

# Searching Keywords with Wildcards on Encrypted Data <sup>\*</sup>

Saeed Sedghi<sup>1</sup>, Peter van Liesdonk<sup>2</sup>,  
Svetla Nikova<sup>1,3</sup>, Pieter Hartel<sup>1</sup>, and Willem Jonker<sup>1,4</sup>

<sup>1</sup> Dept. EWI/DIES, University of Twente, Enschede, The Netherlands  
`s.sedghi@utwente.nl`

<sup>2</sup> Dept. Math. and Comp. Science, T.U. Eindhoven, Eindhoven, The Netherlands  
`p.p.v.liesdonk@tue.nl`

<sup>3</sup> Dept. ESAT/SCD-COSIC and IBBT, K.U. Leuven, Heverlee, Belgium

<sup>4</sup> Philips Research Laboratories, The Netherlands

**Abstract.** A hidden vector encryption scheme (HVE) is a derivation of identity-based encryption, where the public key is actually a vector over a certain alphabet. The decryption key is also derived from such a vector, but this one is also allowed to have “ $\star$ ” (or wildcard) entries. Decryption is possible as long as these tuples agree on every position except where a “ $\star$ ” occurs.

These schemes are useful for a variety of applications: they can be used as a building block to construct attribute-based encryption schemes and sophisticated predicate encryption schemes (for e.g. range or subset queries). Another interesting application – and our main motivation – is to create searchable encryption schemes that support queries for keywords containing wildcards.

Here we construct a new HVE scheme, based on bilinear groups of prime order, which supports vectors over any alphabet. The resulting ciphertext length is equally shorter than existing schemes, depending on a trade-off. The length of the decryption key and the computational complexity of decryption are both constant, unlike existing schemes where these are both dependent on the amount of non-wildcard symbols associated to the decryption key.

Our construction hides both the plaintext and public key used for encryption. We prove security in a selective model, under the decision linear assumption.

## 1 Introduction

With the growing popularity of outsourcing data to third-party data-centers (the cloud), enhancing the security of such remote data is of increasing interest. In an ideal world such data centers may be completely trustworthy, but in practice they may very well be curious for your secrets. To prevent this all data should be encrypted. However, this directly results in problems of selective data retrieval. If a data-center cannot read the stored information, it also cannot answer any search queries.

Consider the following scenario about storage of health care records. Assume that Alice wants to store her medical records on a server. Since these medical records are highly sensitive, Alice wants to control the access to these records in such a way that a legitimate doctor can only see specific parts. Now Alice either has to trust the server to honestly treat her records, or she should encrypt her records in such a way that specific information can only be found by specific doctors.

Searchable encryption is a technique that addresses the mentioned problem. In general we will consider the following public-key setting: Bob wants to send a document to Alice, but to get it to her he has to store it on an untrusted intermediary server. Before sending he encrypts the document with Alice’s public key. To make her interaction with the server easier he also adds some keywords describing the encrypted document. These keywords are also encrypted, but in

---

<sup>\*</sup> This work has been supported in part by the research program Sentinels of STW, under the project SEDAN 07630.

a special way. Later, Alice wants to retrieve all documents from this server containing a specific keyword. She uses her secret key to create a so-called trapdoor that she sends to the server. Using this trapdoor the server can circumvent the encryption of all the encrypted keywords that it has stored, but only just enough to learn whether the encrypted keyword was equal to the keyword Alice had in mind. If the server finds such a match it can return the encrypted document to Alice.

In many applications it is convenient to have some flexibility when searching, like searching for a subset of keywords or searching for multiple keywords at once using a wildcard. Existing solutions address searching with wildcards using a technique called *hidden vector encryption* (HVE) [7]. A HVE scheme is a variation of identity-based encryption where both the encryption and the decryption key are derived from a vector. Decryption can only be done if the vectors are the same in every element except for certain positions, which we call wildcard- or “don’t care”-positions. The relation with searchable encryption comes by viewing a keyword as a vector of symbols. For every keyword Bob will make a HVE encryption of a public message, using the keyword as a ‘public key’. The trapdoor Alice sends to the server is actually a decryption key derived from a keyword. The server can now try to decrypt the HVE encryptions; if the decryption works the server can conclude that two keywords were the same, except for the wildcard positions. Because of this relation this paper will focus on the construction of a HVE scheme.

There have been quite a few proposals for HVE schemes, most notably [3, 7, 14, 16, 18, 22]. These schemes have in general two drawbacks: Firstly, most of them are using *bilinear groups of composite order*, whereas the few schemes that do use the more efficient bilinear groups of prime order [3, 14, 18] are only capable of working with binary alphabets. Secondly, in all these schemes the *size of the ciphertext* is linear in the length of the vector it’s key is derived from. Thirdly, the *size of the decryption key* grows linearly in the amount of non-wildcard symbols. This directly influences the number of computations needed for decryption. Therefore, these schemes are inefficient for applications where the client wishes to query for keywords that contain just a few wildcard values.

## 1.1 Related work

Searchable data encryption was first popularized by the work of Song, Wagner and Perrig [23]. They propose a scheme that allows a client to create both ciphertexts and trapdoors (resulting is a symmetric-key setting), while a server can test whether there is an exact match between a given ciphertext and a trapdoor. Searchable encryption in the symmetric key setting was further developed by [9, 10, 12, 24] to enhance the security and the efficiency of the scheme. While these schemes are useful when you want to backup your own information on a server, the symmetric key makes them hard to use in a multi-user setting

In [5], Boneh et. al. consider searchable encryption in an asymmetric setting, called *public key encryption with keyword search* (PEKS). Here everybody can create an encrypted keyword, but only the owner of the secret key can create a trapdoor, thus making it relevant for multi-user applications. This setting has been enhanced in [2, 19]. The PEKS scheme has a very close connection to anonymous identity-based encryption as introduced in [6], This connection has been studied more thoroughly by [1]. For this reason, most work (including ours) on asymmetric searchable encryption has a direct use for identity-based encryption, and vice versa. Improved IBE schemes useful for searchable encryption have been proposed in [8, 11, 17, 18].

These schemes are usable for equality search, i.e. a message can be decrypted if the trapdoor keyword and the associated keyword of the message are the same. In [13, 20] the concept of attribute-based encryption is introduced. Here, multiple keywords are used at encryption time, but a trapdoor can be made to decrypt using (almost) any access structure. Both schemes lack the anonymity property however, which makes them unusable for searchable encryption.

Adding anonymity results in schemes that offer so-called called *hidden vector encryption*, introduced in [8, 21]; in these schemes the trapdoor is allowed to have wildcard symbols “ $\star$ ” that matches any possible keyword in the encryption, They all use rather inefficient bilinear groups of a composite order. The same holds for [16, 22], which introduce inner product and predicate en-

ryption. Finally, [14] provides a solution for binary hidden-vector encryption that is based purely on bilinear groups of prime order.

## 1.2 Our results

Here, we propose a public-key hidden vector encryption (HVE) scheme, which queries encrypted messages for keywords that contain wildcard entries.

Our contributions in comparison to previous HVE schemes are as follows:

- Our construction uses bilinear groups of prime order, while [7, 21] use hardness assumptions based on groups of composite order. Our scheme can also take keywords over any alphabet, unlike [3, 14, 18] that only take binary symbols.
- The size of the decryption key and the computational complexity for decrypting ciphertexts is constant, while in earlier papers these grow linearly in the number of *non*-wildcard entries of the vector.
- The size of the ciphertext is approximately limited to one group element for every wildcard we are willing to allow (chosen at encryption time), where in previous schemes the ciphertext needs one group element for every symbol in the vector.

Our construction is proven to be semantically secure and keyword-hiding in the selective-keyword model, assuming the Decision Linear assumption [4] holds.

The rest of the paper is organized as follows: in Section 2 we discuss the security definitions we will use and the building blocks required. In Section 3 we introduce our HVE and prove its security properties. In Section 4 we analyze the performance of our scheme and compare it with previous results.

## 2 Preliminaries

Below, we review searchable data encryption, its relation to hidden vector encryption and their security properties.. In addition we review the definition of bilinear group and the Decision linear (DLin) assumption.

### 2.1 Searchable Data Encryption

Our ultimate goal is to provide a technique for searching with wildcards. As a basis we will use the concept of *public key encryption with keyword search* as introduced by Boneh et. al.[5]. Suppose Bob wants to send Alice an encrypted e-mail  $m$  in such a way that it is indexed by some searchable keywords  $W_1, \dots, W_k$ . Then Bob would make a construction of the form

$$(E_{pk}(m) \parallel S_{pk}(W_1) \parallel \dots \parallel S_{pk}(W_k)),$$

where  $E$  is a regular asymmetric encryption function,  $pk$  is Alice’s public key, and  $S$  is a special *searchable encryption* function. Alice can now – using her secret key – create a trapdoor to search for emails sent to her containing a specific keyword  $\bar{W}$ . The e-mail server can now test whether the searchable encryption and the trapdoor contain the same keyword and forward the encrypted mail if this is the case. During this process the server learns nothing about the keywords used.

If the trapdoor-keyword is allowed to have wildcard keywords we can get a much more flexible search. As an example, searching for the word ‘**ba\***’ results in encryptions with ‘bat’, ‘bad’ and ‘bag’. We can also do range queries: ‘200\*’ matches ‘2000’ up to ‘2009’ and ‘04/\*\*/2010’ matches the whole of April in 2010. These and other applications were first studied in [7].

**Definition 1.** *A non-interactive public key encryption with wildcard keyword search (wildcard PEKS) scheme consists of four probabilistic polynomial-time algorithms (KeyGen, Enc, Trapdoor, Test):*

- **Setup**( $\kappa$ ): Given a security parameter  $\kappa$  and a keyword-length  $L$  output a secret key  $sk$  and a public key  $pk$ .
- **Enc**( $pk, W$ ): Given a keyword  $W$  of length at most  $L$  characters, and the public key  $pk$  output a searchable encryption  $S_{pk}(W)$ .
- **Trapdoor**( $sk, \bar{W}$ ): Given a keyword  $\bar{W}$  of length at most  $L$  characters containing wildcard symbols  $\star$  and the secret key  $sk$  output a trapdoor  $T_{\bar{W}}$ .
- **Test**( $S_W, T_{\bar{W}}$ ): Given a searchable encryption  $S_W$  and a trapdoor  $T_{\bar{W}}$ , return ‘true’ if all non-wildcard characters are the same or ‘false’ otherwise.

Such a scheme can typically be made out of a so-called hidden-vector encryption scheme [7], using a variation of the **new-ibe-2-peks** transformation in [1]. If the HVE is semantically secure, then the constructed wildcard PEKS is computationally consistent, i.e. it gives false positives with a negligible probability. If the HVE is keyword-hiding, then the constructed wildcard PEKS does not leak any information about the keyword used to make a searchable encryption.

## 2.2 Hidden Vector Encryption

Let  $\Sigma$  be an alphabet. Let  $\star$  be a special symbol not in  $\Sigma$ . This star  $\star$  will play the role of a wildcard or “don’t care” symbol. Define  $\Sigma_\star = \Sigma \cup \{\star\}$ . The public key used to create a ciphertext will be a vector  $W = (w_1, \dots, w_L) \in \Sigma^L$ , called *attribute vector*. Every decryption key will also be created from a vector  $\bar{W} = (\bar{w}_1, \dots, \bar{w}_L) \in \Sigma_\star^L$ . Decryption is possible if for all  $i = 1 \dots L$  either  $w_i = \bar{w}_i$  or  $\bar{w}_i = \star$ .

**Definition 2 (HVE).** A Hidden Vector Encryption (HVE) scheme consists of the following four probabilistic polynomial-time algorithms (**Setup**, **Extract**, **Enc**, **Dec**):

- **Setup**( $\kappa, \Sigma, L$ ): Given a security parameter  $\kappa$ , an alphabet  $\Sigma$ , and a vector-length  $L$ , output a master secret key  $msk$  and public parameters  $param$ .
- **Extract**( $msk, \bar{W}$ ): Given an attribute vector  $\bar{W} \in \Sigma_\star^L$  and the master secret key  $msk$ , output a decryption key  $T_{\bar{W}}$ .
- **Enc**( $param, W, M$ ): Given an attribute vector  $W \in \Sigma^L$ , a message  $M$ , and the public parameters  $param$ , output a ciphertext  $S_{W,M}$ .
- **Dec**( $S_{W,M}, T_{\bar{W}}$ ): Given a ciphertext  $S_{W,M}$  and a decryption key  $T_{\bar{W}}$ , output a message  $M$ ,

These algorithms must satisfy the following consistency constraint:

$$\text{Dec}(\text{Enc}(param, W, M), \text{Extract}(msk, \bar{W})) = M \quad \text{if } w_i = \bar{w}_i \vee \bar{w}_i = \star \text{ for } i = 1 \dots L. \quad (1)$$

**Security Definitions** Here, we define the notion of security for hidden vector encryption schemes. Informally, this security definition states that a scheme reveals no non-trivial information to an adversary. In other words there is a separation between *semantic security* – which formalizes the notion that an adversary cannot learn any information about the message that has been encrypted – and *keyword hiding* – which formalizes the notion that he cannot learn non-trivial information about the keyword or vector used for encryption. These notions are both integrated into our security definition. This definition uses the selective model, in which the adversary commits to the encryption vector at the beginning of the “game”.

**Definition 3 (Semantic Security).** A HVE scheme (**Setup**, **Extract**, **Enc**, **Dec**) is semantically secure in the selective model if for all probabilistic polynomial-time adversaries  $\mathcal{A}$ ,

$$\left| \Pr[\mathbf{Exp}_{\mathcal{A}}(\kappa) = 1] - \frac{1}{2} \right| < \epsilon(\kappa)$$

for some negligible function  $\epsilon(\kappa)$ , where  $\mathbf{Exp}_{\mathcal{A}}(\kappa)$  is the following experiment:

- **Init.** The adversary  $\mathcal{A}$  chooses an alphabet  $\Sigma$ , a length  $L$  and announces two attribute vectors  $W_0^*, W_1^* \in \Sigma^L$ , different in at least one position, that it wishes to be challenged upon.
- **Setup.** The challenger runs  $\text{Setup}(\kappa, \Sigma, L)$ , which outputs a set of public parameters  $\text{param}$  and a master secret key  $\text{msk}$ . The challenger then sends  $\text{param}$  to the adversary  $\mathcal{A}$ .
- **Query Phase I.** In this phase  $\mathcal{A}$  adaptively issues key extraction queries for attribute vectors  $\bar{W} \in \Sigma_\star^L$ , under the restriction that  $\bar{w}_i \neq w_{0i}^*$  and  $\bar{w}_i \neq w_{1i}^*$  for at least one  $\bar{w}_i \neq \star$ . Given an attribute vector  $\bar{W}$  the challenger runs  $\text{Extract}(\text{msk}, \bar{W})$  which outputs a decryption key  $T_{\bar{W}}$ . The challenger then sends the  $T_{\bar{W}}$  to  $\mathcal{A}$ .
- **Challenge.** Once  $\mathcal{A}$  decides that the query phase is over,  $\mathcal{A}$  picks a pair of messages  $(M_0, M_1)$  on which it wishes to be challenged and sends them to the challenger. Given the challenge message  $(M_0, M_1)$  and the challenge attribute vectors  $(W_0^*, W_1^*)$ , the adversary flips a fair coin  $\nu \in_R \{0, 1\}$ , and invokes the  $\text{Enc}(\text{param}, W_\nu^*, M_\nu)$  algorithm to output  $S_{W_\nu^*, M_\nu}$ . The challenger then sends  $S_{W_\nu^*, M_\nu}$  to  $\mathcal{A}$ .
- **Query Phase II.** Identical to Query Phase I.
- **Output.** Finally, the adversary outputs a bit  $\nu'$  which represents its guess for bit  $\nu$ . If  $\nu = \nu'$  then return 1, else return 0.

Intuitively, this experiment simulates a worst-case scenario attack, where the adversary has access to a lot of information: it knows that the challenge ciphertext is either an encryption of  $M_0$  under  $W_0^*$  or an encryption of  $M_1$  under  $W_1^*$ , all of which are chosen by him. In addition, it is allowed to know any decryption key that does not directly decrypt the challenge. Query phase I allows the adversary to choose the challenge messages based on decryption keys it already knows. Query phase II allows the adversary to ask for more decryption keys based on the challenge ciphertext it received.

If the encryption scheme would have a flaw and leak even a bit of information, a smart adversary would choose the message and attribute vector in such a way that this weakness would come to light. Thus the statement that no adversary can do significantly better than guessing implies that the encryption scheme does not leak information.

We wish to note that there is a stronger notion of security – the non-selective model – where the adversary chooses  $W_0^*$  and  $W_1^*$  in the challenge phase. This allows the adversary to make those dependent on the public parameters and on known decryption keys. Creating a secure HVE in that setting is still an open problem.

### 2.3 Bilinear Groups

**Definition 4 (Bilinear Group).** We say that a cyclic group  $\mathbb{G}$  of prime order  $q$  with generator  $g$  is a bilinear group if there exists a group  $\mathbb{G}_T$  and a map  $e$  such that

- $(\mathbb{G}_T, \cdot)$  is also a cyclic group, of prime order  $q$ ,
- $e(g, g)$  is a generator of  $\mathbb{G}_T$  (non-degenerate).
- $e$  is an bilinear map  $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ . In other words, for all  $u, v \in \mathbb{G}_1$  and  $a, b \in \mathbb{Z}_q^*$ , we have  $e(u^a, v^b) = e(u, v)^{ab}$ .

Additionally, we require that the group actions and the bilinear map can be computed in polynomial time. A bilinear map that satisfies these conditions is called admissible.

Our scheme is proven secure under the Decision Linear assumption (DLin), which has been introduced by [4]:

**Definition 5 (Decision Linear Assumption).** There exist bilinear groups  $\mathbb{G}$  such that for all probabilistic polynomial-time algorithms  $\mathcal{A}$ ,

$$\left| \Pr[\mathcal{A}(\mathbb{G}, g, g^a, g^b, g^{ac}, g^d, g^{b(c+d)}) = 1] - \Pr[\mathcal{A}(\mathbb{G}, g, g^a, g^b, g^{ac}, g^d, g^r) = 1] \right| < \epsilon(\kappa)$$

for some negligible function  $\epsilon(\kappa)$ , where the probabilities are taken over all possible choices of  $a, b, c, d, r \in \mathbb{Z}_q^*$ .

Informally, the assumption states that given a bilinear group  $\mathbb{G}$  and elements  $g^a, g^b, g^{ac}, g^d$  it is hard to distinguish  $h = g^{b(c+d)}$  from a random element in  $\mathbb{G}$ . The Decision Linear assumption implies the decision bilinear Diffie-Hellman assumption. The best known algorithm to solve the Decision Linear Problem is to compute a discrete logarithm in  $\mathbb{G}$ .

### 3 Construction

Before we present our scheme we will first explain the intuition behind it.

#### 3.1 Intuition

A common construction for encryption schemes is to hide a message using a one-time pad construction, i.e. multiplying the message with a random session key. In our HVE scheme we choose a session key based on all the elements of the encryption-vector, while the decryption key contains the information to cancel out the effect of symbols at unwanted wildcard-positions. Ciphertext and decryption key together can thus recover the session key.

Suppose we have an encryption-vector  $W = (w_1, \dots, w_L)$  and a decryption-vector  $\bar{W} = (\bar{w}_1, \dots, \bar{w}_L)$ , both consisting of  $L$  elements. Let  $J = \{j_1, \dots, j_n\} \subset \{1, \dots, L\}$  denote the position of wildcards in the decryption key (i.e.  $J = \{4, 5\}$  for '04/\*\*/2010'). We now consider the polynomial  $\prod_{j \in J} (i - j)$  which equals zero in all the wildcard positions. Now the following statement are equal:

$$w_i = \bar{w}_i \vee \bar{w}_i = \star \text{ for } i = 1 \dots L \quad (2)$$

$$\sum_{i=1}^L w_i \prod_{j \in J} (i - j) = \sum_{\substack{i=1 \\ i \notin J}}^L \bar{w}_i \prod_{j \in J} (i - j), \quad (3)$$

Given that we can expand  $\prod_{j \in J} (i - j) = \sum_{k=0}^n a_k i^k$ , where the  $a_k$  are coefficients only dependent on  $J$ , this is also equivalent with

$$\sum_{k=0}^n a_k \sum_{i=1}^L w_i i^k = \sum_{\substack{i=1 \\ i \notin J}}^L \bar{w}_i \prod_{j \in J} (i - j). \quad (4)$$

Practically we want to hide computations in the exponents of group elements. So instead of  $w_i$  we work with  $U_i^{w_i}$  for some random group element  $U_i$ . Equation (4) still holds if:

$$\prod_{k=0}^n \left( \prod_{i=1}^L U_i^{w_i i^k} \right)^{a_k} = \prod_{\substack{i=1 \\ i \notin J}}^L U_i^{\bar{w}_i \prod_{j \in J} (i - j)} \quad (5)$$

In the ciphertext we introduce new randomness that can only be removed if Eq. (5) – and thus Eq. (2) – is true. For this to work we put pieces of this equation in the ciphertext and in the decryption key, such that it can be evaluated at decryption time. The whole right side is included in the decryption key. The left side of Eq. (5) can be computed using the following two sets of elements

- $J$  is included with the decryption key, which allows for computation of all the coefficients  $a_k$ ,
- for  $k = 1, \dots, n$ , the term  $\prod_{i=1}^L U_i^{w_i i^k}$  is included in elements of the ciphertext.

The ciphertext has to include  $n$  almost similar elements – only different in the value for  $k$  – since it is infeasible to compute them from a single source. However, the amount of wildcards  $n$  used for decryption is unknown at decryption time. The best we can do is choose an upper bound  $N$  to  $n$  and include  $N$  elements. Choosing a small  $N$  results in smaller ciphertexts, but also in less

flexibility when creating decryption keys. Choosing  $N = L$  results in larger ciphertexts, but also allows for the creation of decryption keys that consist of only wildcards.

We can reconstruct the coefficients  $a_k$  of the polynomial  $\prod_{j \in J} (x - j)$  that occurs in (4) by using Viète's formulas:

$$a_{n-k} = (-1)^k \sum_{1 \leq i_1 < i_2 < \dots < i_k \leq n} j_{i_1} j_{i_2} \dots j_{i_k}, \quad 0 \leq k \leq n \quad (6)$$

where  $n = |J|$ . If  $J$  is clear from the context we will write  $a_i$ .

For instance when  $J = \{j_1, j_2, j_3\}$  we get for the polynomial  $(x - j_1)(x - j_2)(x - j_3)$ ,

$$\begin{aligned} a_3 &= 1 \\ a_2 &= -(j_1 + j_2 + j_3) \\ a_1 &= (j_1 j_2 + j_1 j_3 + j_2 j_3) \\ a_0 &= -j_1 j_2 j_3 \end{aligned}$$

### 3.2 Construction

We are now ready to give our construction for a hidden vector encryption scheme. Without loss of generality, we look at vectors of maximum length  $L$  over a fixed alphabet  $\Sigma \subset \mathbb{Z}_q^*$ . Other alphabets – like ASCII characters – can always be mapped onto such a subset. In addition, we need to pick an upper bound  $N$  to the number of wildcards that are allowed in a decryption vector. While this upper bound can be equal to  $L$ , performance increases if  $N \ll L$ .

This construction allows for shorter vectors of a length  $\ell < L$ . Intuitively we'll pad these vectors with zeroes up to a length  $L$ , but in practice this padding can be safely ignored in the computations.

Our scheme comprises of the following algorithms:

- **Setup**( $\kappa, \Sigma, L$ ): First, choose an upper bound  $N \leq L$  to the number of wildcard symbols in decryption vectors. Next, given security parameter  $\kappa$ :
  1. Generate a bilinear group  $\mathbb{G}$  of a large prime order  $q$  and choose a bilinear map  $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ .
  2. Pick  $L + 1$  random elements  $V_0, U_1, \dots, U_L \in_R \mathbb{G}$ .
  3. Pick random exponents  $\alpha, \beta_1, \beta_2, (x_1, \dots, x_N) \in_R \mathbb{Z}_q$ .
  4. Let  $\Omega_1 = e(g, V_0)^{\alpha\beta_1}$  and  $\Omega_2 = e(g, V_0)^{\alpha\beta_2}$ .
  5. Let  $V_j = V_0^{x_j}$  for  $j = 1, \dots, N$ .

The public parameters are:

$$param = \left( (V_0, V_1, \dots, V_N), (U_1, \dots, U_L), g^\alpha, \Omega_1, \Omega_2, q, \mathbb{G}, \mathbb{G}_T, e(\cdot, \cdot) \right)$$

The master secret key is  $msk = (\alpha, \beta_1, \beta_2, (x_1, \dots, x_N))$ .

- **Extract**( $msk, \bar{W}$ ): Let  $\bar{W} = (\bar{w}_1, \dots, \bar{w}_\ell) \in \Sigma_\star^\ell$ , where  $\ell \leq L$ . Assume that  $W$  contains  $n \leq N$  wildcards which occur at positions  $J = \{j_1, \dots, j_n\}$ . Pick a random  $s \in_R \mathbb{Z}_q$  and compute:  $s_1 = \beta_1 + s, s_2 = \beta_2 + s$ . By means of Viète's formulas  $a_i$  for  $i = 1, \dots, n$ , first compute  $t = \sum_{k=0}^n x_k a_k$  and then the decryption key  $T_{\bar{W}}$  (where  $x_0 = 1$ ):

$$T_{\bar{W}} = \left( \begin{array}{l} T_0 = g^{\frac{\alpha s}{t}} \\ T_1 = V_0^{s_1} \prod_{i=1}^{\ell} U_i^{\frac{s}{t} \prod_{k=1}^n (i-j_k) \bar{w}_i} \\ T_2 = V_0^{\alpha s_2} \prod_{i=1}^{\ell} U_i^{\frac{\alpha s}{t} \prod_{k=1}^n (i-j_k) \bar{w}_i} \\ J = \{j_1, \dots, j_n\} \end{array} \right).$$

- $\text{Enc}(param, W, M)$ : Let  $W = (w_1, \dots, w_\ell) \in \Sigma^\ell$ , where  $\ell \leq L$  and  $M \in \mathbb{G}_T$  a message. Pick two random values  $r_1, r_2 \in_R \mathbb{Z}_q^*$ . The ciphertext  $S_{W,M}$  is:

$$S_{W,M} = \left( \hat{C} = M\Omega_1^{r_1}\Omega_2^{r_2}, \begin{pmatrix} C_0 = (V_0 \prod_{i=1}^{\ell} U_i^{w_i})^{r_1+r_2} \\ C_1 = (V_1 \prod_{i=1}^{\ell} U_i^{i w_i})^{r_1+r_2} \\ \vdots \\ C_N = (V_N \prod_{i=1}^{\ell} U_i^{i^N w_i})^{r_1+r_2} \end{pmatrix}, \begin{pmatrix} g^{\alpha r_1} \\ g^{r_2} \end{pmatrix} \right).$$

- $\text{Dec}(S_{W,M}, T_{\bar{W}})$ : Given a decryption key  $T_{\bar{W}}$  and a ciphertext  $S_{W,M}$ , first use  $J$  to compute Viète's formulas  $a_i$   $i = 1, \dots, n$ , then decrypt the message as:

$$M = \hat{C} \frac{e(T_0, \prod_{k=0}^n C_k^{a_k})}{e(T_1, g^{\alpha r_1})e(T_2, g^{r_2})}$$

### 3.3 Correctness

We now show that the Dec algorithm indeed returns the correct message when using a decryption key that should be able to decrypt a given ciphertext. Without loss of generality we assume that the vectors contain  $l$  symbols and that there are  $n$  wildcards at positions  $\{j_1, \dots, j_n\}$ . Then

$$\begin{aligned} e(T_0, \prod_{k=0}^n C_k^{a_k}) &= e(g^{\frac{\alpha s}{\sum_{m=0}^n x_m a_m}}, \prod_{k=0}^n V_k^{a_k(r_1+r_2)}) e(g^{\frac{\alpha s}{\sum_{m=0}^n x_m a_m}}, \prod_{k=0}^n \prod_{i=1}^{\ell} U_i^{i^k a_k w_i(r_1+r_2)}) \\ &= \prod_{k=0}^n \left( e(g, V_0)^{\frac{\alpha s(r_1+r_2)x_k a_k}{\sum_{m=0}^n x_m a_m}} \prod_{i=1}^{\ell} e(g, U_i)^{\frac{\alpha s(r_1+r_2)w_i i^k a_k}{\sum_{m=0}^n x_m a_m}} \right) \\ &= e(g, V_0)^{\frac{\alpha s(r_1+r_2)\sum_{k=0}^n x_k a_k}{\sum_{m=0}^n x_m a_m}} \prod_{i=1}^{\ell} e(g, U_i)^{\frac{\alpha s(r_1+r_2)w_i \sum_{k=0}^n i^k a_k}{\sum_{m=0}^n a_m x_m}} \\ &= e(g, V_0)^{\alpha s(r_1+r_2)} \prod_{i=1}^{\ell} e(g, U_i)^{\frac{\alpha s(r_1+r_2)w_i \prod_{k=1}^n (i-j_k)}{\sum_{m=0}^n a_m x_m}} \end{aligned} \quad (7)$$

where for (7) we use that  $\sum_{k=0}^n i^k a_k = \prod_{k=1}^n (i - j_k)$ .

$$\begin{aligned} e(T_1, g^{\alpha r_1}) &= e(V_0, g)^{\alpha r_1 s_1} e\left(\prod_{i=1}^{\ell} U_i^{\frac{s \prod_{k=1}^n (i-j_k) \bar{w}_i}{\sum_{m=0}^n a_m x_m}}, g^{\alpha r_1}\right) \\ &= \Omega_1^{r_1} e(g, V_0)^{\alpha s r_1} \prod_{i=1}^{\ell} e(g, U_i)^{\frac{\alpha s r_1 \prod_{k=1}^n (i-j_k) \bar{w}_i}{\sum_{m=0}^n a_m x_m}} \end{aligned} \quad (8)$$

$$\begin{aligned} e(T_2, g^{r_2}) &= e(V_0, g)^{\alpha r_2 s_2} e\left(\prod_{i=1}^{\ell} U_i^{\frac{\alpha s \prod_{k=1}^n (i-j_k) \bar{w}_i}{\sum_{m=0}^n a_m x_m}}, g^{r_2}\right) \\ &= \Omega_2^{r_2} e(g, V_0)^{\alpha s r_2} \prod_{i=1}^{\ell} e(g, U_i)^{\frac{\alpha s r_2 \prod_{k=1}^n (i-j_k) \bar{w}_i}{\sum_{m=0}^n a_m x_m}} \end{aligned} \quad (9)$$

$$e(T_{n+1}, g^{\alpha r_1})e(T_{n+2}, g^{r_2}) = \Omega_1^{r_1}\Omega_2^{r_2} e(g, V_0)^{\alpha s(r_1+r_2)} \prod_{i=1}^{\ell} e(g, U_i)^{\frac{\alpha s(r_1+r_2)\bar{w}_i \prod_{k=1}^n (i-j_k)}{\sum_{m=0}^n a_m x_m}} \quad (10)$$

If the decryption key is valid, then  $w_i = \bar{w}_i$  when  $i \notin \{j_1, \dots, j_n\}$ . Thus

$$\hat{C} \frac{e(T_0, \prod_{k=0}^n C_k^{a_k})}{e(T_1, g^{\alpha r_1})e(T_2, g^{r_2})} = \frac{M\Omega_1^{r_1}\Omega_2^{r_2}e(T_0, \prod_{k=0}^n C_k^{a_k})}{e(T_{n+1}, g^{\alpha r_1})e(T_{n+2}, g^{r_2})} = M \quad (11)$$

### 3.4 Semantic Security

**Theorem 1.** *The hidden vector encryption scheme in Section 3 is semantically secure in the selective model assuming that the Decision Linear assumption holds in group  $\mathbb{G}$ .*

*Proof.* Suppose there exists a PPT adversary  $\mathcal{A}$  that can break the selective semantic security, i.e.  $\mathcal{A}$  has an advantage in the experiment of Definition 3 larger than some non-negligible  $\epsilon$ . We build an algorithm  $\mathcal{B}$  that uses  $\mathcal{A}$  to solve the Decision Linear problem in  $\mathbb{G}$ .

The challenger selects a bilinear group  $\mathbb{G}$  of prime order  $q$  and chooses a generator  $g \in \mathbb{G}$ , the group  $\mathbb{G}_T$  and an efficient bilinear map  $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ . Then the challenger picks four random values  $a, b, c, d \in_R \mathbb{Z}_q^*$ , computes  $Z_0 = g^{b(c+d)}$  and chooses  $Z_1 \in_R \mathbb{G}$ . After flipping a fair coin  $\nu \in_R \{0, 1\}$  the challenger hands the tuple  $(g, g^a, g^b, g^{ac}, g^d, Z_\nu)$  to  $\mathcal{B}$ . Algorithm  $\mathcal{B}$ 's goal is to guess  $\nu$  with a better chance of being correct than  $\frac{1}{2}$ . In order to come up with a guess,  $\mathcal{B}$  interacts with adversary  $\mathcal{A}$  in a selective semantic security experiment as follows:

**Init.** Adversary  $\mathcal{A}$  chooses an alphabet  $\Sigma \subset \mathbb{Z}_q^*$ , a length  $L$  and announces two attribute vectors  $W_0^* \in \Sigma^{\ell_0}$ ,  $W_1^* \in \Sigma^{\ell_1}$ , where  $\ell_0, \ell_1 \leq L$ , which are different in at least one position.  $\mathcal{B}$  flips a coin  $\mu \in \{0, 1\}$ . Let  $W_\mu^* = (w_1^*, \dots, w_{\ell_\mu}^*)$ .

**Setup.**  $\mathcal{B}$  chooses an upper bound  $N \leq L$  to the number of wildcard symbols. Then  $\mathcal{B}$  picks random values  $v_0, u_1, \dots, u_L, x_1, \dots, x_N \in_R \mathbb{Z}_q^*$  and sets

$$V_j = (g^b)^{x_j v_0} g^{-\sum_{i=1}^{\ell_\mu} i^j u_i} \quad \text{for } j = 0, \dots, N$$

$$U_i = \begin{cases} g^{\frac{u_i}{w_i^*}} & \text{for } i = 1 \dots \ell_\mu \\ g^{u_i} & \text{for } i = \ell_\mu + 1, \dots, L, \end{cases}$$

where  $x_0 = 1$ .  $\mathcal{B}$  picks  $\sigma_1, \sigma_2, \sigma_3 \in_R \mathbb{Z}_q$  and computes  $\Omega_1 = e(g^a, V_0)^{\sigma_1 - \sigma_2}$  and  $\Omega_2 = e(g^{\sigma_3} (g^a)^{-\sigma_2}, V_0)$ . The public parameters are:

$$param = \left( (V_0, V_1, \dots, V_N), (U_1, \dots, U_L), g^a, \Omega_1, \Omega_2, q, \mathbb{G}, \mathbb{G}_T, e(\cdot, \cdot) \right)$$

The master secret key is implicitly given by

$$msk = (\alpha = a, t_1 = \sigma_1 - \sigma_2, t_2 = \frac{\sigma_3}{a} - \sigma_2, (x_1, \dots, x_N)).$$

**Query Phase I.** In this phase  $\mathcal{A}$  adaptively issues key extraction queries. Each time  $\mathcal{A}$  queries for the decryption key of an attribute vector  $\bar{W} = (\bar{w}_1, \dots, \bar{w}_\ell) \in \Sigma_\star^\ell$ , consisting of  $\ell \leq L$  symbols and  $n \leq N$  wildcards at positions  $J = \{j_1, \dots, j_n\}$ , algorithm  $\mathcal{B}$  responds by computing

$$T_0 = (g^a)^{\frac{\sigma_2}{\sum_{m=0}^n x_m a_m}},$$

$$T_1 = V_0^{\sigma_1} \prod_{i=1}^{\ell} U_i^{\frac{\sigma_2 \prod_{k=1}^n (i-j_k) \bar{w}_i}{\sum_{m=0}^n x_m a_m}},$$

$$T_2 = (g^b)^{\sigma_3 v_0} g^{-\sigma_3 \sum_{i=1}^{\ell_\mu} u_i} (g^a)^{\frac{\sigma_2 \sum_{i=1}^{\ell_\mu} \frac{u_i}{w_i^*} \prod_{k=1}^n (i-j_k) \bar{w}_i}{\sum_{m=0}^n x_m a_m} + \frac{\sigma_2 \sum_{i=\ell_\mu+1}^L u_i \prod_{k=1}^n (i-j_k) \bar{w}_i}{\sum_{m=0}^n x_m a_m}},$$

which is basically a correct trapdoor for  $\bar{W}$  with  $s = \sigma_2$ .  $\mathcal{B}$  returns to  $\mathcal{A}$  the decryption key

$$T_{\bar{W}} = (T_0, T_1, T_2, J). \quad (12)$$

**Challenge.** Once  $\mathcal{A}$  decides that the query phase is over,  $\mathcal{A}$  picks a pair of messages  $M_0, M_1 \in \mathbb{G}_T$  on which it wishes to be challenged.  $\mathcal{B}$  computes  $S_{W_\mu^*, M_\mu}$  by first computing

$$\hat{C} = M_\mu \cdot e(g^{ac}, g^b)^{\sigma_1 v_0} \cdot e(g^{ac}, g)^{(\sigma_1 - \sigma_2) \sum_{i=0}^{\ell_\mu} u_i} \cdot e(g^a, g^d)^{\sigma_2 \sum_{i=0}^{\ell_\mu} u_i} \cdot e(g^b, g^d)^{\sigma_3 v_0} \cdot e(g^d, g)^{\sigma_3 \sum_{i=0}^{\ell_\mu} u_i} \cdot e(g^a, Z_\nu)^{\sigma_2 v_0} \quad (13)$$

and then computing  $C_0 = Z_\nu^{v_0}$  and  $C_k = Z_\nu^{x_k v_0}$  for  $k = 1, \dots, N$ .  $\mathcal{B}$  sends the challenge ciphertext

$$S_{W_\mu^*, M_\mu} = \left( \hat{C}, \{C_k\}_{k=0}^N, \begin{pmatrix} g^{ac} \\ g^d \end{pmatrix} \right), \quad (14)$$

to  $\mathcal{A}$ . When  $\nu = 0$  this is actually a correct encryption of  $M_\mu$  under  $W_\mu^*$  with  $r_1 = c$  and  $r_2 = d$ .

**Query Phase II.** In Query Phase II  $\mathcal{B}$  behaves exactly the same as in Query Phase I.

**Output.** Eventually,  $\mathcal{A}$  outputs a bit  $\mu'$ .

Finally,  $\mathcal{B}$  outputs 1 if  $\mu' = \mu$  and 0 if  $\mu' \neq \mu$ .

We will now analyze the probability of success for algorithm  $\mathcal{B}$ . First, note that if  $\nu = 0$ , then  $\mathcal{B}$  will behave correctly as a challenger to  $\mathcal{A}$ . Thus,  $\mathcal{A}$  will have probability of  $\frac{1}{2} + \epsilon$  of guessing  $\mu$ . Next note that if  $\nu = 1$ , then  $Z_\nu$  is random in  $\mathbb{G}$  and  $S_{W_\mu^*, M_\mu}$  is independent from  $\mu$ , thus  $\mathcal{A}$  will have a probability of  $\frac{1}{2}$  of guessing  $\mu$ .

To conclude the proof we have

$$\begin{aligned} & \left| \Pr[\mathcal{B}(\mathbb{G}, g, g^a, g^b, g^{ac}, g^d, g^{b(c+d)}) = 1] - \Pr[\mathcal{B}(\mathbb{G}, g, g^a, g^b, g^{ac}, g^d, g^r) = 1] \right| \\ & \geq \left| \Pr[\nu = 0 \wedge \mu' = \mu] - \Pr[\nu = 1 \wedge \mu' = \mu] \right| \\ & = \left| \frac{1}{2} \Pr[\mu' = \mu \mid \nu = 0] - \frac{1}{2} \Pr[\mu' = \mu \mid \nu = 1] \right| \\ & = \frac{1}{2} \left| \Pr[\mathbf{Exp}_{\mathcal{A}}(\kappa) = 1] - \frac{1}{2} \right| \\ & \geq \frac{1}{2} \epsilon, \end{aligned}$$

which is non-negligible, contradicting the Decision Linear Assumption.  $\square$

## 4 Conclusion

We presented a new hidden vector encryption scheme which can work as a wildcard searchable encryption scheme that is more efficient than existing schemes in some scenarios. The tables below summarize the efficiency of our scheme when compared to other schemes. The scheme is proven selectively secure in the sense of hiding the contents of the message and hiding the keywords associated to the message. This is the same model as is used in the other schemes in the literature. A hidden vector encryption scheme that is secure in the adaptive standard model is still an open problem, as is finding any other construction for wildcard searchable encryption in that model.

The following table compares the performance of our scheme with existing searchable encryption schemes from the point of view of memory requirement. Table 1 shows that constructing the decryption key is more efficient than the existing schemes. Moreover, since  $N$  is always less than  $\ell$  (depending on the application scenario), the ciphertext can be constructed in a more efficient way.

Schemes	Size of ciphertext	Size of Decryption key	Size of public parameters	Maximum allowed Wildcards
Boneh, Waters [7] Katz et al. [16]	$2\ell + 2$	$2(\ell - n) + 1$	$3L + 3$	Arbitrary
Shi, Waters [22]	$\ell + 4$	$\ell - n + 3$	$4L + 2$	Arbitrary
Iovino, Persiano [14] Blundo et al. [3]	$2\ell + 2$	$\ell - n + 3$	$2L + 4$	Arbitrary
Nishide et al. [18]	$\ell + 2$	$\ell + 1$	$3L + 1$	Arbitrary
This Work	$N + 4$	3	$L + N + 1$	$N$

**Table 1.** Comparison of the performance of our scheme with existing searchable encryption schemes from the memory requirement point of view. The notation in this table is as follows:  $\ell$ : the length of the (ciphertext or decryption key) keyword,  $L$ : the maximum allowed number of entries in the ciphertext keyword,  $n$ : the number of wildcard entries,  $N$ : the maximum allowed number of wildcard entries.

The next table compares the performance of our scheme with existing searchable encryption schemes from the point of view of decryption cost. Table 2 shows that the decryption cost in our scheme is constant and less than other schemes since only three pairings is required for the decryption.

Schemes	Number of pairings for decryption	Order of bilinear group	Alphabet of entries
Boneh, Waters [7] and Katz et al. [16]	$2(\ell - n) + 1$	Composite	Arbitrary
Shi, Waters [22]	$(\ell - n) + 3$	Composite	Arbitrary
Iovino, Persiano [14] Blundo et al. [3]	$2(\ell - n)$	Prime	Binary
Nishide et al. [18]	$\ell + 1$	Prime	Binary
This Work	3	Prime	Arbitrary

**Table 2.** Comparison of the performance of our scheme with existing searchable encryption schemes from the point of view of decryption cost. The notation in this table is as follows:  $\ell$ : the length of the (ciphertext or decryption key) keyword,  $n$ : the number of wildcard entries.

## References

1. Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions. *J. Cryptology*, 21(3):350–391, 2008.
2. Joonsang Baek, Willy Susilo, and Jianying Zhou. New constructions of fuzzy identity-based encryption. In Feng Bao and Steven Miller, editors, *ASIACCS*, pages 368–370. ACM, 2007.
3. Carlo Blundo, Vincenzo Iovino, and Giuseppe Persiano. Private-key hidden vector encryption with key confidentiality. In Juan A. Garay, Atsuko Miyaji, and Akira Otsuka, editors, *CANS*, volume 5888 of *Lecture Notes in Computer Science*, pages 259–277. Springer, 2009.
4. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew K. Franklin, editor, *CRYPTO*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer, 2004.
5. Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 506–522. Springer, 2004.

6. Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. *SIAM J. Comput.*, 32(3):586–615, 2003.
7. Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In Salil P. Vadhan, editor, *TCC*, volume 4392 of *Lecture Notes in Computer Science*, pages 535–554. Springer, 2007.
8. Xavier Boyen and Brent Waters. Anonymous hierarchical identity-based encryption (without random oracles). In Cynthia Dwork, editor, *CRYPTO*, volume 4117 of *Lecture Notes in Computer Science*, pages 290–307. Springer, 2006.
9. Yan-Cheng Chang and Michael Mitzenmacher. Privacy preserving keyword searches on remote encrypted data. In John Ioannidis, Angelos D. Keromytis, and Moti Yung, editors, *ACNS*, volume 3531 of *Lecture Notes in Computer Science*, pages 442–455, 2005.
10. Reza Curtmola, Juan A. Garay, Seny Kamara, and Rafail Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. In Juels et al. [15], pages 79–88.
11. Craig Gentry. Practical identity-based encryption without random oracles. In Serge Vaudenay, editor, *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 445–464. Springer, 2006.
12. Eu-Jin Goh. Secure indexes. Cryptology ePrint Archive, Report 2003/216, 2003. <http://eprint.iacr.org/>.
13. Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Juels et al. [15], pages 89–98.
14. Vincenzo Iovino and Giuseppe Persiano. Hidden-vector encryption with groups of prime order. In Steven D. Galbraith and Kenneth G. Paterson, editors, *Pairing*, volume 5209 of *Lecture Notes in Computer Science*, pages 75–88. Springer, 2008.
15. Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors. *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, October 30 - November 3, 2006*. ACM, 2006.
16. Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In Nigel P. Smart, editor, *EUROCRYPT*, volume 4965 of *Lecture Notes in Computer Science*, pages 146–162. Springer, 2008.
17. Eike Kiltz. From selective-id to full security: The case of the inversion-based boneh-boyen ibe scheme. Cryptology ePrint Archive, Report 2007/033, 2007. <http://eprint.iacr.org/>.
18. Takashi Nishide, Kazuki Yoneyama, and Kazuo Ohta. Attribute-based encryption with partially hidden ciphertext policies. *IEICE Transactions*, 92-A(1):22–32, 2009.
19. Hyun Sook Rhee, Jong Hwan Park, Willy Susilo, and Dong Hoon Lee. Improved searchable public key encryption with designated tester. In Wanqing Li, Willy Susilo, Udaya Kiran Tupakula, Reihaneh Safavi-Naini, and Vijay Varadharajan, editors, *ASIACCS*, pages 376–379. ACM, 2009.
20. Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 457–473. Springer, 2005.
21. Elaine Shi, John Bethencourt, Hubert T.-H. Chan, Dawn Xiaodong Song, and Adrian Perrig. Multi-dimensional range query over encrypted data. In *IEEE Symposium on Security and Privacy*, pages 350–364. IEEE Computer Society, 2007.
22. Elaine Shi and Brent Waters. Delegating capabilities in predicate encryption systems. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfssdóttir, and Igor Walukiewicz, editors, *ICALP (2)*, volume 5126 of *Lecture Notes in Computer Science*, pages 560–578. Springer, 2008.
23. Dawn Xiaodong Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In *IEEE Symposium on Security and Privacy*, pages 44–55. IEEE Computer Society, 2000.
24. Brent R. Waters, Dirk Balfanz, Glenn Durfee, and Diana K. Smetters. Building an encrypted and searchable audit log. In *NDSS*. The Internet Society, 2004.