the same whether $c_i = 0$ or $c_i = 1$. Therefore, the probability that this query causes $\mathcal{B}$ to abort is at most $1/(q_T+1)$. Since $\mathcal{A}$ makes at most q_T trapdoor queries the probability that $\mathcal{B}$ does not abort as a result of all trapdoor queries is at least $(1 - 1/(q_T+1))^{q_T} \geq 1/e$. □

Claim 2: The probability that algorithm $\mathcal{B}$ does not abort during the challenge phase is at least $1/q_T$. Hence, $\Pr[\mathcal{E}_2] \geq 1/q_T$.

Proof. Algorithm $\mathcal{B}$ will abort during the challenge phase if $\mathcal{A}$ is able to produce W_0, W_1 with the following property: $c_0 = c_1 = 1$ where for $i = 0, 1$ the tuple $\langle W_i, h_i, a_i, c_i \rangle$ is the tuple on the H_1-list corresponding to W_i. Since $\mathcal{A}$ has not queried for the trapdoor for W_0, W_1 we have that both c_0, c_1 are independent of $\mathcal{A}$'s current view. Therefore, since $\Pr[c_i = 0] = 1/(q_T+1)$ for $i = 0, 1$, and the two values are independent of one another, we have that $\Pr[c_0 = c_1 = 1] = (1 - 1/(q_T+1))^2 \leq 1 - 1/q_T$. Hence, the probability that $\mathcal{B}$ does not abort is at least $1/q_T$. □

Observe that since $\mathcal{A}$ can never issue a trapdoor query for the challenge keywords W_0, W_1 the two events $\mathcal{E}_1$ and $\mathcal{E}_2$ are independent. Therefore, $\Pr[\mathcal{E}_1 \wedge \mathcal{E}_2] \geq 1/(eq_T)$.

To complete the proof of Theorem 3.1 it remains to show that $\mathcal{B}$ outputs the solution to the given BDH instance with probability at least ϵ/q_{H_2}. To do we show that during the simulation $\mathcal{A}$ issues a query for $H_2(e(H_1(W_b), u_1^\gamma))$ with probability at least ϵ.

Claim 3: Suppose that in a real attack game $\mathcal{A}$ is given the public key $[g, u_1]$ and $\mathcal{A}$ asks to be challenged on words W_0 and W_1. In response, $\mathcal{A}$ is given a challenge $C = [g^r, J]$. Then, in the real attack game $\mathcal{A}$ issues a query for either $H_2(e(H_1(W_0), u_1^r))$ or $H_2(e(H_1(W_1), u_1^r))$ with probability at least 2ϵ.

Proof. Let $\mathcal{E}_3$ be the event that in the real attack $\mathcal{A}$ does not issue a query for either one of $H_2(e(H_1(W_0), u_1^r))$ and $H_2(e(H_1(W_1), u_1^r))$. Then, when $\mathcal{E}_3$ occurs we know that the bit $b \in \{0,1\}$

indicating whether C is an SPKE of W_0 or W_1 is independent of $\mathcal{A}$'s view. Therefore, $\mathcal{A}$'s output b' will satisfy $b = b'$ with probability at most $\frac{1}{2}$. By definition of $\mathcal{A}$, we know that in the real attack $|\Pr[b = b'] - 1/2| \geq \epsilon$. We show that these two facts imply that $\Pr[\neg\mathcal{E}_3] \geq 2\epsilon$. To do so, we first derive simple upper and lower bounds on $\Pr[b = b']$:

$$\begin{aligned}
\Pr[b = b'] &= \Pr[b = b'|\mathcal{E}_3]\Pr[\mathcal{E}_3] + \Pr[b = b'|\neg\mathcal{E}_3]\Pr[\neg\mathcal{E}_3] \\
&\leq \Pr[b = b'|\mathcal{E}_3]\Pr[\mathcal{E}_3] + \Pr[\neg\mathcal{E}_3] = \frac{1}{2}\Pr[\mathcal{E}_3] + \Pr[\neg\mathcal{E}_3] = \frac{1}{2} + \frac{1}{2}\Pr[\neg\mathcal{E}_3], \\
\Pr[b = b'] &\geq \Pr[b = b'|\mathcal{E}_3]\Pr[\mathcal{E}_3] = \frac{1}{2} - \frac{1}{2}\Pr[\neg\mathcal{E}_3].
\end{aligned}$$

It follows that $\epsilon \leq |\Pr[b = b'] - 1/2| \leq \frac{1}{2}\Pr[\neg\mathcal{E}_3]$. Therefore, in the real attack, $\Pr[\neg\mathcal{E}_3] \geq 2\epsilon$ as required. $\square$

Now, assuming $\mathcal{B}$ does not abort, we know that $\mathcal{B}$ simulates a real attack game perfectly up to the moment when $\mathcal{A}$ issues a query for either $H_2(e(H_1(W_0), u_1^\gamma))$ or $H_2(e(H_1(W_1), u_1^\gamma))$. Therefore, by Claim 3, by the end of the simulation $\mathcal{A}$ will have issued a query for either $H_2(e(H_1(W_0), u_1^\gamma))$ or $H_2(e(H_1(W_1), u_1^\gamma))$ with probability at least 2ϵ. It follows that $\mathcal{A}$ issues a query for $H_2(e(H_1(W_b), u_1^\gamma))$ with probability at least ϵ. Consequently, the value $e(H_1(W_b), u_1^\gamma) = e(g^{\beta+a_b}, g)^{\alpha\gamma}$ will appear on the left hand side of some pair in the H_2-list. Algorithm $\mathcal{B}$ will choose the correct pair with probability at least $1/q_{H_2}$ and therefore, assuming $\mathcal{B}$ does not abort during the simulation, it will produce the correct answer with probability at least ϵ/q_{H_2}. Since $\mathcal{B}$ does not abort with probability at least $1/(eq_T)$ we see that $\mathcal{B}$'s success probability overall is at least $\epsilon/(eq_T q_{H_2})$ as required. $\square$

3.2 Construction using Jacobi symbols

Our second construction for SPKE is based on Jacobi symbols. This construction is derived from an IBE system proposed by Cocks [2]. The security of Cocks' IBE system is based on the difficulty of distinguishing quadratic residues from non-residues modulo $N = pq$ where $p = q = 3(\text{mod}4)$. Surprisingly, to obtain a secure SPKE based on this IBE system we need a much stronger complexity assumption than quadratic residousity. This is more evidence that constructing an SPKE is harder than building an IBE system.

We first describe the SPKE. We will need a hash function $H : \{0,1\}^* \to QR(N)$ where $QR(N) \subset \mathbb{Z}_N^*$ is the set of quadratic residues in $\mathbb{Z}_N^*$.

- KeyGen: pick random s-bit primes p, q such that $p = q = 3(\bmod 4)$ and set $N = pq$. Here s is the security parameter. Also, let $k > 0$ be an integer. The Test algorithm will have an error probability of $1/2^k$. Output $A_{pub} = [N, k]$; $A_{priv} = p$.
- $\mathsf{SPKE}(A_{pub}, W)$: Pick k random values $r_1, \ldots, r_k \in \mathbb{Z}_N^*$ such that the Jacobi symbols $\left(\frac{r_i}{N}\right) = 1$ for all $i = 1, \ldots, k$. Output $\mathsf{SPKE}(A_{pub}, W) = \left[r_1 + \frac{H(W)}{r_1},\ \ldots,\ r_k + \frac{H(W)}{r_k}\right]$.
- $\mathsf{Trapdoor}(A_{priv}, W)$: Output $T_W = H(W)^{1/2} \bmod N$
- $\mathsf{Test}(A_{pub}, S, T_W)$: Let $S = [A_1, ..., A_k]$. Test if the Jacobi-symbols $\left(\frac{A_i + 2T_W}{N}\right) = 1$ for all $i = 1, \ldots, k$. If so, output 'yes', if not, output 'no'.

Observe that $A_i + 2T_W = r_i + H(W)/r_i + 2H(W)^{1/2} = [r_i + H(W)^{1/2}]^2/r_i \bmod N$ and therefore $\left(\frac{A_i+2T_W}{N}\right) = \left(\frac{r_i}{N}\right) = 1$. Hence, the test algorithm will output 'yes' given $\mathsf{SPKE}(A_{pub}, W)$ and T_W.

However, there is a small probability (approximately $1/2^k$) that the test algorithm will output 'yes' even if S is not $\mathsf{SPKE}(A_{pub}, W)$. Note that the size of the resulting SPKE is substantially bigger than the size of the SPKE of the previous section.

We prove that this system is a non-interactive searchable encryption scheme semantically secure under a chosen keyword attack. To prove security we need a strange looking complexity assumption we call the Quadratic Indistinguishability Problem (QIP).

Quadratic Indistinguishability Problem (QIP). Let $N = pq$ be an s-bit RSA composite with $p = q = 3 \bmod 4$. Here s is the security parameter. Let $a \in \mathbb{Z}_N^*$ be a quadratic residue. Denote by $S_a \subset \mathbb{Z}_N^*$ the set $S_a = \{r + (a/r) \mid r \in \mathbb{Z}_N^* \text{ and } \left(\frac{r}{N}\right) = 1\}$. The QIP problem is to distinguish the uniform distribution on S_a from the uniform distribution on $\mathbb{Z}_N^*$. Here r is a random element in $\mathbb{Z}_N^*$ such that $\left(\frac{r}{N}\right) = 1$. Algorithm $\mathcal{A}$ has advantage ϵ in solving QIP if for all quadratic residues a in $\mathbb{Z}_N^*$ we have:

$$|\Pr[\mathcal{A}(g, a, N) = \text{'yes'}] - \Pr[\mathcal{A}(h, a, N) = \text{'yes'}]| > \epsilon$$

where g is chosen uniformly from S_a and h is chosen uniformly from $\mathbb{Z}_N^*$. We say that QIP is intractable if all polynomial time algorithms (in s) have a negligible advantage in solving QIP.

Theorem 3.2. *The non-interactive searchable encryption scheme above is semantically secure against a chosen keyword attack in the random oracle model assuming QIP is intractable.*

Proof. The proof is essentially the same as the proof of Theorem 3.1 and is omitted. □

3.3 Construction using any trapdoor permutation

Our third SPKE construction is based on general trapdoor permutations. We assume that the total space of keywords $\Sigma \subset \{0,1\}^*$ is of polynomial size (as a function of the security parameter). We will also need a family of semantically-secure encryptions where given a ciphertext it is computationally hard to say which public-key this ciphertext is associated with. We call such schemes **source-indistinguishable** public-key encryption schemes. More precisely, we define source-indistinguishability for an encryption scheme (G, E, D) using the following game between a challenger and an attacker $\mathcal{A}$ (here G is the key generation algorithm, and E/D are encryption/decryption algorithms). The security parameter s is given to both players.
Source Indistinguishability security game:

1. The challenger runs algorithm $G(s)$ two times to generate two public/private key pairs $(PK_0, Priv_0)$ and $(PK_1, Priv_1)$.
2. The challenger picks a random $M \in \{0,1\}^s$ and a random $b \in \{0,1\}$ and computes an encryption $C = PK_b(M)$. The challenger gives (M, C) to the attacker.
3. The attacker outputs b' and wins the game if $b = b'$.

In other words, the attacker wins if he correctly guesses whether he was given the encryption of M under PK_0 or under PK_1. We define $\mathcal{A}$'s advantage in winning the game as:

$$\text{AdvSI}_{\mathcal{A}}(s) = |\Pr[b = b'] - \frac{1}{2}|$$

Definition 3.3. We say that a public-key encryption scheme is source-indistinguishable if for any polynomial time attacker $\mathcal{A}$ we have that $\text{AdvSI}_{\mathcal{A}}(s)$ is a negligible function.

It is easy to check that this property can be attained from any trapdoor permutation based encryption scheme, where to encrypt a bit b we pick a random x, and output $[f(x), GL(x) \oplus b]$ where GL is the Goldreich-Levin hard-core bits [9].

Lemma 3.4. *Given any trapdoor permutation family, we can construct a semantically-secure source-indistinguishable encryption scheme.*

We note that source indistinguishability is an orthogonal property to semantic security. One can build a semantically secure system that is not source indistinguishable (by embedding the public key in every ciphertext). Conversely, one can build a source indistinguishable system that is not semantically secure (by embedding the plaintext in every ciphertext).

A simple SPKE from trapdoor permutations. When the keyword family Σ is of polynomial size (in the security parameter) it is easy to construct searchable encryption from any source-indistinguishable public-key system (G, E, D). We let s be the security parameter for the scheme.

- KeyGen: For each $W \in \Sigma$ run $G(s)$ to generate a new public/private key pair $PK_W/Priv_W$ for the source-indistinguishable encryption scheme. The SPKE public key is $A_{pub} = \{PK_W \mid W \in \Sigma\}$. The private key is $A_{priv} = \{Priv_W \mid W \in \Sigma\}$.
- SPKE(A_{pub}, W): Pick a random $M \in \{0,1\}^s$ and output SPKE$(A_{pub}, W) = (M, E[PK_W, M])$, i.e. encrypt M using the public key PK_W.
- Trapdoor(A_{priv}, W): The trapdoor for word W is simply $T_W = Priv_W$.
- Test(A_{pub}, S, T_W): Test if the decryption $D[T_W, S] = 0^s$. Output 'yes' if so and 'no' otherwise.

Note that the dictionary must be of polynomial size (in s) so that the public and private keys are of polynomial size (in s).

This construction gives a semantically secure public-key searchable encryption as stated in the following simple theorem. Semantically secure public-key searchable encryption is defined as in Definition 2.2 except that the adversary is not allowed to make chosen keyword queries.

Theorem 3.5. *The* SPKE *scheme above is semantically secure assuming the underlying public key encryption scheme (G, E, D) is source-indistinguishable.*

Proof sketch: Let $\Sigma = \{W_1, \ldots, W_k\}$ be the keyword dictionary. Suppose we have an SPKE attacker $\mathcal{A}$ for which $\mathrm{Adv}_{\mathcal{A}}(s) > \epsilon(s)$. We build an attacker $\mathcal{B}$ that breaks the source indistinguishability of (G, E, D) where $\mathrm{AdvSI}_{\mathcal{B}}(s) > \epsilon(s)/k^2$.

The reduction is immediate: $\mathcal{B}$ is given two public keys PK_0, PK_1 and a pair (M, C) where M is random in $\{0,1\}^s$ and $C = PK_b(M)$ for $b \in \{0,1\}$. Algorithm $\mathcal{B}$ generates $k-2$ additional public/private keys using $G(s)$. It creates A_{pub} as a list of all k public keys with PK_0, PK_1 embedded in a random location in the list. Let W_i, W_j be the words associated with the public keys PK_0, PK_1. $\mathcal{B}$ sends A_{pub} to $\mathcal{A}$ who then responds with two words $W_k, W_\ell \in \Sigma$ on which $\mathcal{A}$ wishes to be challenged. If $\{i, j\} \neq \{k, \ell\}$ algorithm $\mathcal{B}$ reports failure and aborts. Otherwise, $\mathcal{B}$ sends the challenge (M, C) to $\mathcal{A}$ who then responds with a $b' \in \{0,1\}$. Algorithm $\mathcal{B}$ outputs b' as its response to the source indistinguishability challenge. We have that $b = b'$ if algorithm $\mathcal{B}$ did not abort and $\mathcal{A}$'s response was correct. This happens with probability at least $\frac{1}{2} + \epsilon/k^2$. Hence, $\mathrm{AdvSI}_{\mathcal{B}}(s) > \epsilon(s)/k^2$ as required. $\square$

Reducing the public key size. The drawback of the above scheme is that the public key length grows linearly with the total dictionary size. If we have an upper-bound on the total number of keyword trapdoors that the user will release to the email gateway (though we do not need to know these keywords a-priori) we can do much better using cover-free families [7]. Typically a user will only allow a third party to search for a limited number of keywords so that assuming an upper bound on the number of released trapdoors is within reason. We begin by recalling the definition of cover-free families.

Definition 3.6. Cover-free families. Let d, t, k be positive integers, let G be a ground set of size d, and let $F = \{S_1, \ldots, S_k\}$ be a family of subsets of G. We say that subset S_j does not *cover* S_i if it holds that $S_i \not\subseteq S_j$. We say that family F is *t-cover free* over G if each subset in F is not covered by the union of t subsets in F. Moreover, we say that a family of subsets is *q-uniform* if all subsets in the family have size q.

We will use the following fact from [6].

Lemma 3.7. [6] There exists a deterministic algorithm that, for any fixed t, k, constructs a q-uniform t-cover free family F over a ground set of size d, for $q = \lceil d/4t \rceil$ and $d \leq 16t^2(1 + \log(k/2)/\log 3)$.

The SPKE. Given the previous SPKE construction as a starting point, we can significantly reduce the size of public file A_{pub} by allowing user to re-use individual public keys for different keywords. We associate to each keyword a subset of public keys chosen from a cover free family. Let k be the size of the dictionary $\Sigma = \{W_1, \ldots, W_k\}$ and let t be an upper bound on the number of keyword trapdoors released to the mail gateway by user Alice. Let d, q satisfy the bounds of Lemma 3.7. The $\mathsf{SPKE}(d, t, k, q)$ construction is as follows:

- KeyGen: For $i = 1, \ldots, d$ run algorithm $G(s)$ to generate a new public/private key pair $PK_i/Priv_i$ for the source-indistinguishable encryption scheme. The SPKE public key is $A_{pub} = \{PK_1, \ldots, PK_d\}$. The private key is $A_{priv} = \{Priv_1, \ldots, Priv_d\}$. We will be using a q-uniform t-cover free family of subsets $F = \{S_1, \ldots, S_k\}$ of $\{PK_1, \ldots, PK_d\}$. Hence, each S_i is a subset of public keys.
- $\mathsf{SPKE}(A_{pub}, W_i)$: Let $S_i \in F$ be the subset associated with the word $W_i \in \Sigma$. Let $S_i = \{PK^{(1)}, \ldots, PK^{(q)}\}$. Pick random messages $M_1, \ldots, M_q \in \{0,1\}^s$ and let $M = M_1 \oplus \cdots \oplus M_q$. Output the tuple:

$$\mathsf{SPKE}(A_{pub}, W_i) = \Big(M,\ E[PK^{(1)}, M_1],\ \ldots,\ E[PK^{(q)}, M_q]\Big)$$

- $\mathsf{Trapdoor}(A_{priv}, W_i)$: Let $S_i \in F$ be the subset associated with word $W_i \in \Sigma$. The trapdoor for word W_i is simply the set of private keys that correspond to the public keys in the set S_i.
- $\mathsf{Test}(A_{pub}, R, T_W)$: Let $T_W = \{Priv^{(1)}, \ldots, Priv^{(q)}\}$ and let $R = (M,\ C_1,\ \ldots,\ C_q)$ be an SPKE. For $i = 1, \ldots, q$ decrypt each C_i using private key $Priv^{(i)}$ to obtain M_i. Output 'yes' if $M = M_1 \oplus \cdots \oplus M_q$, and output 'no' otherwise.

The size of the public key file A_{pub} is much smaller now: logarithmic in the size of the dictionary. The downside is that Alice can only release t keywords to the email gateway. Once t trapdoors are released privacy is no longer guaranteed. Also, notice that the size of the SPKE is larger now (logarithmic in the dictionary size and linear in t). The following corollary of Theorem 3.5 shows that the resulting SPKE is secure.

Corollary 3.8. *Let d, t, k, q satisfy the bounds of Lemma 3.7. The* $\mathsf{SPKE}(d, t, k, q)$ *scheme above is semantically secure under a chosen keyword attack assuming the underlying public key encryption scheme* (G, E, D) *is source-indistinguishable and semantically secure, and that the adversary makes no more than* t *trapdoors queries.*

Proof sketch: Let $\Sigma = \{W_1, \dots, W_k\}$ be the keyword dictionary. Suppose we have an SPKE attacker $\mathcal{A}$ for which $\mathrm{Adv}_{\mathcal{A}}(s) > \epsilon(s)$. We build an attacker $\mathcal{B}$ that breaks the source indistinguishability of (G, E, D).

Algorithm $\mathcal{B}$ is given two public keys PK_0, PK_1 and a pair (M, C) where M is random in $\{0,1\}^s$ and $C = PK_b(M)$ for $b \in \{0,1\}$. Its goal is to output a guess for b which it does by interacting with $\mathcal{A}$. Algorithm $\mathcal{B}$ generates $d-2$ additional public/private keys using $G(s)$. It creates A_{pub} as a list of all d public keys with PK_0, PK_1 embedded in a random location in the list. Let W_i, W_j be the words associated with the public keys PK_0, PK_1.

$\mathcal{B}$ sends A_{pub} to $\mathcal{A}$. Algorithm $\mathcal{A}$ issues up to t trapdoor queries. $\mathcal{B}$ responds to a trapdoor query for $W \in \Sigma$ as follows: let $S \in F$ be the subset corresponding to the word W. If $PK_0 \in S$ or $PK_1 \in S$ algorithm $\mathcal{B}$ reports failure and aborts. Otherwise, $\mathcal{B}$ gives $\mathcal{A}$ the set of private keys $\{Priv_i \,|\, i \in S\}$.

At some point, Algorithm $\mathcal{A}$ outputs two words $W_0', W_1' \in \Sigma$ on which it wishes to be challenged. Let $S_0', S_1' \in F$ be the subsets corresponding to W_0', W_1' respectively. Let $\mathcal{E}$ be the event that $PK_0 \in S_0'$ and $PK_1 \in S_1'$. If event $\mathcal{E}$ did not happen then $\mathcal{B}$ reports failure and aborts.

We now know that $PK_0 \in S_0'$ and $PK_1 \in S_1'$. For $j = 0, 1$ let $S_j' = \{PK_j^{(1)}, \dots, PK_j^{(q)}\}$. We arrange things so that $PK_0 = PK_0^{(c)}$ and $PK_1 = PK_1^{(c)}$ for some random $1 \le c \le q$. Next, $\mathcal{B}$ picks random $M_1, \dots, M_{c-1}, M_{c+1}, \dots, M_q \in \{0,1\}^s$ and sets $M_c = M$. Let $M' = M_1 \oplus \cdots \oplus M_q$. Algorithm $\mathcal{B}$ defines the following hybrid tuple:

$$R = \Big(M',\ \ E[PK_0^{(1)}, M_1],\ \dots,\ E[PK_0^{(c-1)}, M_{c-1}],\ \ C,\ \ E[PK_1^{(c+1)}, M_{c+1}],\ \dots,\ E[PK_1^{(q)}, M_q]\Big)$$

It gives R as the challenge SPKE to algorithm $\mathcal{A}$. Algorithm $\mathcal{A}$ eventually responds with some $b' \in \{0,1\}$ indicating whether R is $\mathsf{SPKE}(A_{pub}, W_0')$ or $\mathsf{SPKE}(A_{pub}, W_1')$. Algorithm $\mathcal{B}$ outputs b' as its guess for b. One can show using a standard hybrid argument that if $\mathcal{B}$ does not abort then $|\Pr[b = b'] - \frac{1}{2}| > \epsilon/q^2$. The probability that $\mathcal{B}$ does not abort at a result of a trapdoor query is at least $1 - (tq/d)$. The probability that $\mathcal{B}$ does not abort as a result of the choice of words W_0', W_1' is at least $(q/d)^2$. Hence, $\mathcal{B}$ does not abort with probability at least $1/poly(t, q, d)$. Repeatedly running $\mathcal{B}$ until it does not abort shows that we can get advantage ϵ/q^2 in breaking the source indistinguishability of (G, E, D) in expected polynomial time in the running time of $\mathcal{A}$. $\square$

4 Conclusions

We defined the concept of a searchable public key encryption (SPKE) and gave several constructions. Constructing an SPKE is related to Identity Based Encryption (IBE), though SPKE seems to be harder to construct. We showed that SPKE implies Identity Based Encryption, but the converse is currently an open problem. Our constructions for SPKE are based on recent IBE constructions. We are able to prove security by exploiting extra properties of these schemes. Interestingly, the security of some of our SPKE's require stronger complexity assumptions than their corresponding IBE's. We hope that searchable public key encryption will be used for processing encrypted email and other public-key encrypted data.

Acknowledgments

We thank Glenn Durfee for suggesting the use of H_2 in the construction of Section 3.1.

References

[1] D. Boneh and M. Franklin, *Identity-based Encryption from the Weil Pairing,* SIAM J. of Computing, Vol. 32, No. 3, pp. 586-615, 2003, Extended abstract in Crypto 2001.

[2] C. Cocks, *An identity based encryption scheme based on quadratic residues*, Eighth IMA International Conference on Cryptography and Coding, Dec. 2001, Royal Agricultural College, Cirencester, UK.

[3] C. Cachin, S. Micali, M. Stadler *Computationally Private Information Retrieval with Polylogarithmic Communication* Eurcrypt 1999.

[4] B. Chor, O. Goldreich, E. kushilevitz and M. Sudan, *Private Information Retrieval,* in FOCS 95 (also Journal of ACM).

[5] J. Coron, "On the exact security of Full-Domain-Hash", in *Advances in Cryptology – Crypto 2000*, Lecture Notes in Computer Science, Vol. 1880, Springer-Verlag, pp. 229–235, 2000.

[6] D. Z. Du and F. K. Hwang, *Combinatorial Group Testing and its Applications,* World Scientific, Singapore, 1993.

[7] P. Erdos, P. Frankl and Z. Furedi, *Families of finite sets in which no set is covered by the union of r others,* in Israeli Journal of Mathematics, 51: 79–89, 1985.

[8] O. Goldreich, S. Micali, A. Wigderson, "How to play any mental game", Proc. 19th Annual ACM Symposium on Theory of Computing, pp. 218–229, 1987.

[9] S. Goldwasser and S. Micali, *Probabilistic Encryption*, in Journal of Computer and System Sciences. vol. 28 (1984), n. 2, pp. 270–299.

[10] O. Goldreich and R. Ostrovsky. Software protection and simulation by oblivious RAMs. *JACM*, 1996.

[11] A. Joux, "The Weil and Tate Pairings as Building Blocks for Public Key Cryptosystems", in *Proc. Fifth Algorithmic Number Theory Symposium*, Lecture Notes in Computer Science, Springer-Verlag, 2002.

[12] E. Kushilevitz and R. Ostrovsky, *Replication is not needed: Single Database, Computationally-Private Information Retrieval,* in FOCS 97.

[13] P. Maniatis, M. Roussopoulos, E. Swierk, K. Lai, G. Appenzeller, X. Zhao, and M. Baker, *The Mobile People Architecture.* ACM Mobile Computing and Communications Review (MC2R), Volume 3, Number 3, July 1999.

[14] R. Ostrovsky. Software protection and simulation on oblivious RAMs. M.I.T. Ph. D. Thesis in Computer Science, June 1992. Preliminary version in *Proc. 22nd Annual ACM Symp. Theory Comp.*, 1990.

[15] D. Song, D. Wagner, and A. Perrig, *Practical Techniques for Searches on Encrypted Data*, in Proc. of the 2000 IEEE symposium on Security and Privacy (S&P 2000).

[16] A. Shamir, *Identity-based Cryptosystems and Signature Schemes,* in CRYPTO 84.

[17] B. Waters, D. Balfanz, G. Durfee, D. Smetters, “Building an encrypted and searchabe audit log”, manuscript.

[18] A. Yao, “How to generate and exchange secrets”, Proc. 18th Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 162–167, 1986.