Searching Keywords with Wildcards on Encrypted Data

Saeed Sedghi[1], Peter van Liesdonk[2], Svetla Nikova[1], Pieter Hartel[1], and Willem Jonker[1]

[1] Universiteit Twente
[2] Technische Universiteit Eindhoven

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 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 datacenters (the cloud), enhancing the security of such remote data is of increasing interest. In an ideal world such datacenters 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 datacenter 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 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, 15, 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, 15, 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 Perring [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 [10, 11, 13, 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 [4], 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, 12, 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 [14, 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 anonimity property however, which makes them unusable for searchable encryption.

Adding anonimity results in schemes that offer so-called called *hidden vector encryption*, introduced in [9, 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 encryption. Finally, [15] 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, 15, 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 [5] 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.[4]. Suppose Bob wants to send Alice an encrypted e-mail m in such a way that it is indexed by some searchable keywords $W_1, \ldots, W_k$. Then Bob would make a construction of the form

$$(\mathsf{E}_{pk}(m) \parallel S_{pk}(W_1) \parallel \cdots \parallel S_{pk}(W_k)),$$

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 the following four probabilistic polynomial-time algorithms (KeyGen, Enc, Trapdoor, Test):*

- $\mathsf{Setup}(\kappa)$*: Given a security parameter* κ *and a keyword-length* L *output a* secret key sk *and a* public key pk*.*
- $\mathsf{Enc}(pk, W)$*: Given a keyword* W *of length at most* L *characters, and the public key* pk *output a searchable encryption* $S_{pk}(W)$*.*
- $\mathsf{Trapdoor}(sk, \bar{W})$*: Given a keyword* $\bar{W}$ *of length at most* L *characters containing wildcard symbols* $\star$ *and the secret key* sk *output a trapdoor* $T_{\bar{W}}$*.*
- $\mathsf{Test}(S_W, T_{\bar{W}})$*: 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 Σ be an alphabet. Let $\star$ be a special symbol not in Σ. 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 $W = (w_1, \ldots, w_L) \in \Sigma^L$, called *attribute vector*. Every decryption key will also be created from a vector $\bar{W} = (\bar{w}_1, \ldots, \bar{w}_L) \in \Sigma_\star^L$. Decryption is possible if for all $i = 1...L$ 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):*

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

These algorithms must satisfy the following consistency contraint:

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

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 works there is a seperation 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 use for encryption. These notions are both integrated into our security definition. As setting, we assume 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 $\mathcal{A}$,*

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

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

- **Init**. *The adversary $\mathcal{A}$ chooses an alphabet Σ, a length L and announces two attribute vectors $W_0^*, W_1^* \in \Sigma^L$, different in at least one position, that it whishes to be challenged upon.*
- **Setup**. *The challenger runs $\mathsf{Setup}(\kappa, \Sigma, L)$, which outputs a set of public parameters param and a master secret key msk. The challenger then sends 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 $\mathsf{Extract}(msk, \bar{W})$ 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 picks a fair coin $\beta \in_R \{0,1\}$, and invokes the $\mathsf{Enc}(param, W_\beta^*, M_\beta)$ algorithm to output $S_{W_\beta^*, M_\beta}$. The challenger then sends $S_{W_\beta^*, M_\beta}$ to $\mathcal{A}$.*
- **Query Phase II**. *Identical to Query Phase I.*
- **Output**. *Finally, the adversary outputs a bit β' which represents its guess for bit β. If $\beta = \beta'$ 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 $e: \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. 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 admissable.

Our scheme is proven secure under the Decision Linear assumption (DLin), which has been introduced by [5]:

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

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

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

Existing HVE schemes hide a message using a one-time pad construction, i.e. multiplying the message with a session key. This session key is constructed using a secret sharing method over the elements of the encryption-vector, in such a way that not all of the elements are needed for decryption. This automatically leads to a ciphertext that is linear in the length of the vector and a decryption key that is linear in the amount of non-wildcard symbols in the vector.

Our construction works quite different. We also choose a session key based on all the elements of the encryption-vector, but the trapdoor contains the information to cancel out the effect of the symbols at unwanted wildcard-positions. More specifically, we exploit the following polynomial identity that can be evaluated using a bilinear map in Dec:

$$\sum_{i=1}^{l} \prod_{j \in J} (i - j) w_i = \sum_{\substack{i=1 \\ i \notin J}}^{l} \prod_{j \in J} (i - j) w_i, \tag{1}$$

where the set $J \subset \{1, \ldots, l\}$ denotes the position of wildcard symbols, and w_i is the entry of the ciphertext keyword at position i. This identity can be computed using pairings, leading to a ciphertext and decryption key length dependent on $|J|$. However, since this value is not known at the time of encryption, we'll have to replace it by an upper bound.

As an example consider an encryption using the vector $W = (w_1, w_2, w_3)$ and a decryption key using $\bar{W} = (\bar{w}_1, \star, \bar{w}_3)$, i.e. there is a wildcard at position 2. In the Dec we will compute the following in the exponent of the pairing:

$$\sum_{i=1}^{3}(i-2)w_i = (1-2)w_1 + (2-2)w_2 + (3-2)w_3 = (1-2)\bar{w}_1 + (3-2)\bar{w}_3,$$

Since the polynomial $(i-2)$ has a root at 2, the second entry of the ciphertext keyword is canceled out, while the rest will be used in the computation of the session key.

We can construct the polynomial $\prod_{j \in J}(x-j)$ that occurs in (1) by using Viète's formulas. $\prod_{j \in J}(x-j)$ is a polynomial of degree $n = |J|$ defined over an integral domain $\mathbb{Z}_q$ with the roots in J. Then $\prod_{j \in J}(x-j) = x^n + a_{n-1}x^{n-1} + \ldots + a_0$, where each coefficient can be computed according to Viète's formulas:

$$a_{n-k} = (-1)^{i-n} \sum_{1 \le i_1 < i_2 < \ldots < i_k \le n} j_{i_1} j_{i_2} \ldots j_{i_k}, \quad 0 \le k \le n \tag{2}$$

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 $(x-j_1)(x-j_2)(x-j_3)$,

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

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 $l < L$. 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:

- $\mathsf{Setup}(\kappa, \Sigma, L)$: First, choose an upper bound $N \le L$ to the number of wildcard symbols in decryption vectors. Next, given security parameter κ:
  1. Generate a bilinear group $\mathbb{G}$ of a large prime order q and choose a bilinear map $e : \mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{G}_T$.
  2. Pick $L+1$ random elements $V_0, U_1, ..., U_L \in_R \mathbb{G}$.

3. Pick random exponents $\alpha, t_1, t_2, (x_1, \ldots, x_N) \in_R \mathbb{Z}_q$.
4. Let $\Omega_1 = e(g, V_0)^{\alpha t_1}$ and $\Omega_2 = e(g, V_0)^{\alpha t_2}$.
5. Let $V_j = V_0^{x_j}$ for $j = 1, \ldots, N$.

The public parameters are:

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

The master secret key is $msk = \big(\alpha, t_1, t_2, (x_1, \ldots, x_N)\big)$.

– $\mathsf{Extract}(msk, \bar{W})$: Let $\bar{W} = (\bar{w}_1, \ldots, \bar{w}_l) \in \Sigma_\star^l$, where $l \leq L$. Assume that W contains $n \leq N$ wildcards which occur at positions $J = \{j_1, \ldots, j_n\}$. Pick a random $s \in_R \mathbb{Z}_q$ and compute: $s_1 = t_1 + s, s_2 = t_2 + s$. By means of Viète's formulas a_i for $i = 1, \ldots, n$, first compute $m = (\sum_{k=0}^{n} x_k a_k)^{-1}$ and then the decryption key $T_{\bar{W}}$ (where $x_0 = 1$):

$$T_{\bar{W}} = \begin{pmatrix} T_0 = g^{\alpha m s} \\ T_1 = V_0^{s_1} \prod_{i=1}^{l} U_i^{ms \prod_{k=1}^{n}(i-j_k)\bar{w}_i} \\ T_2 = V_0^{\alpha s_2} \prod_{i=1}^{l} U_i^{\alpha ms \prod_{k=1}^{n}(i-j_k)\bar{w}_i} \\ A = \{\alpha m s_2 a_1, \ldots, \alpha m s_2 a_n\} \end{pmatrix}.$$

– $\mathsf{Enc}(param, W, M)$: Let $W = (w_1, \ldots, w_l) \in \Sigma^l$, where $l \leq L$ 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:

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

– $\mathsf{Dec}(S_{W,M}, T_{\bar{W}})$: Given a decryption key $T_{\bar{W}}$ and a ciphertext $S_{W,M}$, first use J to compute Viète's formulas a_i $i = 1, \ldots, n$, 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 $\{j_1, \ldots, j_n\}$. Then

$$\begin{aligned} e(T_0, \prod_{k=0}^{n} C_k^{a_k}) &= e\big(g^{\frac{\alpha s}{\sum_{m=0}^{n} x_m a_m}}, \prod_{k=0}^{n} V_k^{a_k(r_1+r_2)}\big)\ e\big(g^{\frac{\alpha s}{\sum_{m=0}^{n} x_m a_m}}, \prod_{k=0}^{n}\prod_{i=1}^{l} U_i^{i^k a_k w_i (r_1+r_2)}\big) \\ &= \prod_{k=0}^{n} \Big(e(g, V_0)^{\frac{\alpha s (r_1+r_2) x_k a_k}{\sum_{m=0}^{n} x_m a_m}} \prod_{i=1}^{l} e(g, U_i)^{\frac{\alpha s (r_1+r_2) w_i i^k a_k}{\sum_{m=0}^{n} x_m a_m}} \Big) \\ &= e(g, V_0)^{\frac{\alpha s (r_1+r_2) \sum_{k=0}^{n} x_k a_k}{\sum_{m=0}^{n} x_m a_m}} \prod_{i=1}^{l} e(g, U_i)^{\frac{\alpha s (r_1+r_2) w_i \sum_{k=0}^{n} i^k a_k}{\sum_{m=0}^{n} a_m x_m}} \\ &= e(g, V_0)^{\alpha s (r_1+r_2)} \prod_{i=1}^{l} e(g, U_i)^{\frac{\alpha s (r_1+r_2) w_i \prod_{k=1}^{n}(i-j_k)}{\sum_{m=0}^{n} a_m x_m}} \end{aligned} \tag{3}$$

where for (3) we use that $\sum_{k=0}^{n} i^k a_k = \prod_{k=1}^{n}(i - j_k)$.

$$e(T_1, g^{\alpha r_1}) = e(V_0, g)^{\alpha r_1 s_1}\ e\Big(\prod_{i=1}^{l} U_i^{\frac{s\prod_{k=1}^{n}(i-j_k)\bar{w}_i}{\sum_{m=0}^{n} a_m x_m}}, g^{\alpha r_1}\Big)$$

$$= \Omega_1^{r_1}\ e(g, V_0)^{\alpha s r_1}\ \prod_{i=1}^{l} e(g, U_i)^{\frac{\alpha s r_1 \prod_{k=1}^{n}(i-j_k)\bar{w}_i}{\sum_{m=0}^{n} a_m x_m}} \tag{4}$$

$$e(T_2, g^{r_2}) = e(V_0, g)^{\alpha r_2 s_2}\ e\Big(\prod_{i=1}^{l} U_i^{\frac{\alpha s\prod_{k=1}^{n}(i-j_k)\bar{w}_i}{\sum_{m=0}^{n} a_m x_m}}, g^{r_2}\Big)$$

$$= \Omega_2^{r_2}\ e(g, V_0)^{\alpha s r_2}\ \prod_{i=1}^{l} e(g, U_i)^{\frac{\alpha s r_2 \prod_{k=1}^{n}(i-j_k)\bar{w}_i}{\sum_{m=0}^{n} a_m x_m}} \tag{5}$$

$$e(T_{n+1}, g^{\alpha r_1}) e(T_{n+2}, g^{r_2}) = \Omega_1^{r_1}\Omega_2^{r_2}\ e(g, V_0)^{\alpha s(r_1+r_2)} \prod_{i=1}^{l} e(g, U_i)^{\frac{\alpha s(r_1+r_2)\bar{w}_i \prod_{k=1}^{n}(i-j_k)}{\sum_{m=0}^{n} a_m x_m}} \tag{6}$$

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

$$\hat{C}\frac{\prod_{k=0}^{n} e(T_k, C_k)}{e(T_{n+1}, g^{\alpha r_1}) e(T_{n+2}, g^{r_2})} = \frac{M\Omega_1^{r_1}\Omega_2^{r_2}\prod_{k=0}^{n} e(T_k, C_k)}{e(T_{n+1}, g^{\alpha r_1}) e(T_{n+2}, g^{r_2})} = M \tag{7}$$

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 nonnegligible ϵ. 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 $e: \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. 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 $\beta \in_R \{0, 1\}$ the challenger hands the tuple $(g, g^a, g^b, g^{ac}, g^d, Z_\beta)$ to $\mathcal{B}$. Algorithm $\mathcal{B}$'s goal is to guess β 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 $W_0^* \in \Sigma^{l_0}$, $W_1^* \in \Sigma^{l_1}$, where $l_0, l_1 \leq L$, which are different in at least one position. $\mathcal{B}$ flips a coin $\gamma \in \{0, 1\}$. Let $W_\gamma^* = \big(w_1^*, \ldots, w_{l_\gamma}^*\big)$.

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

$$x_j = \frac{\sum_{i=1}^{l} i^j u_i}{\sum_{i=1}^{l} u_i} \quad \text{for } j = 0, \ldots, N$$

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

$$u_i = \begin{cases} g^{\frac{u_i}{w_i^*}} & \text{for } i = 1 \ldots l_\gamma \\ g^{u_i} & \text{for } i = l_\gamma + 1, \ldots, L \end{cases}$$

$\mathcal{B}$ 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 $\Omega_2 = e\big(g^{\sigma_3}(g^a)^{-\sigma_2}, V_0\big)$. The public parameters are:

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

The master secret key is implicitly given by

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

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 $\bar{W} = (\bar{w}_1, \ldots, \bar{w}_l) \in \Sigma_\star^l$, consisting of $l \leq L$ symbols and $n \leq N$ wildcards at positions $J = \{j_1, \ldots, j_n\}$, algorithm $\mathcal{B}$ responds by computing

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

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

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

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

$$T_{\bar{W}} = \Big(T_0, T_1, T_2, J\big). \tag{8}$$

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 whishes to be challenged. $\mathcal{B}$ computes $S_{W_\gamma^*, M_\gamma}$ by first computing

$$\hat{C} = M_\gamma \cdot e\big(g^{ac}, g^b\big)^{(\sigma_1 - 2\sigma_2)v_0} \cdot e\big(g^{ac}, g\big)^{(\sigma_1 - \sigma_2)\sum_{i=0}^{l_\gamma} u_i} \cdot e\big(g^a, g^d\big)^{-\sigma_2 \sum_{i=0}^{l_\gamma} u_i} \cdot e\big(g^b, g^d\big)^{\sigma_3 v_0} \cdot e\big(g^a, Z_\beta\big)^{\sigma_2 v_0} \tag{9}$$

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

$$S_{W_\gamma^*, M_\gamma} = \Big(\hat{C}, \{C_k\}_{k=0}^{N}, \binom{g^{ac}}{g^d}\Big), \tag{10}$$

to $\mathcal{A}$. When $\beta = 0$ this is actually a correct encryption of M_γ under W_γ^* 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 γ'.

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

We will now analyze the probability of success for algorithm $\mathcal{B}$. First, note that if $\beta = 0$, 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 γ. Next note that if $\beta = 1$, then Z_β is random in $\mathbb{G}$ and $S_{W_\gamma^*, M_\gamma}$ is independent from γ, thus $\mathcal{A}$ will have a probability of $\frac{1}{2}$ of guessing γ.

To conclude the proof we have

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

which is nonnegligible, contradicting the Decision Linear Assumption. □

4 Conclusion

We presented a new hidden vector encryption scheme which can work as a wildcard searchable encryption scheme that is a more efficient than existing schemes in some scenarios. The tables below summarize the efficiency of our scheme when compared with 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 for the situations where $n \ll l$ (i.e. the number of wildcards is not large) constructing the decryption key is more efficient than the existing schemes. Moreover, in this situation since N could be small, the ciphertext is 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]	$2l + 2$	$2(l - n) + 1$	$3L + 3$	Arbitrary
Shi, Waters [22]	$l + 4$	$l - n + 3$	$4L + 2$	Arbitrary
Iovino , Persiano [15] Blundo et al. [3]	$2l + 2$	$l - n + 3$	$2L + 4$	Arbitrary
Nishide et al. [18]	$l + 2$	$l + 1$	$3L + 1$	Arbitrary
Our Scheme	$N + 4$	$n + 3$	$L + N + 1$	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: l: 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]	$2(l-n)+1$	Composite order	Arbitrary
Shi, Waters [22]	$(l-n)+3$	Composite order	Arbitrary
Iovino,Persiano [15] and Blundo, Iovino, Persiano [3]	$2(l-n)$	Prime order	Binary
Nishide et al. [18]	$l+1$	Prime order	Binary
Our Scheme	3	Prime order	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: l: 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. Cryptol.*, 21(3):350–391, 2008.
2. Joonsang Baek, Reihaneh Safavi-Naini, and Willy Susilo. Public key encryption with keyword search revisited. In *ICCSA '08: Proceedings of the international conference on Computational Science and Its Applications, Part I*, pages 1249–1259, Berlin, Heidelberg, 2008. Springer-Verlag.
3. Carlo Blundo, Vincenzo Iovino, and Giuseppe Persiano. Private-key hidden vector encryption with key confidentiality. In *CANS '09: Proceedings of the 8th International Conference on Cryptology and Network Security*, pages 259–277, Berlin, Heidelberg, 2009. Springer-Verlag.
4. D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. In C. Cachin and J. Camenisch, editors, *23rd Int. Conf. on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, volume LNCS 3027, pages 506–522, Interlaken, Switzerland, May 2004. Springer.
5. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *CRYPTO*, pages 41–55, 2004.
6. Dan Boneh and Matthew 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. Xavier Boyen and Brent Waters. Anonymous hierarchical identity-based encryption (without random oracles). In *CRYPTO*, pages 290–307, 2006.
10. Yan cheng Chang and Michael Mitzenmacher. Privacy preserving keyword searches on remote encrypted data. In *In Proc. of 3rd Applied Cryptography and Network Security Conference (ACNS*, pages 442–455, 2005.
11. Reza Curtmola, Juan Garay, Seny Kamara, and Rafail Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. In *CCS '06: Proceedings of the 13th ACM conference on Computer and communications security*, pages 79–88, New York, NY, USA, 2006. ACM.
12. 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.

13. Eu-Jin Goh. Secure indexes. Cryptology ePrint Archive, Report 2003/216, 2003. http://eprint.iacr.org/2003/216/.
14. Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM Conference on Computer and Communications Security*, pages 89–98. ACM, 2006.
15. Vincenzo Iovino and Giuseppe Persiano. Hidden-vector encryption with groups of prime order. In *Pairing '08: Proceedings of the 2nd international conference on Pairing-Based Cryptography*, pages 75–88, Berlin, Heidelberg, 2008. Springer-Verlag.
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 encryptor-specified access structures. In Steven M. Bellovin, Rosario Gennaro, Angelos D. Keromytis, and Moti Yung, editors, *ACNS*, volume 5037 of *Lecture Notes in Computer Science*, pages 111–129, 2008.
19. Hyun Sook Rhee, Jong Hwan Park, Willy Susilo, and Dong Hoon Lee. Improved searchable public key encryption with designated tester. In *ASIACCS '09: Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, pages 376–379, New York, NY, USA, 2009. ACM.
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, T-H. Hubert Chan, Dawn Song, and Adrian Perrig. Multi-dimensional range query over encrypted data. In *SP '07: Proceedings of the 2007 IEEE Symposium on Security and Privacy*, pages 350–364, Washington, DC, USA, 2007. IEEE Computer Society.
22. Elaine Shi and Brent Waters. Delegating capabilities in predicate encryption systems. In *ICALP '08: Proceedings of the 35th international colloquium on Automata, Languages and Programming, Part II*, pages 560–578, Berlin, Heidelberg, 2008. Springer-Verlag.
23. Dawn Xiaodong Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In *SP '00: Proceedings of the 2000 IEEE Symposium on Security and Privacy*, page 44, Washington, DC, USA, 2000. IEEE Computer Society.
24. Brent R. Waters, Dirk Balfanz, Glenn Durfee, and D. K. Smetters. Building an encrypted and searchable audit log. In *Proceedings of Network and Distributed System Security Symposium 2004 (NDSS'04)*, San Diego, CA, February 2004.