

Secure Conjunctive Keyword Searches For Unstructured Text

Florian Kerschbaum
SAP Research
Karlsruhe, Germany
Email: florian.kerschbaum@sap.com

Abstract—There are a number of searchable encryption schemes that allow secure conjunctive keyword searches over encrypted data, but all of them assume that the position of the keywords is known. This is a pity, since in unstructured text, e.g. the body of an e-mail, this position is unknown and one has to construct $O(m^n)$ search tokens for n keywords in a text of length m . In this paper we present a searchable encryption scheme that allows conjunctive keyword searches without specifying the position requiring only one search token with constant ciphertext length. We prove the security of our scheme using the external Diffie-Hellman assumption in the random oracle model.

I. INTRODUCTION

Searchable encryption allows a secret key holder to issue search tokens for keywords that enable other parties to perform equality comparisons with ciphertexts. Assume a set of text documents with overall w keywords where each document is associated with m of these (possibly identical) keywords. A querier is asking the secret key holder to issue him a search token t for n distinct keywords. The ciphertext of a document matches the search token if the set of n keywords of the search token is a subset of the set of m keywords of the document.

Searchable encryption has many practical applications. A typical scenario is outsourced storage where the secret key holder uses a data storage provider to host his encrypted documents, but wants to enable the storage provider to search his documents without decrypting them. Another example is outsourced data analytics and auditing [14]. In this paper we stick to the scenario of an outsourced storage service. The client may store a number of encrypted documents, e.g. e-mails, holding only his secret key. Furthermore, he may want to search the free text of his documents at the provider without decrypting them.

Since its first proposal [16] many searchable encryption schemes have been proposed. Some are secret-key [3], [4], [7], [8], [9], [11], [12] and some include the possibility to encrypt documents using only a public-key [1], [2], [5], [6], [10], [13], [15], [17]. Many schemes also include the possibility to search for a conjunction (“and” combination) of keywords [3], [6], [7], [12], [13], [15], [17]. Nevertheless all of the conjunctive keyword search proposals face a severe obstacle when applied to our scenario. Following the original setup of [12] each keyword is associated with an index – a position in the ciphertext. The search token matches only, if all keywords at the specified positions match, i.e. it is not sufficient for a

match that the search token keywords are a subset. Instead, both, keyword and position, must match.

This setup works well for structured data, e.g. header fields in e-mail. It is easily possible to check whether an e-mail is from Alice to Bob, since both fields “From:” and “To:” appear at a fixed position in the e-mail. It is much more difficult to search in unstructured text, such as the body of the e-mail itself. Assume searching for “Alice” and “Bob” in the body of the e-mail, these words could appear at any position.

There are two methods of adapting this setup to our scenario. First, one can create a sorted index of all keywords in the documents. This index would be of size w and each keyword w_i would have a fixed position i . When a document does not contain a keyword w_i a dummy keyword would be included at position i . Therefore a ciphertext would always be of size w , the number of all keywords. Nevertheless, given a conjunction of keywords there is only one search token necessary to check whether a document contains them.

The second method of adapting the setup to our scenario is word-by-word encryption. A document then is associated with a vector of m keywords where second or higher occurrences are removed. A keyword appears at its natural position, e.g. the position of a word in the plaintext document.

The size of the ciphertext remains on the order of m . Nevertheless, searching has become difficult. The querier needs to guess the positions of the keywords and he has m options. He must request search tokens for all possible positions of his n keywords in the vector of length m . This results in $O(m^n)$ search tokens – one for each combination of indices which results in a significant computation and communication overhead.

In this paper we propose a secret-key encryption scheme that allows conjunctive keyword searches without the need to specify the position of the keywords. The querier needs to only specify the set of keywords and the secret key holder can create a search token for their conjunction that matches at any position.

This perfectly suits our scenario of an outsourced storage provider. We also use word-by-word encryption, but our search tokens are more efficient. Given a search token, a querier may access the encrypted storage and compare every combination of ciphertext (of one document). The search token will indicate a match, if all keywords match, e.g. they are “Alice” and “Bob”. This enables the envisioned free text search with only

one search token.

As a result of a match between search token and ciphertext, the querier will learn the positions of the keywords. We emphasize that this information is either publicly available in case of an index, or it is also revealed by the specific search token of the match in case of word-by-word encryption. Even worse, all conjunctive keyword-searchable encryption schemes except [17] reveal the positions of the keywords to the querier.

A. Our Contribution

We propose an encryption scheme that allows conjunctive keyword searches without the need to specify the position of the keywords. The basic idea is simple: Use a homomorphic encryption scheme and make it searchable. Conjunctions are implicit. Compute the homomorphic operation on the “combination” of plaintexts and issue a search token for it. The search operation then first computes the “combination” of ciphertexts and compares it to the search token. This immediately enables a position-independent, index-free conjunctive keyword search.

Nevertheless the construction has been quite difficult and we could not achieve some desirable properties, such as proofs in the standard model. On the one hand existing searchable encryption schemes escape this construction, simply because they are not homomorphic. On the other hand there is a subtle problem. The straight-forward way of making most homomorphic encryption schemes “searchable” results in homomorphic search tokens. Given a search token for keywords a and b , it is easy to compute a search token for a AND b . This leads to a trivial attack on the IND-CPA security game for searchable encryption, namely to encrypt a and encrypt b , get search tokens for each and when challenged for two plaintexts use a AND b as one of them. Now, we provide a proof of our encryption scheme in the IND-CPA game for searchable encryption where we augment our secret-key encryption scheme with an encryption oracle.

Despite its novel security and functionality properties our scheme is based on a number of existing encryption schemes. Essentially, we use the old, but popular Pohlig-Hellman encryption to deterministically encrypt the keywords and then use a “searchable” variant of El-Gamal encryption to randomize ciphertexts and search tokens. The randomization between search tokens and ciphertexts can be matched using bilinear maps.

Our construction has a limitation due to its security definition. In an additively homomorphic encryption scheme an adversary can compute the inverse of the plaintext as a ciphertext. This implies that the adversary given an arbitrary search token t and matching ciphertext c can compare two other ciphertexts for equality of plaintexts. Assume c' and c'' are such ciphertexts. The adversary computes the inverse of the plaintext of c'' , i.e. c''^{-1} , and homomorphically combines c , c' and c''^{-1} . If the combined ciphertext still matches the search token t , then c' and c'' have the same plaintext. We circumvent this problem by allowing only distinct plaintexts in our security definition.

The remainder of the paper is structured as follows. First we review the basics, i.e. algorithm interfaces, security assumptions and definitions in Section II. We then present the encryption scheme in Section III. Related work is presented in Section IV. Section V concludes the paper.

II. BASICS

A. Definition

Let w be a keyword. We consider the scenario where one wants to match a conjunction of distinct keywords w_i to a set of ciphertexts. Let k be the security parameter. A searchable encryption scheme then consists of the following algorithms:

- 1) $KeyGen(k) \rightarrow sk$: Takes a security parameter k and outputs a secret key sk .
- 2) $Encrypt(w, sk) \rightarrow c$: Takes a keyword w and a secret key sk and outputs a ciphertext c searchable for w .
- 3) $Trapdoor(w_1, \dots, w_n, sk) \rightarrow t$: Takes a multi-set of keywords w_1 through w_n and a secret key sk and outputs a search token t .
- 4) $Test(c_1, \dots, c_n, t) \rightarrow \top/\perp$: Takes a set of ciphertexts c_1 through c_n and a search token t and outputs a bit \top or \perp .

We can now clarify the difference in construction between our encryption scheme and the one introduced in [12]. In the model of [12] the positions of the keywords are fixed first, then given the positions i_1, \dots, i_n and the keywords w_{i_1}, \dots, w_{i_n} a search token is constructed using $TrapDoor$. In a third step, this search token can be matched at the positions initially fixed using $Test$.

Our encryption scheme improves over this in the following way. The search token is created first using $TrapDoor$, independently of the positions of the keywords. Then a subset of ciphertexts potentially containing the keywords is selected, i.e. the keyword positions are chosen as the second step. Finally, as before, we can perform a match using $Test$ between the choice of ciphertexts and the search token.

B. Consistency

An essential property of a searchable encryption scheme is that search tokens match, if compared to matching keywords. We allow a negligible error probability – in the security parameter k – in case of a mismatch.

Definition 1: A searchable encryption scheme is consistent, if

$$\begin{aligned} \forall w_i = w'_i \quad & Test(\dots, Encrypt(w_i, sk), \dots, \\ & Trapdoor(\dots, w'_i, \dots)) = \top \\ \exists w_i \neq w'_i \quad & Pr[\dots, Test(Encrypt(w_i, sk), \dots, \\ & Trapdoor(\dots, w'_i, \dots)) = \top] < \frac{1}{\text{poly}(k)} \end{aligned}$$

C. Security

1) *Definition:* We follow the IND-CPA game for searchable encryption and define the following game with an augmented encryption oracle for assessing the security of our secret-key searchable encryption scheme. An adversary \mathcal{A} engages in a game with a challenger \mathcal{B} . If the adversary wins, he can break the security of our encryption scheme.

Game IND-CPA-SEARCH:

- 1) *Phase I*: The adversary \mathcal{A} requests from the challenger in arbitrary order
 - encryptions of p distinct keywords w_i ($i = 1, \dots, p$).
 - search tokens for q keyword conjunctions $C_j = \{w_{j_1}, \dots, w_{j_{n_j}}\}$ ($j = 1, \dots, q, 1 \leq n_j \leq n$).
- 2) *Challenge*: The adversary \mathcal{A} outputs two different keywords w_0^* and w_1^* .
Constraints: We require that no requested search token contains any of the challenge keywords.

$$\nexists C_j, w_0^* \in C_j \vee w_1^* \in C_j$$

We also require that the keywords are distinct from the requested ciphertexts

$$\forall i w_i \neq w_0^* \wedge w_i \neq w_1^*$$

The challenger \mathcal{C} flips a coin $b \in \{0, 1\}$ and outputs the encryption of keyword w_b^* .

- 3) *Phase II*: The adversary \mathcal{A} continues to request up to p ciphertexts and q search tokens from the challenger in arbitrary order as long as the constraint still holds.
- 4) *Guess*: The adversary \mathcal{A} outputs a guess b^* of b and is successful if $b^* = b$.

In this game the challenger maintains the encryption oracle and is queried for the encryptions. We stress that the adversary can request to encrypt arbitrary, distinct keywords. Nevertheless, loosely speaking, the adversary may not ask for search tokens that distinguish his challenge plaintexts.

We define the adversary \mathcal{A} 's advantage in this game as $Adv_{\mathcal{A}}^{CPA}(1^k) = |Pr[b^* = b] - \frac{1}{2}|$.

Definition 2: We call a searchable encryption scheme with conjunctive keyword search secure according to game IND-CPA-SEARCH, if the adversary \mathcal{A} 's advantage $Adv_{\mathcal{A}}^{CPA}(1^k) < \frac{1}{poly(k)}$ is a negligible function of the security parameter k .

D. Tools

Our searchable encryption scheme operates on elliptic curves and uses bilinear maps. Let $\mathbb{G}_1, \mathbb{G}_2$ and \mathbb{G}_3 be groups of order p for some large prime p where the bit-size of p is determined by the security parameter k . A bilinear map is a function $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \mapsto \mathbb{G}_3$ with the following properties:

- **Bilinear**: for $g \in \mathbb{G}_1, h \in \mathbb{G}_2$

$$\hat{e}(g^a, h^b) = \hat{e}(g, h)^{ab}$$

- **Non-degenerate**: $\hat{e}(g, h) \neq 1$ is a generator of \mathbb{G}_3
- **Computable**: there exists an efficient algorithm to compute $\hat{e}(g, h)$ for all $g \in \mathbb{G}_1$ and $h \in \mathbb{G}_2$

Modified Weil or Tate pairings on supersingular elliptic curves are examples of such bilinear maps. Let $H : \{0, 1\}^* \mapsto \mathbb{G}_1$ be a cryptographic hash function mapping unto \mathbb{G}_1 .

1) *Assumptions*: Next, we define the assumption we use in order to prove our encryption scheme secure.

Definition 3: We say that the *Decisional Diffie Hellman* (DDH) assumption holds in \mathbb{G} , if given values $g, g^a, g^b, g^c \in \mathbb{G}$ it is not computationally feasible to decide if $c = ab$.

Definition 4: We say that the *External Diffie Hellman* (XDH) assumption holds, if there exists a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \mapsto \mathbb{G}_3$ and the DDH assumption holds in \mathbb{G}_1 .

Definition 5: We say that the *Bilinear Decisional Diffie Hellman* (BDDH) assumption holds, if given values $g, g^a, g^b, g^c \in \mathbb{G}_1, h, h^a, h^b, h^c \in \mathbb{G}_2$ and $\hat{e}(g, h)^d \in \mathbb{G}_3$ it is not computationally feasible to decide if $d = abc$.

Note that the XDH assumption implies the DDH (by definition) and also the BDDH assumption.

Definition 6: We say that in the *Random Oracle* (RO) model the cryptographic hash function H can be modeled as a random source.

III. ALGORITHMS

We now describe the algorithms for our position-independent, conjunctive keyword-searchable encryption scheme.

- *KeyGen*: Let p be a large prime. Recall that $\mathbb{G}_1, \mathbb{G}_2$ and \mathbb{G}_3 are groups of order p . Let g be a random generator of \mathbb{G}_1 and h a random generator of \mathbb{G}_2 . Let y and z be random elements of \mathbb{Z}_p^* . The public parameters are

$$g \quad h$$

The secret key consists of

$$y \quad z$$

- *Encrypt*: We encrypt each keyword as a single ciphertext. To encrypt a keyword w one chooses a random number $r \in \mathbb{Z}_p^*$. One then computes

$$A = g^r \quad B = g^{ry} H(w)^z$$

The ciphertext is $\langle A, B \rangle$.

- *Trapdoor*: We create one search token for a conjunction of keywords. To create a search token for the keywords w_i ($1 \leq i \leq n$) one chooses a random $s \in \mathbb{Z}_p^*$. One then computes

$$S = h^s \quad T = h^{sy} \\ U = \hat{e}(\prod_{i=1}^n H(w_i)^z, h^s)$$

The search token is $\langle S, T, U \rangle$.

- *Test*: Let $c_i = \langle A_i, B_i \rangle$ ($1 \leq i \leq n$) be the set of ciphertexts to be compared against the search token $t = \langle S, T, U \rangle$. For a match one evaluates

$$\frac{\hat{e}(\prod_{i=1}^n B_i, S)}{\hat{e}(\prod_{i=1}^n A_i, T)U} \stackrel{?}{=} 1$$

For consistency, we observe, if there is a match, then

$$\begin{aligned} \frac{\hat{e}(\prod_{i=1}^n B_i, S)}{\hat{e}(\prod_{i=1}^n A_i, T)U} &= \frac{\hat{e}(\prod_{i=1}^n g^{r_i y} H(w_i)^z, h^s)}{\hat{e}(\prod_{i=1}^n g^{r_i}, h^{sy}) \hat{e}(\prod_{i=1}^n H(w_i)^z, h^s)} \\ &= \frac{\hat{e}(\prod_{i=1}^n g^{r_i}, h)^{ys} \hat{e}(H(w_i), h)^{zs}}{\hat{e}(\prod_{i=1}^n g^{r_i}, h)^{ys} \hat{e}(H(w_i), h)^{zs}} \\ &= 1 \end{aligned}$$

If there is no match, then the hashes of the keywords are uniformly distributed in \mathbb{G}_1 .

A. Proof

Theorem 1: Assume the RO model and XDH assumptions holds, then an adversary \mathcal{A} has a negligible advantage in winning game *IND-CPA-SEARCH*.

Lemma 1: Suppose an adversary \mathcal{A} has advantage ϵ in winning game *IND-CPA-SEARCH*, then there exists an algorithm \mathcal{B} that solves the DDH problem in \mathbb{G}_1 with probability at least $\frac{\epsilon}{4e^{(p+qn+\frac{1}{2})}}$ where e is Euler's constant (the base of the natural logarithm).

Proof Outline: We construct an algorithm \mathcal{B} that given an instance g, g^a, g^b, g^c of the DDH problem in \mathbb{G}_1 will construct a challenge ciphertext. The search token will be a valid search token, iff $c = ab$. Loosely speaking, we construct the ciphertext by setting $\log_g H(w) = a$ and $z = b$ in our encryption scheme. In order to do so, we need to control the output of the hash function random oracle. If the adversary \mathcal{A} guesses the coin flip b correctly, we leverage its advantage ϵ and guess $c = ab$.

Proof: We construct the algorithm \mathcal{B} corresponding to the phases of the game *IND-CPA-SEARCH*.

Phase I: \mathcal{B} is given an instance g, g^a, g^b, g^c of the DDH problem in \mathbb{G}_1 . \mathcal{B} chooses a random number $y \in \mathbb{Z}_p$.

Oracle Queries: \mathcal{B} maintains a list $L = \langle w_i, \alpha_i, l_i \rangle$ of random choices α_i for keywords w_i with coin flip l_i . This list is initially empty and simulates the hash function as a random oracle. If the random oracle is queried for a hash of w , \mathcal{B} searches the list L .

- 1) If the flag $l_i = 0$ equals zero, then \mathcal{B} responds with g^{α_i} .
- 2) If the flag $l_i = 1$ equals one, then \mathcal{B} responds with g^a .
- 3) If there is no entry for keyword w on the list, then \mathcal{B} flips a random coin $l \in \{0, 1\}$ so that $\Pr[\text{coin}' = 0] = \delta$ for some δ that will be determined later.
 - a) If $l = 0$ is zero, \mathcal{B} chooses a random number $\alpha \in \mathbb{Z}_p^*$ and adds $\langle w, \alpha, 0 \rangle$ to the list L .
 - b) If $l = 1$ is one, then \mathcal{B} adds $\langle w, \perp, 1 \rangle$ to the list L .
 - c) \mathcal{B} responds accordingly (in both cases).

Let H be a random oracle controlled by \mathcal{B} as described above.

If the adversary \mathcal{A} requests an encryption of keyword w , \mathcal{B} queries the random oracle for keyword w . He then searches the list L for an entry $\langle w, \alpha, l \rangle$ for keyword w . If the coin flip is $l = 1$ is one, \mathcal{B} aborts. We now know that the coin flip $l = 0$ is zero and therefore $H(w) = g^\alpha$. \mathcal{B} chooses a random number $r \in \mathbb{Z}_p^*$ and computes

$$A = g^r \quad B = g^{yr} H(w)^z = g^{yr} (g^b)^\alpha$$

If the adversary \mathcal{A} requests a search token for keywords w_i ($1 \leq i \leq n$), \mathcal{B} queries the random oracle for keywords w_i . He then searches the list L for all entries $\langle w_i, \alpha_i, l_i \rangle$ for keywords w_i . If any coin flip $l_i = 1$ is one, he aborts. We now know that all coin flips $l_i = 0$ are zero and consequently

all $H(w_i) = g^{\alpha_i}$. He chooses a random number $s \in \mathbb{Z}_p^*$. He computes

$$S = h^s \quad T = h^{sy} \\ U = \hat{e}(\prod_{i=1}^n H(w_i)^z, h^s) = \hat{e}(\prod_{i=1}^n (g^{\alpha_i})^\alpha, h^s)$$

Challenge: The adversary \mathcal{A} outputs two keywords w_0^* and w_1^* . \mathcal{B} flips a coin $b \in \{0, 1\}$. He queries the random oracle H for keywords w_b^* and subsequently searches the list L for an entry $\langle w_b^*, \alpha, l \rangle$. If the coin flip $l = 0$ is zero, he aborts. We now know that $l = 1$ and therefore $H(w_b^*) = g^a$. He computes

$$A = g^r \quad B = g^{yr} H(w_b^*)^z = g^{yr} g^c$$

Phase II: \mathcal{B} responds to the requests from the adversary \mathcal{A} as in phase I.

Guess: The adversary outputs its b^* . \mathcal{B} outputs $c = ab$, if $b^* = b$ and $c = r$ otherwise.

Claim: If algorithm \mathcal{B} does not abort during the simulation and the problem instance is a DDH triple, then \mathcal{A} 's view is identical to its view in a real attack. The responses to H queries are as in a real attack, since each is uniformly and independently distributed in \mathbb{G}_1 . If the problem instance is not a DDH triple, the challenge ciphertext is uniformly distributed and contains no information about the keywords. According to the rules, all plaintexts are distinct from the challenge plaintexts and no search token may distinguish the challenge plaintexts.

If $g^c = g^{ab}$, then \mathcal{A} has advantage $\text{Adv}_{\mathcal{A}} \geq \epsilon$ in breaking game *IND-CPA-SEARCH*, since it receives a valid ciphertext. Therefore if \mathcal{B} does not abort, $|\Pr[b = b^*] - \frac{1}{2}| \geq \frac{1}{2}\epsilon$.

To complete the proof of Lemma 1 we need to calculate the probability that algorithm \mathcal{B} aborts during the simulation. Suppose \mathcal{A} makes p encryption requests and q search token requests in each phase. Then the probability that \mathcal{B} does not abort in query phases 1 or 2 is $\delta^{2(p+qn)}$. The probability that it does not abort during the challenge step is $1 - \delta$ which results in an overall probability that \mathcal{B} does not abort is $\delta^{2(p+qn)}(1 - \delta)$. This value is maximized at $\delta_{opt} = 1 - \frac{1}{2(p+qn)+1}$. Using δ_{opt} the probability that \mathcal{B} does not abort is at least $\frac{1}{2e^{(p+qn+1)}}$ where e is Euler's constant. Then \mathcal{B} 's advantage in breaking *DDH* is at least $|\Pr[b^* = b] - \frac{1}{2}| \geq \frac{\epsilon}{4e^{(p+qn+\frac{1}{2})}}$. ■

IV. RELATED WORK

Searchable encryption has a long history and very many different schemes have been proposed. The idea was first presented in [16]. It used symmetric keys with single keyword search.

Since then, a number of refinements have been developed. The first public-key searchable encryption scheme was presented in [5]. It particular caters for the outsourced e-mail scenario where multiple parties may store documents. It only allowed searching for single keywords.

The first searchable encryption scheme for conjunctive keyword searches appeared in [12]. It introduced the notion of positions for keywords which we challenge in this paper. The scheme is also a symmetric key encryption scheme.

The first public-key encryption scheme with conjunctive keyword searches was proposed in [15]. It carried over the position design choice unmodified. A second performance-improved scheme in [13] does so as well. Albeit these schemes are public-key and support conjunctive keyword searches we argue they fail for unstructured text.

A public-key searchable encryption scheme that not only supports conjunctive, but also range and subset queries has been proposed in [6]. Loosely speaking, it allows to compare an encrypted bit string with a search token that besides bits may also contain “don’t care” symbols.

A public-key, searchable encryption scheme with conjunctive keyword searches and enhanced security has been presented in [17]. The improvement of the scheme lies in the ability to hide the keyword positions from the querier. This is particularly useful in the index-based method, since the position reveals the keyword, if the index is known. We stress that this does not overcome the position design choice, since the secret key holder still needs to know the positions when creating the search token.

The first scheme enabling conjunctive keyword searches was also a symmetric scheme [12] as ours. It has later been refined, but kept secret-key, in [7] to only rely on standard security assumptions. More efficient, but still position-dependent, techniques for conjunctive keyword search in the symmetric model have been given in [3].

The earliest searchable encryption techniques have all been symmetric [8], [11], [16]. Later on, their security has been challenged and improved definitions, efficient constructions and according proofs were given in [9].

Recently it has been noted that for sub-linear time search on encrypted data, the encryption must be deterministic [4]. This notion does not translate to conjunctive keyword search, since otherwise the attack described in Section I-A is always feasible, but based on the ciphertexts. Therefore a scheme with our security definition must imply linear time search.

The correspondence between searchable encryption and identity-based encryption has been first noted in [5] and later formalized in [1]. Our scheme essentially follows the construction of [1], but requires a combination of identities.

Combining ciphertexts is the challenge in our construction, since it implies homomorphism. We no longer can rely on the position of the keyword to match one ciphertext to one identity and then cleverly combining the result. Instead, we need to combine the ciphertexts before matching them to a combination of identities. This clearly separates our work from previous results.

V. CONCLUSIONS

We have reviewed the design choice in all searchable encryption schemes that support conjunctive keyword searches to associate each keyword with a position. In the most practical method of word-by-word encryption this results in an exponential number of search tokens when searching in unstructured text. We then propose a searchable encryption scheme that supports conjunctive keyword searches without

the need to specify a position for the keywords. This reduces the required number of search tokens to one. We have proven ciphertext indistinguishability under a chosen plain text attack with encryption oracle adapted to searchable encryption.

REFERENCES

- [1] M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi. Searchable Encryption Revisited: Consistency Properties, Relation to Anonymous IBE, and Extensions. *Proceedings of CRYPTO*, 2005.
- [2] J. Baek, R. Safavi-Naini, W. Susilo. Public Key Encryption with Keyword Search Revisited. *Proceedings of the International Conference on Computational Science and Its Applications*, 2008.
- [3] L. Ballard, S. Kamara, and F. Monrose. Achieving Efficient Conjunctive Keyword Searches over Encrypted Data. *Proceedings of the 7th International Conference on Information and Communications Security*, 2005.
- [4] M. Bellare, A. Boldyreva, and A. O’Neill. Deterministic and efficiently searchable encryption. *Proceedings of CRYPTO*, 2007.
- [5] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano. Public Key Encryption with Keyword Search. *Proceedings of EUROCRYPT*, 2004.
- [6] D. Boneh, and B. Waters. Conjunctive, Subset, and Range Queries on Encrypted Data. *Proceedings of the 4th Theory of Cryptography Conference*, 2007.
- [7] J. Byun, D. Lee, and J. Lim. Efficient Conjunctive Keyword Search on Encrypted Data Storage System. *Proceedings of the 3rd European PKI Workshop*, 2006.
- [8] Y. Chang and M. Mitzenmacher. Privacy preserving keyword searches on remote encrypted data. *Proceedings of the International Conference on Applied Cryptography and Network Security*, 2005.
- [9] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. *Proceedings of the 13th ACM Conference on Computer and Communications Security*, 2006.
- [10] G. Di Crescenzo, and V. Saraswat. Public Key Encryption with Searchable Keywords Based on Jacobi Symbols. *Proceedings of INDOCRYPT*, 2007.
- [11] E. Goh. Secure indexes. *IACR ePrint Technical Report 2003/216*, 2003.
- [12] P. Golle, B. Waters, and J. Staddon. Secure Conjunctive Keyword Search over Encrypted Data. *Proceedings of the 2nd International Conference on Applied Cryptography and Network Security*, 2004.
- [13] Y. Hwang, and P. Lee. Public Key Encryption with Conjunctive Keyword Search and Its Extension to a Multi-user System. *Proceedings of the 1st International Conference on Pairing-Based Cryptography*, 2007.
- [14] F. Kerschbaum, and A. Sorniotti. Searchable Encryption for Outsourced Data Analytics. *Proceedings of the 7th European PKI Workshop*, 2010.
- [15] D. Park, K. Kim, and P. Lee. Public Key Encryption with Conjunctive Field Keyword Search. *Proceedings of the 5th International Workshop on Information Security Applications*, 2004.
- [16] D. Song, D. Wagner, and A. Perrig. Practical Techniques for Searches on Encrypted Data. *Proceedings of the IEEE Symposium on Security and Privacy*, 2000.
- [17] P. Wang, H. Wang, and J. Pieprzyk. Keyword Field-Free Conjunctive Keyword Searches on Encrypted Data and Extension for Dynamic Groups. *Proceedings of the 7th International Conference on Cryptology and Network Security*, 2008.