Private-Key Hidden Vector Encryption with Key Confidentiality

Carlo Blundo [*] Vincenzo Iovino [†] Giuseppe Persiano [‡]

September 4, 2009

Abstract

Predicate encryption is an important cryptographic primitive that has been recently studied [BDOP04, BW07, GPSW06, KSW08] and that has found wide applications. Roughly speaking, in a predicate encryption scheme the owner of the master secret key K can derive secret key $\tilde{K}$, for any *pattern* vector $\vec{k}$. In encrypting a message M, the sender can specify an *attribute* vector $\vec{x}$ and the resulting ciphertext $\tilde{X}$ can be decrypted only by using keys $\tilde{K}$ such that $P(\vec{x}, \vec{k}) = 1$, for a fixed predicate P. A predicate encryption scheme thus gives the owner of the master secret key fine-grained control on which ciphertexts can be decrypted and this allows him to delegate the decryption of different types of messages (as specified by the attribute vector) to different entities.

In this paper, we give a construction for *hidden vector encryption* which is a special case of predicate encryption schemes introduced by [BW07]. Here the ciphertext attributes are vectors $\vec{x} = \langle x_1, \ldots, x_\ell \rangle$ over alphabet Σ, key patterns are vectors $\vec{k} = \langle k_1, \ldots, k_\ell \rangle$ over alphabet $\Sigma \cup \{\star\}$ and we consider the $\mathsf{Match}(\vec{x}, \vec{k})$ predicate which is true if and only if $k_i \neq \star$ implies $x_i = k_i$. Besides guaranteeing the security of the attributes of a ciphertext, our construction also gives security guarantees for the key patterns. We stress that security guarantees for key patterns only make sense in a private-key setting and have been recently considered by [SSW09] which gave a construction in the symmetric bilinear setting with groups of composite (product of four primes) order. In contrast, our construction uses *asymmetric* bilinear groups of *prime* order and the length of the key is equal to the weight of the pattern, thus resulting in an increased efficiency. We remark that our construction is based on falsifiable (in the sense of [BW06, Nao03]) complexity assumptions for the asymmetric bilinear setting and are proved secure in the standard model (that is, without random oracles).

Keywords: private-key predicate encryption, key confidentiality.

1 Introduction

Predicate encryption is an important cryptographic primitive that has been recently studied [BDOP04, BW07, GPSW06, KSW08] and that has found wide applications. Roughly speaking, in a predicate encryption scheme the owner of the master secret key SK, can derive secret key $\tilde{K}$, for any pattern vectors $\vec{k}$. Similarly, in encrypting a message M, the sender can specify an attribute vector $\vec{x}$ and the resulting ciphertext $\tilde{X}$ can be decrypted only by using keys $\tilde{K}$ such that $P(\vec{x}, \vec{k}) = 1$, for a fixed predicate P.

In this paper, we consider *hidden vector encryption* that is a special class of predicate encryptions first studied in [BW07]. In a hidden vector encryption scheme, ciphertexts are associated with *attribute vectors* $\vec{x}$ of length ℓ over an alphabet Σ and keys are associated with *pattern vectors* $\vec{k}$ of length ℓ over the alphabet $\Sigma \cup \{\star\}$. The predicate we are interested in is the Match predicate defined as follows: $\mathsf{Match}(\vec{x}, \vec{k}) = 1$ if and only if for $i = 1, \ldots, \ell$ either $k_i = \star$ or $k_i = x_i$. Constructions for hidden vector encryption have been given in [BW07] (based on hardness assumptions in groups of composite order) and in [IP08] (based on hardness assumptions in groups of prime order).

[*]Dipartimento di Informatica ed Applicazioni, Università di Salerno, 84084 Fisciano (SA), Italy. `carblu@dia.unisa.it`.
[†]Dipartimento di Informatica ed Applicazioni, Università di Salerno, 84084 Fisciano (SA), Italy. `iovino@dia.unisa.it`.
[‡]Dipartimento di Informatica ed Applicazioni, Università di Salerno, 84084 Fisciano (SA), Italy. `giuper@dia.unisa.it`.

Until now research has concentrated on guaranteeing the security of the ciphertext with respect to the cleartext and to the attribute vector and not much attention has been devoted to the security of the key. Specifically, one would like a key not to reveal the associated pattern. This is particularly important in some applications in which a user generates the key for a certain pattern and gives it to a third party to perform some operations. Knowledge of the pattern associated with the key might reveal some information about the operation being performed. Obviously, this is impossible to achieve in a public-key setting. Indeed an adversary $\mathcal{A}$ holding a key $\tilde{K}$ associated to a secret pattern $\vec{k}$ can simply produce a ciphertext $\tilde{X}$ with attribute $\vec{x}$ and then try to decrypt $\tilde{X}$ using $\tilde{K}$. If $\mathcal{A}$ succeeds in decrypting $\tilde{K}$ then $\mathcal{A}$ knows that $P(\vec{x}, \vec{k}) = 1$. This attack does not hold in the private key setting as $\mathcal{A}$ cannot produce ciphertext $\tilde{X}$. Simply keeping the public key secret from the adversary does not seem to work for previous predicate encryption schemes (see, for example [BW07, KSW08]) and the problem seems to call for a new construction. The scheme of [SSW09] is constructed modifying the previous scheme of [KSW08], likewise, we build our scheme from the scheme of [IP08].

Prior work and our contribution. Shen, Shi and Waters [SSW09] were the first to consider key confidentiality in the context of predicate encryption and they provided a construction for the inner-product predicate (that is, a key can decrypt a ciphertext if and only if the pattern vector of the key is orthogonal to the attribute vector of the ciphertext). In this paper we present a construction for an *hidden vector encryption scheme* which, besides guaranteeing privacy of the attribute vector of ciphertext, guarantees that keys do not leak any information on the associated pattern, *besides the location of the $\star$'s.* We stress that the construction of [SSW09] for the inner-product predicate implies (with a small loss of efficiency) a construction also for hidden vector encryption scheme. The security of the construction of [SSW09] is based on bilinear assumptions on groups of order product of *four* primes, and thus, it is less efficient. In our construction we show that, by slightly relaxing the notion of key confidentiality, we can obtain construction using asymmetric bilinear groups of prime order (which results in much more efficient constructions). We remark that our construction is based on falsifiable (in the sense of [BW06, Nao03]) complexity assumptions for the *asymmetric* bilinear setting for groups of prime order and are proved secure in the standard model (that is, without random oracles).

Moving from composite order groups to prime order groups, besides giving very efficient constructions, is also important since assumptions based on prime order groups are considered weaker than the corresponding assumptions that intertwine and compound potential vulnerabilities from factoring and pairings (see the discussion in [Boy08]).

Finally, we stress that the only previous construction of hidden vector encryption schemes based on prime order groups of [IP08] does not give any security guarantee for the key.

2 Hidden Vector Encryption Schemes

In this paper we consider a special type of predicate encryption schemes called *Hidden Vector Encryption Scheme*, (an HVE scheme, in short). We present the definition and the construction for $\Sigma = \{0, 1\}$. In Section 8 we briefly explain how the constructions can be extended to larger alphabets.

An HVE scheme consists of four algorithms:

1. $\mathsf{MasterKeyGen}(1^n, 1^\ell)$: Given security parameter n, and number of attributes $\ell = \mathsf{poly}(n)$, procedure $\mathsf{MasterKeyGen}$ outputs the private key SK.
2. $\mathsf{Enc}(\mathsf{SK}, \vec{x})$: Given attribute vector $\vec{x} \in \{0, 1\}^\ell$ and secret key SK, procedure Enc outputs an encrypted attribute vector $\tilde{X}$.
3. $\mathsf{KeyGen}(\mathsf{SK}, \vec{k})$: Given private key SK, a pattern vector $\vec{k}$ of length ℓ over the alphabet $\{0, 1, \star\}$, procedure KeyGen outputs a *key* $\tilde{K}$ for the $\vec{k}$.
4. $\mathsf{Test}(\tilde{X}, \tilde{K})$: given encrypted attribute vector $\tilde{X}$ and key $\tilde{K}$ corresponding to pattern $\vec{k}$, procedure Test returns $\mathsf{Match}(\vec{x}, \vec{k})$ except with negligible probability.

We state security in the selective attribute model using the following experiments.

2.1 Semantic security

The first experiment considers an adversary that tries to learn information from an encryption. We model this using an indistinguishability experiment in which the adversary $\mathcal{A}$ selects two challenge attribute vectors $\vec{z}_0$ and $\vec{z}_1$ and receives an encrypted attribute vector corresponding to a randomly chosen challenge attribute vector. We allow the adversary to issue key queries for patterns $\vec{y}$ that match neither of $\vec{z}_0$ and $\vec{z}_1$ and to see encryption of attribute vectors of his choice (see Section 7 for a stronger notion). Following is the description of experiment $\mathsf{SemanticExp}_{\mathcal{A}}$.

$\mathsf{SemanticExp}_{\mathcal{A}}(1^n, 1^\ell)$

1. Initialization Phase. The adversary $\mathcal{A}$ announces two challenge attribute vectors $\vec{z}_0, \vec{z}_1 \in \{0,1\}^\ell$.
2. Key-Generation Phase. The secret key SK is generated by the $\mathsf{MasterKeyGen}$ procedure.
3. Query Phase I. $\mathcal{A}$ can make any number of key and encryption query.
   A key query for pattern $\vec{k}$ is answered as follows. If $\mathsf{Match}(\vec{z}_0, \vec{k}) = 0$ and $\mathsf{Match}(\vec{z}_1, \vec{k}) = 0$ then $\mathcal{A}$ receives the output of $\mathsf{KeyGen}(\mathsf{SK}, \vec{k})$. Otherwise, $\mathcal{A}$ receives $\perp$. An encryption query for attribute vectors $\vec{x}$ is answered by returning $\mathsf{Enc}(\mathsf{SK}, \vec{x})$.
4. Challenge construction. η is chosen at random from $\{0,1\}$ and $\mathcal{A}$ is given $\mathsf{Enc}(\mathsf{SK}, \vec{z}_\eta)$.
5. Query Phase II. Identical to Query Phase I.
6. Output Phase. $\mathcal{A}$ returns η'.
   If $\eta = \eta'$ then the experiments returns 1 else 0.

Definition 1 *An* HVE *scheme* (MasterKeyGen, Enc, KeyGen, Test) *is* semantically secure*, if for all probabilistic poly-time adversaries* $\mathcal{A}$

$$\left|\mathrm{Prob}[\mathsf{SemanticExp}_{\mathcal{A}}(1^n, 1^\ell) = 1] - 1/2\right|$$

is negligible in n for all $\ell = \mathsf{poly}(n)$.

2.2 Key confidentiality

In this section we present our definition for key confidentiality. We model this property by using an indistinguishability experiment in which the adversary $\mathcal{A}$ outputs two challenge patterns $\vec{k}_0$ and $\vec{k}_1$ of his choice. $\mathcal{A}$ is then allowed to issue encryption queries for vectors $\vec{x}$ that match neither of $\vec{k}_0$ and $\vec{k}_1$ and key queries for patterns $\vec{k}$ of his choice. At the end $\mathcal{A}$ is presented with the key associated with a randomly chosen challenge pattern. In our notion of key confidentiality, the adversary is limited to challenges on patterns in which the "don't care" entries (that is, $\star$) are in the same positions.

$\mathsf{KeyExp}_{\mathcal{A}}(1^n, 1^\ell)$

1. Initialization Phase. The adversary $\mathcal{A}$ announces two challenge patterns $\vec{k}_0, \vec{k}_1 \in \{0,1,\star\}^\ell$. If the set of positions for which $\vec{k}_0$ and $\vec{k}_1$ have $\star$ differ then the experiment returns 0.
2. Key-Generation Phase. The secret key SK is generated by the $\mathsf{MasterKeyGen}$ procedure.
3. Query Phase I. $\mathcal{A}$ can make any number of key and encryption query.
   A key query for pattern $\vec{k}$ is answered by returning $\mathsf{KeyGen}(\mathsf{SK}, \vec{k})$.
   An encryption query for attribute vector $\vec{x}$ is answered as follows.
   If $\mathsf{Match}(\vec{x}, \vec{k}_0) = \mathsf{Match}(\vec{x}, \vec{k}_1) = 0$ then $\mathcal{A}$ receives $\mathsf{Enc}(\mathsf{SK}, \vec{x})$. Otherwise, $\mathcal{A}$ receives $\perp$.
4. Challenge construction. η is chosen at random from $\{0,1\}$ and receives $\mathsf{KeyGen}(\mathsf{SK}, \vec{k}_\eta)$.
5. Query Phase II. Identical to Query Phase I.
6. Output Phase. $\mathcal{A}$ returns η'.
   If $\eta = \eta'$ then the experiments returns 1 else 0.

Definition 2 *A predicate encryption scheme* (MasterKeyGen, Enc, KeyGen, Test) *is* key secure *if for all probabilistic poly-time adversaries* $\mathcal{A}$*,*

$$\left|\mathrm{Prob}[\mathsf{KeyExp}_{\mathcal{A}}(1^n, 1^\ell) = 1] - 1/2\right|$$

is negligible in n for all $\ell = \mathsf{poly}(n)$.

2.3 Secure HVE

Finally we have,

Definition 3 *An* HVE*scheme* (MasterKeyGen, Enc, KeyGen, Test) *is* secure *if it is both semantically secure and key secure.*

Remark on the notion of key confidentiality. In our notion of key confidentiality the key might reveal the position of the $\star$'s in the associated pattern, since no requirement is made for adversary choosing challenge patterns with $\star$'s in different positions. In some applications, this might not be a drawback. For example, predicate encryption can be used for performing searches on encrypted data. For example, a user interested in selecting ciphertexts for which *Name=Alex* and *Sex=M* gets a key corresponding to a pattern that has $\star$ in all positions other than *Name* and *Sex*. An eavesdropper learns that the user is searching the fields *Name* and *Sex* but no information is given on the name the user is searching for and whether the user is searching for a male or a female. We remark that the construction of [SSW09] hides all information of the key, but their construction is less efficient than ours since it uses groups of composite order of *four* primes. Roughly speaking, by slightly relaxing the security notion, we manage to build a more efficient scheme.

3 Complexity Assumptions

We work in asymmetric prime order bilinear groups of 'Type 3' (see [Boy08]). Specifically, we have cyclic multiplicative groups $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ of order p such that there exists no efficiently computable morphism from $\mathbb{G}_1$ to $\mathbb{G}_2$ or from $\mathbb{G}_2$ to $\mathbb{G}_1$. In addition we have a non-degenerate pairing function $\mathsf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$; that is, for all $x \in \mathbb{G}_1, y \in \mathbb{G}_2$, $x \neq 1$ or $y \neq 1$, we have $\mathsf{e}(x, y) \neq 1$ and for all $a, b \in \mathbb{Z}_p$ we have $\mathsf{e}(x^a, y^b) = \mathsf{e}(x, y)^{ab}$. We denote by g_1, g_2, and $\mathsf{e}(g_1, g_2)$ generators of $\mathbb{G}_1, \mathbb{G}_2$, and $\mathbb{G}_T$, respectively.

We call a tuple $\mathcal{I} = [p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, \mathsf{e}]$ an *asymmetric bilinear* instance and assume that there exists an efficient generation procedure $\mathcal{G}$ that, on input security parameter 1^n, outputs an instance with $|p| = \Theta(n)$.

We now present a new assumption, which we call the (d, m)-Q Assumption, on which we base the proof of key security of our construction. Semantic security is based instead on the Decision Linear Assumption and on the Bilinear Decision Diffie-Hellman Assumption which we review in Section 3. We present the assumption in the form of a game between a challenger Ch and a distinguisher $\mathcal{D}$ on input the security parameter n.

Game (d, m)**-**$Q(1^n)$

1. The challenger Ch picks a random asymmetric bilinear instance $\mathcal{I} = [p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, \mathsf{e}]$ by running generator $\mathcal{G}$ on input security parameter 1^n and sets $\mathsf{ChOutput} = \emptyset$.
2. For $i = 1, \ldots, d$ and $b = 0, 1$, Ch chooses random $\hat{t}_{i,b}, \hat{v}_{i,b} \in \mathbb{Z}_p$.
3. For $i = 1, \ldots, d$, Ch chooses random $\hat{a}_i \in \mathbb{Z}_p$ such that their sum is equal to 0.
4. Define set of pairs $JH = \{(j, h) | 1 \leq j \leq m,\ 1 \leq h \leq m,\ j \neq h \text{ or } j = h,\ m + 1 \leq j \leq d\}$.
   For $(j, h) \in JH$, Ch chooses a random $\hat{s}_{(j,h)} \in \mathbb{Z}_p$ and computes matrices $\mathsf{A_{j,h}}$ and $\mathsf{B_{j,h}}$ as follows, where $\times$ denotes a missing entry in the matrices:[1]

   $\mathsf{A_{j,h}} =$

$$\begin{cases} \begin{bmatrix} g_1^{\hat{s}_{j,h}\hat{t}_{1,0}}, & \ldots, & \times, & \ldots, & g_1^{\hat{s}_{j,h}\hat{t}_{h,0}}, & \ldots, & g_1^{\hat{s}_{j,h}\hat{t}_{d,0}} \\ g_1^{\hat{s}_{j,h}\hat{t}_{1,1}}, & \ldots, & g_1^{\hat{s}_{j,h}\hat{t}_{j,1}}, & \ldots, & \times & \ldots, & g_1^{\hat{s}_{j,h}\hat{t}_{d,1}} \end{bmatrix} & \text{if } j \neq h \text{ and } j, h \leq m \\ \\ \begin{bmatrix} g_1^{\hat{s}_{j,h}\hat{t}_{1,0}}, & \ldots, & \times, & \ldots, & g_1^{\hat{s}_{j,h}\hat{t}_{d,0}} \\ g_1^{\hat{s}_{j,h}\hat{t}_{1,1}}, & \ldots, & g_1^{\hat{s}_{j,h}\hat{t}_{j,1}}, & \ldots, & g_1^{\hat{s}_{j,h}\hat{t}_{d,1}} \end{bmatrix} & \text{if } j = h \text{ and } j > m \end{cases}$$

[1]For the sake of simplicity of exposition, in the definition we have implicitly assumed that $j \leq h$.

and $\mathsf{B_{j,h}} =$

$$\begin{cases} \begin{bmatrix} g_1^{\hat{s}_{j,h}\hat{v}_{1,0}}, & \ldots, & \times, & \ldots, & g_1^{\hat{s}_{j,h}\hat{v}_{h,0}}, & \ldots, & g_1^{\hat{s}_{j,h}\hat{v}_{d,0}} \\ g_1^{\hat{s}_{j,h}\hat{v}_{1,1}}, & \ldots, & g_1^{\hat{s}_{j,h}\hat{v}_{j,1}}, & \ldots, & \times & \ldots, & g_1^{\hat{s}_{j,h}\hat{v}_{d,1}} \end{bmatrix} & \text{if } j \neq h \text{ and } j,h \leq m \\ \begin{bmatrix} g_1^{\hat{s}_{j,h}\hat{v}_{1,0}}, & \ldots, & \times, & \ldots, & g_1^{\hat{s}_{j,h}\hat{v}_{d,0}} \\ g_1^{\hat{s}_{j,h}\hat{v}_{1,1}}, & \ldots, & g_1^{\hat{s}_{j,h}\hat{v}_{j,1}}, & \ldots, & g_1^{\hat{s}_{j,h}\hat{v}_{d,1}} \end{bmatrix} & \text{if } j = h \text{ and } j > m \end{cases}$$

Ch appends the above matrices to $\mathsf{ChOutput}$.

5. For $i = 1, \ldots, d$ and $b = 0, 1$, Ch computes and appends to $\mathsf{ChOutput}$

$$C_{i,b} = g_2^{1/\hat{t}_{i,b}} \quad \text{and} \quad D_{i,b} = g_2^{1/\hat{v}_{i,b}}.$$

6. Ch chooses random $\eta \in \{0, 1\}$ and let $\vec{z} = \langle z_1, \ldots, z_d \rangle = \eta^m \cdot 0^{d-m}$.
For $i = 1, \ldots, d$, Ch computes

$$E_i = C_{i,z_i}^{\hat{a}_i} \quad \text{and} \quad F_i = D_{i,z_i}^{\hat{a}_i}$$

and appends the values E_i and F_i to $\mathsf{ChOutput}$.

7. Challenger Ch runs $\mathcal{D}$ on input sequence $\mathsf{ChOutput}$ and receives output η'.

We define the advantage $\mathsf{Adv}_{\mathcal{D}}(n, d, m)$ of distinguisher $\mathcal{D}$ in the Game (d, m)-$Q(1^n)$ as

$$\mathsf{Adv}_{\mathcal{D}}(n, d, m) = \left| \text{Prob}[\eta = \eta'] - \frac{1}{2} \right|.$$

We are now ready to formally state Assumption (d, m)-Q.

Assumption 1 (Assumption (d, m)-Q) *For all probabilistic poly-time distinguishers $\mathcal{D}$, we have that $\mathsf{Adv}_{\mathcal{D}}(n, d, m)$ is negligible in n, for $d = \mathsf{poly}(n)$, and $1 \leq m \leq d$.*

The (d, m)-Q Assumption can be justified by extending the framework of the Uber-Assumption [BBG05, Boy08] to rational functions along the lines of [Boy08]. In the rest of this section we review other hardness assumptions used in the paper.

Bilinear Decision Diffie-Hellman Given a tuple $[g_1, g_2, g_1^a, g_1^b, g_2^a, g_2^b, g_1^c, Z]$ for random exponents $a, b, c \in \mathbb{Z}_p$ it is hard to distinguish between $Z = \mathsf{e}(g_1, g_2)^{abc}$ and a random Z from $\mathbb{G}_T$. More specifically, for an algorithm $\mathcal{A}$ we define experiment $\mathsf{BDDHExp}_{\mathcal{A}}$ as follows.

```
$\mathsf{BDDHExp}_{\mathcal{A}}(1^n)$
1.  Choose instance $\mathcal{I} = [p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, \mathsf{e}]$ with security parameter $1^n$.
2.  Choose $a, b, c \in \mathbb{Z}_p$ at random.
3.  Choose $\eta \in \{0, 1\}$ at random.
4.  If $\eta = 1$ then choose $z \in \mathbb{Z}_p$ at random; else, set $z = abc$.
5.  Set $A = g_1^a, B = g_1^b, \hat{A} = g_2^a, \hat{B} = g_2^b, C = g_1^c$ and $Z = \mathsf{e}(g_1, g_2)^z$.
6.  Let $\eta' = \mathcal{A}(\mathcal{I}, A, B, \hat{A}, \hat{B}, C, Z)$.
7.  If $\eta = \eta'$ then return 1 else return 0.
```

Assumption 2 (Bilinear Decisional Diffie-Hellman (BDDH)) *For all probabilistic poly-time algorithms $\mathcal{A}$, $|\text{Prob}[\mathsf{BDDHExp}_{\mathcal{A}}(1^n) = 1] - 1/2|$ is negligible in n.*

Decision Linear Given a tuple $[g_1, g_2, g_1^{z_1}, g_1^{z_2}, g_2^{z_1}, g_2^{z_2}, g_1^{z_1 z_3}, g_1^{s}, Z]$ for random exponents $z_1, z_2, z_3, s \in \mathbb{Z}_p$ it is hard to distinguish between $Z = g_1^{z_2(s-z_3)}$ and a random Z from $\mathbb{G}_1$. More specifically, for an algorithm $\mathcal{A}$ we define experiment $\mathsf{DLExp}_{\mathcal{A}}$ as follows.

```
$\mathsf{DLExp}_{\mathcal{A}}(1^n)$
1.  Choose instance $\mathcal{I} = [p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, \mathsf{e}]$ with security parameter $1^n$.
2.  Choose $u_1, u_2, u_3, u \in \mathbb{Z}_p$ at random.
3.  Choose $\eta \in \{0, 1\}$ at random.
4.  If $\eta = 1$ then choose $z \in \mathbb{Z}_p$ at random; else, set $z = u_2(u - u_3)$.
5.  Set $U_1 = g_1^{u_1}, U_2 = g_1^{u_2}, \hat{U}_1 = g_2^{u_1}, \hat{U}_2 = g_2^{u_2}, U_{13} = g_1^{u_1 u_3}, U = g_1^{u}$, and $Z = g_1^{z}$.
6.  Let $\eta' = \mathcal{A}(\mathcal{I}, U_1, U_2, \hat{U}_1, \hat{U}_2, U_{13}, U, Z)$.
7.  If $\eta = \eta'$ then return 1 else return 0.
```

Assumption 3 (Decision Linear (DLinear)) *For all probabilistic poly-time algorithms $\mathcal{A}$,*

$$|\text{Prob}[\mathsf{DLExp}_{\mathcal{A}}(1^n) = 1] - 1/2|$$

is negligible in n.

Note that Decision Linear implies Decision BDDH and the Decision Linear assumption has been used in [BW06].

4 The basic scheme

In this section, we describe our proposal for a secure HVE.

The MasterKeyGen procedure. On input security parameter 1^n and the number of attributes $\ell = \mathsf{poly}(n)$, MasterKeyGen proceeds as follows.

```
1. Select an asymmetric bilinear instance $\mathcal{I} = [p, q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, \mathsf{e}]$ with $|N| = \Theta(n)$ by running $\mathcal{G}$.
2. Pick $y$ at random in $\mathbb{Z}_p$ and set $Y = \mathsf{e}(g_1, g_2)^y$.
   For $i = 1, \ldots, \ell$,
      Choose $t_{i,0}, t_{i,1}, v_{i,0}, v_{i,1}$ at random from $\mathbb{Z}_p$.
      Set
         $T_{i,0} = g_1^{t_{i,0}}, \quad T_{i,1} = g_1^{t_{i,1}}, \quad V_{i,0} = g_1^{v_{i,0}}, \quad V_{i,1} = g_1^{v_{i,1}},$
         $\bar{T}_{i,0} = g_2^{1/t_{i,0}}, \quad \bar{T}_{i,1} = g_2^{1/t_{i,1}}, \quad \bar{V}_{i,0} = g_2^{1/v_{i,0}}, \quad \bar{V}_{i,1} = g_2^{1/v_{i,1}}.$
      Set $\mathsf{SK}_i = (T_{i,0}, T_{i,1}, V_{i,0}, V_{i,1}, \bar{T}_{i,0}, \bar{T}_{i,1} \bar{V}_{i,0}, \bar{V}_{i,1}, )$.
3. Return $\mathsf{SK} = (\mathcal{I}, Y, y, \mathsf{SK}_1, \ldots, \mathsf{SK}_\ell)$.
```

The Enc procedure. On input secret key SK and attribute vector $\vec{x}$ of length ℓ, Enc proceeds as follows.

```
1. Pick $s$ at random from $\mathbb{Z}_p$ and set $\Omega = Y^{-s}$.
2. For $i = 1, \ldots, \ell$,
      pick $s_i$ at random from $\mathbb{Z}_p$.
      set $X_i = T_{i,x_i}^{s-s_i}$ and $Z_i = V_{i,x_i}^{s_i}$.
3. Return encrypted attribute vector $\tilde{X} = (\Omega, (X_i, Z_i)_{i=1}^{\ell})$.
```

In the following sometimes will use the writing $\mathsf{Enc}(\mathsf{SK}, \vec{x}; s, s_1, \ldots, s_\ell)$ to denote the encrypted attribute vector $\tilde{X}$ output by Enc on input SK and $\vec{x}$ when using $s, s_1, \ldots, s_\ell$ as random elements.

The KeyGen procedure. On input secret key SK and pattern vector $\vec{k}$, KeyGen proceeds as follows.

1. Let $S_{\vec{k}}$ be the set of positions in which $k_i \neq \star$.
2. Choose $(a_i)_{i \in S_{\vec{k}}}$ at random in $\mathbb{Z}_p$ under the constraint that their sum is y.
3. For $i \in S_{\vec{k}}$, set $R_i = \bar{T}_{i,k_i}^{a_i}$ and $W_i = \bar{V}_{i,k_i}^{a_i}$.
4. Return $\tilde{K} = (i, R_i, W_i)_{i \in S_{\vec{k}}}$.

In the following sometimes will use the writing $\mathsf{KeyGen}(\mathsf{SK}, \vec{k}; (a_i)_{i \in S_{\vec{k}}})$ to denote the key $\tilde{K}$ computed by KeyGen on input SK and $\vec{k}$ and using $(a_i)_{i \in S_{\vec{k}}}$ as random elements.

The Test procedure. On input an encrypted attribute vector $\tilde{X} = (\Omega, (X_i, Z_i)_{i=1}^{\ell})$ and a key $\tilde{K} = ((i_1, R_{i_1}, W_{i_1}), \ldots, (i_m, R_{i_m}, W_{i_m}))$, Test proceeds as follows.

1. Compute $a = \Omega \cdot \prod_{j=1}^{m} \mathsf{e}(X_{i_j}, R_{i_j}) \mathsf{e}(Z_{i_j}, W_{i_j})$.
2. If $a = 1$ then return TRUE else return FALSE.

We next prove that the quadruple is indeed a predicate encryption scheme.

Theorem 1 *The quadruple of algorithms* (MasterKeyGen, Enc, KeyGen, Test) *specified above is a predicate encryption scheme.*

Proof. It is sufficient to verify that the procedure Test returns 1 when $\mathsf{Match}(\vec{x}, \vec{k}) = 1$. Let $\tilde{X} = (\Omega, (X_i, Z_i)_{i=1}^{\ell})$ be the output of $\mathsf{Enc}(\mathsf{SK}, \vec{x}; s, s_1, \ldots, s_\ell)$ and let $\tilde{K} = (i, R_i, W_i)_{i \in S_{\vec{k}}}$ be the output of $\mathsf{KeyGen}(\mathsf{SK}, \vec{k}; (a_i)_{i \in S_{\vec{k}}})$. Then we have

$\mathsf{Test}(\tilde{X}, \tilde{K})$

$$\begin{aligned}
&= \Omega \cdot \prod_{i \in S_{\vec{k}}} \mathsf{e}(X_i, R_i) \cdot \mathsf{e}(Z_i, W_i) \\
&= \mathsf{e}(g_1, g_2)^{-ys} \cdot \prod_{i \in S_{\vec{k}}} \mathsf{e}(T_{i,x_i}^{s-s_i}, \bar{T}_{i,k_i}^{a_i}) \cdot \mathsf{e}(V_{i,x_i}^{s_i}, \bar{V}_{i,k_i}^{a_i}) \text{ (since } x_i = k_i \text{ for } \ i \in S_{\vec{k}}) \\
&= \mathsf{e}(g_1, g_2)^{-ys} \cdot \prod_{i \in S_{\vec{k}}} \mathsf{e}(T_{i,k_i}^{s-s_i}, \bar{T}_{i,k_i}^{a_i}) \cdot \mathsf{e}(V_{i,k_i}^{s_i}, \bar{V}_{i,k_i}^{a_i}) \\
&\quad \text{(since } \mathsf{e}(T_{i,k_i}, \bar{T}_{i,k_i}) = \mathsf{e}(V_{i,k_i}, \bar{V}_{i,k_i}) = \mathsf{e}(g_1, g_2) \in \mathbb{G}_T) \\
&= \mathsf{e}(g_1, g_2)^{-ys} \cdot \prod_{i \in S_{\vec{k}}} \mathsf{e}(g_1, g_2)^{(s-s_i)a_i} \cdot \mathsf{e}(g_1, g_2)^{s_i a_i} \\
&= \mathsf{e}(g_1, g_2)^{-ys} \cdot \prod_{i \in S_{\vec{k}}} \mathsf{e}(g_1, g_2)^{s a_i} \text{ (since } \sum_{i \in S_{\vec{k}}} a_i = y) \\
&= \mathsf{e}(g_1, g_2)^{-ys} \cdot \mathsf{e}(g_1, g_2)^{ys} = 1.
\end{aligned}$$

□

5 Proof of Semantic Security

In this section, we prove that the scheme presented in Section 4 is semantically secure. Consider the following experiments, for $j = 0, \cdots, \ell$.

$\mathsf{SemanticExp}_{\mathcal{A}}(1^n, 1^\ell, \vec{z}, j)$

1. Key-generation Phase. Compute $\mathsf{SK} = (\mathcal{I}, y, \mathsf{SK}_1, \cdots, \mathsf{SK}_\ell)$ by executing $\mathsf{MasterKeyGen}(1^n, 1^\ell)$.
2. Query Phase I. Answer Enc queries for attribute vectors $\vec{x}$ by using secret key SK.
   Answer KeyGen queries for pattern vectors $\vec{k}$ such that $\mathsf{Match}(\vec{z}, \vec{k}) = 0$ using secret key SK.
3. Challenge Construction.
   1. If $j = 0$ set $\Omega = \mathsf{e}(g_1, g_2)^{-ys}$.

2. If $j \geq 1$ choose Ω uniformly at random from $\mathbb{G}_T$.
3. For $i = 1, \dots, j-1$,
   choose X_i and Z_i uniformly at random in $\mathbb{G}_1$.
4. If $j = 0$ set $\alpha = 1$ else set $\alpha = j$.
5. For $i = \alpha, \dots, \ell$,
   choose s_i uniformly at random in $\mathbb{Z}_p$ and set $X_i = g_1^{t_{i,z_i}(s-s_i)}$ and $Z_i = g_1^{s_i v_{i,z_i}}$.
6. Set $\tilde{X} = (\Omega, (X_i, Z_i)_{i=1}^{\ell})$.

7. Query Phase II. Identical to Query Phase I.
8. **return:** $\mathcal{A}(\tilde{X})$.

We will use the writing $\mathsf{SemanticExp}_{\mathcal{A}}(1^n, 1^\ell, \vec{z}, j; s, s_\alpha, \dots, s_\ell)$ to denote the tuple $\tilde{X}$ computed by $\mathsf{SemanticExp}_{\mathcal{A}}(1^n, 1^\ell, \vec{z}, j)$ using $s, s_\alpha, \dots, s_\ell$ as random values, where $\alpha = 1$ for $j = 0$ and $\alpha = j$ for $j > 0$.

We will denote by $p_j^{\mathcal{A}}(\vec{z})$ the probability that experiment $\mathsf{SemanticExp}_{\mathcal{A}}(1^n, 1^\ell, \vec{z}, j)$ returns 1. Notice that in $\mathsf{SemanticExp}_{\mathcal{A}}(1^n, 1^\ell, \vec{z}, 0)$ adversary $\mathcal{A}$ receives a valid encrypted attribute vector $\tilde{X}$ for attribute vector $\vec{z}$ and secret key SK whereas in $\mathsf{SemanticExp}_{\mathcal{A}}(1^n, 1^\ell, \vec{z}, \ell)$ adversary $\mathcal{A}$ receives $\tilde{X}$ consisting of one random element of $\mathbb{G}_T$ and 2ℓ random elements of $\mathbb{G}_1$. Next we prove that, under the Decision Linear assumption, for all attribute vectors $\vec{z}$, the difference $|p_0^{\mathcal{A}}(\vec{z}) - p_\ell^{\mathcal{A}}(\vec{z})|$ is negligible. This implies the semantic security of the scheme.

Lemma 1 *Assume BDDH holds. Then for any attribute string $\vec{z}$ and for any adversary $\mathcal{A}$,*

$$|p_0^{\mathcal{A}}(\vec{z}) - p_1^{\mathcal{A}}(\vec{z})|$$

is non-negligible.

PROOF. Suppose that there exist PPT adversary $\mathcal{A}$ and attribute vector $\vec{z}$ for which

$$|p_0^{\mathcal{A}}(\vec{z}) - p_1^{\mathcal{A}}(\vec{z})|$$

is non-negligible. We construct a successful adversary $\mathcal{B}$ for the experiment $\mathsf{BDDHExp}$.

$\mathcal{B}$ takes in input $[\mathcal{I}, A = g_1^a, B = g_1^b, \hat{A} = g_2^a, \hat{B} = g_2^b, C = g_1^c, Z]$. $\mathcal{B}$, depending on whether $Z = \mathsf{e}(g_1, g_2)^{abc}$ or Z is a random element of $\mathbb{G}_T$, simulates experiment $\mathsf{SemanticExp}(1^n, 1^\ell, \vec{z}, 0)$ or $\mathsf{SemanticExp}(1^n, 1^\ell, \vec{z}, 1)$ for $\mathcal{A}$. We next describe algorithm $\mathcal{B}$.

$\mathcal{B}$ starts by simulating the Key-generation Phase. $\mathcal{B}$ sets $Y = \mathsf{e}(A, \hat{B})$, which implicitly sets $y = a \cdot b$. For $i = 1, \dots, \ell$, $\mathcal{B}$ chooses random $t'_{i,0}, t'_{i,1}, v'_{i,0}, v'_{i,1} \in \mathbb{Z}_p$ and sets

$$T_{i,z_i} = g_1^{t'_{i,z_i}}, V_{i,z_i} = g_1^{v'_{i,z_i}}, T_{i,1-z_i} = B^{t'_{i,1-z_i}}, V_{i,1-z_i} = B^{v'_{i,1-z_i}}.$$

This setting implicitly defines values $\bar{T}_{i,d}$ and $\bar{V}_{i,d}$ as follows

$$\bar{T}_{i,z_i} = g_1^{1/t'_{i,z_i}}, \bar{V}_{i,z_i} = g_1^{1/v'_{i,z_i}}, \bar{T}_{i,1-z_i} = g_2^{1/(bt'_{i,1-z_i})}, \bar{V}_{i,1-z_i} = g_2^{1/(bv'_{i,1-z_i})}.$$

Therefore, after this step key $\mathsf{SK} = (\mathcal{I}, Y, y, \mathsf{SK}_1, \dots, \mathsf{SK}_\ell)$ with

$\mathsf{SK}_i = (T_{i,0}, T_{i,1}, V_{i,0}, V_{i,1}, \bar{T}_{i,0}, \bar{T}_{i,1}\bar{V}_{i,0}, \bar{V}_{i,1})$ is implicitly defined. Notice that SK has the same distribution as a key given in output by $\mathsf{MasterKeyGen}$.

$\mathcal{B}$ answers $\mathcal{A}$'s Enc queries for vector $\vec{x}$ by executing procedure Enc. Notice that Enc only needs values $T_{i,d}$'s and $V_{i,d}$'s, for $d = 0, 1$, which are known to $\mathcal{B}$ from the previous step.

$\mathcal{A}$'s queries to KeyGen for pattern vector $\vec{k}$ such that $\mathsf{Match}(\vec{z}, \vec{k}) = 0$ are answered as follows. Let $j \in S_{\vec{k}}$ be an index for which $z_j \neq k_j$ (there must exist at least one such index). If $k_i = z_i$ then $\mathcal{B}$ sets

$$R_i = \hat{B}^{a'_i/t'_{i,k_i}} \text{ and } W_i = \hat{B}^{a'_i/v'_{i,k_i}}$$

whereas, if $k_i \neq z_i$, $\mathcal{B}$ sets

$$R_i = g_2^{a'_i/t'_{i,k_i}} \text{ and } W_i = g_2^{a'_i/v'_{i,k_i}}.$$

Finally, $\mathcal{B}$ sets

$$R_j = \hat{A}^{1/t'_{j,k_j}} \cdot g_2^{-a'/t'_{j,k_j}} \text{ and } W_j = \hat{A}^{1/v'_{j,k_j}} \cdot g_2^{-a'/v'_{j,k_j}},$$

where $a' = \sum_{i \in S_{\vec{k}} \setminus \{j\}} a'_i$. $\mathcal{B}$ returns $\tilde{K} = (R_i, W_i)_{i \in S_{\vec{k}}}$.

We next show that, even though $\mathcal{B}$ does not have complete access to SK, $\tilde{K}$ has the same distribution of the output of the KeyGen procedure on input SK and $\vec{k}$.

Set $a_j = b(a - a')$ and $a_i = ba'_i$ for $i \in S_{\vec{k}} \setminus \{j\}$. Then we have

$$R_j = g_2^{(a-a')/t'_{j,k_j}} = \bar{T}_{j,k_j}^{b(a-a')} = \bar{T}_{j,k_j}^{a_j}.$$

Similarly, we have

$$W_j = \bar{V}_{j,k_j}^{a_j}.$$

For $i \in S_{\vec{k}} \setminus \{j\}$ such that $z_i = k_i$ we have

$$R_i = g_2^{a_i/t'_{i,k_i}} = \bar{T}_{i,k_i}^{a_i}$$

and similarly we have that $W_i = \bar{V}_{i,k_i}^{a_i}$. Finally, for $i \in S_{\vec{k}} \setminus \{j\}$ such that $z_i \neq k_i$ we have

$$R_i = g_2^{a_i/(bt'_{i,k_i})} = \bar{T}_{i,k_i}^{a_i}$$

and $W_i = \bar{V}_{i,k_i}^{a_i}$. We can thus conclude that $\tilde{K} = \mathsf{KeyGen}(\mathsf{SK}, \vec{k}; (a_i)_{i \in S_{\vec{k}}})$. Notice that the a_i are randomly chosen in $\mathbb{Z}_p$ under the constraint that $\sum_{i \in S_{\vec{k}}} a_i = ab = y$. Therefore the answer $\mathcal{A}$ receives from $\mathcal{B}$ has the same distribution as in $\mathsf{SemanticExp}(1^n, 1^l, \vec{z}, 0)$ and $\mathsf{SemanticExp}(1^n, 1^l, \vec{z}, 1)$.

When $\mathcal{B}$ is asked by $\mathcal{A}$ to provide encrypted attribute vector $\tilde{X}$, $\mathcal{B}$ picks s_i at random from $\mathbb{Z}_p$, for $i = 1, \ldots, \ell$, and sets $\Omega = Z^{-1}$ and

$$X_i = C^{t'_{i,z_i}} T_{i,z_i}^{-s_i} \text{ and } Z_i = V_{i,z_i}^{s_i}.$$

$\mathcal{B}$ returns $\tilde{X} = (\Omega, (X_i, Z_i)_{i=1}^{\ell})$.

Observe that, if $Z = \mathsf{e}(g_1, g_2)^{abc}$, then $\Omega = Y^{-c}$, $X_i = g_1^{t'_{i,z_i} c} \cdot T_{i,z_i}^{-s_i} = T_{i,z_i}^{c-s_i}$, $Z_i = g_1^{v'_{i,z_i} s_i} = V_{i,z_i}^{s_i}$, for $i = 1, \ldots, \ell$, and thus $\tilde{X} = \mathsf{Enc}(\mathsf{SK}, \vec{z}; c, s_1, \cdots, s_l)$. Since c is random in $\mathbb{Z}_p$ we can conclude that $\tilde{X}$ has the same distribution as in $\mathsf{SemanticExp}_{\mathcal{A}}(1^n, 1^\ell, \vec{z}, 0)$. If instead Z is random in $\mathbb{G}_T$ then $\tilde{X}$ is distributed as in $\mathsf{SemanticExp}_{\mathcal{A}}(1^n, 1^\ell, \vec{z}, 1)$.

Finally $\mathcal{B}$ returns $\mathcal{A}$'s output.

By the above reasoning, we have that when $Z = \mathsf{e}(g_1, g_2)^{abc}$, $\mathcal{B}$ outputs 1 with probability $p_0^{\mathcal{A}}(\vec{z})$ and if Z is random then $\mathcal{B}$ outputs 1 with probability $p_1^{\mathcal{A}}(\vec{z})$. Since the difference between the two probabilities is assumed non-negligible, $\mathcal{B}$ breaks the BDDH assumption. $\square$

Lemma 2 *Assume DLinear holds. Then, for any attribute string $\vec{z}$, for any adversary $\mathcal{A}$, and for $1 \leq j \leq \ell - 1$*

$$|p_j^{\mathcal{A}}(\vec{z}) - p_{j+1}^{\mathcal{A}}(\vec{z})|$$

is negligible.

Proof. Suppose that there exist PPT adversary $\mathcal{A}$ and attribute vector $\vec{z}$ for which $|p_j^{\mathcal{A}}(\vec{z}) - p_{j+1}^{\mathcal{A}}(\vec{z})|$ is non-negligible. We next construct an adversary $\mathcal{B}$ for the experiment DLExp. $\mathcal{B}$ takes in input $[\mathcal{I}, U_1 = g_1^{u_1}, U_2 = g_2^{u_2}, \hat{U}_1 = g_2^{u_1}, \hat{U}_2 = g_2^{u_2}, U_{13} = g_1^{u_1 u_3}, U = g_1^u, Z]$, and depending on whether $Z = g_1^{u_2(u-u_3)}$ or Z is a random element of $\mathbb{G}_1$ simulates experiment $\mathsf{SemanticExp}(1^n, 1^\ell, \vec{z}, j)$ or $\mathsf{SemanticExp}(1^n, 1^\ell, \vec{z}, j+1)$ for $\mathcal{A}$. We next describe algorithm $\mathcal{B}$.

$\mathcal{B}$ start by simulating the key-generation phase and sets $Y = \mathsf{e}(U_1, \hat{U}_2)$ thus implicitly setting $y = u_1 \cdot u_2$. $\mathcal{B}$ chooses random $t'_{i,0}, v'_{i,0}, t'_{i,1}, v'_{i,1} \in \mathbb{Z}_p$, for $i = 1, \ldots, \ell$, and computes $T_{j,d}$ and $V_{j,d}$ as follows. If $z_j = 0$, $\mathcal{B}$ sets

$$T_{j,0} = U_2^{t'_{j,0}}, T_{j,1} = U_1^{t'_{j,1}}, V_{j,0} = U_1^{v'_{j,0}}, V_{j,1} = U_1^{v'_{j,1}},$$

whereas, if $z_j = 1$, $\mathcal{B}$ sets

$$T_{j,0} = U_1^{t'_{j,0}}, T_{j,1} = U_2^{t'_{j,1}}, V_{j,0} = U_1^{v'_{j,0}}, V_{j,1} = U_1^{v'_{j,1}}.$$

This setting implicitly defines values $\bar{T}_{j,d}$ and $\bar{V}_{j,d}$ induced by the values $T_{j,d}$ and $V_{j,d}$, for $i \neq j$. Then $\mathcal{B}$ computes values $T_{i,d}$ and $V_{i,d}$ for $i \neq j$ as follows. If $z_i = 0$ then $\mathcal{B}$ sets

$$T_{i,1} = U_1^{t'_{i,1}}, T_{i,0} = g_1^{t'_{i,0}}, V_{i,1} = U_1^{v'_{i,1}}, V_{i,0} = g_1^{v'_{i,0}},$$

whereas, if $z_i = 1$, then $\mathcal{B}$ sets

$$T_{i,1} = g_1^{t'_{i,1}}, T_{i,0} = U_1^{t'_{i,0}}, V_{i,1} = g_1^{v'_{i,1}}, V_{i,0} = U_1^{v'_{i,0}}.$$

This setting implicitly defines values $\bar{T}_{i,d}$ and $\bar{V}_{i,d}$ induced by the values $T_{i,d}$ and $V_{i,d}$. After this step key $\mathsf{SK} = (\mathcal{I}, Y, y, \mathsf{SK}_1, \ldots, \mathsf{SK}_\ell)$ with $\mathsf{SK}_i = (T_{i,0}, T_{i,1}, V_{i,0}, V_{i,1}, \bar{T}_{i,0}, \bar{T}_{i,1}, \bar{V}_{i,0}, \bar{V}_{i,1})$ is implicitly defined. Notice that SK has the same distribution as a key given in output by $\mathsf{MasterKeyGen}$.

$\mathcal{B}$ answers $\mathcal{A}$'s Enc queries for vector $\vec{x}$ by executing procedure Enc. Notice that Enc only needs values $T_{i,b}$'s and $V_{i,b}$'s which are known to $\mathcal{B}$ from the previous step.

To describe how $\mathcal{B}$ answers $\mathcal{A}$'s queries to the oracle KeyGen we distinguish two cases.

Case 1: $k_j = z_j$ or $k_j = \star$. In this case there exists index $h \in S_{\vec{k}}$ such that $z_h \neq k_h$. Then, for $i \in S_{\vec{k}} \setminus \{j\}$, B chooses random $a'_i \in \mathbb{Z}_p$, and sets $a' = \sum_{i \in S_{\vec{k}} \setminus \{j,h\}} a'_i$.

For $i \in S_{\vec{k}} \setminus \{j, h\}$, if $k_i = z_i$ then $\mathcal{B}$ sets

$$R_i = \hat{U}_1^{a'_i/t'_{i,k_i}} \quad \text{and} \quad W_i = \hat{U}_1^{a'_i/v'_{i,k_i}}$$

and if $k_i \neq z_i$ then $\mathcal{B}$ sets

$$R_i = g_2^{a'_i/t'_{i,k_i}} \quad \text{and} \quad W_i = g_2^{a'_i/v'_{i,k_i}}.$$

Moreover, if $j \in S_{\vec{k}}$, $\mathcal{B}$ sets

$$R_j = \hat{U}_1^{a'_j/t'_{j,k_j}} \quad \text{and} \quad W_j = \hat{U}_2^{a'_j/t'_{j,k_j}}.$$

Finally, $\mathcal{B}$ sets

$$R_h = \hat{U}_2^{(1-a'_j)/t'_{h,k_h}} \cdot g_2^{-a'/t'_{h,k_h}} \quad \text{and} \quad W_h = \hat{U}_2^{(1-a'_j)/v'_{h,k_h}} \cdot g_2^{-a'/v'_{h,k_h}}.$$

$\mathcal{B}$ returns $\tilde{K} = (R_i, W_i)_{i \in S_{\vec{k}}}$.

We next show that, even though $\mathcal{B}$ does not have complete access to SK, $\tilde{K}$ has the same distribution of the output of the KeyGen procedure on input SK and $\vec{k}$.

Set $a_i = u_1 a'_i$, for $i \in S_{\vec{k}} \setminus \{h, j\}$, $a_j = u_1 u_2 a'_j$, and $a_h = u_1 u_2 - u_1 u_2 a'_j - u_1 a'$. Then we have that for $i \in S_{\vec{k}} \setminus \{j, h\}$, if $k_i = z_i$ then

$$R_i = \hat{U}_1^{a'_i/t'_{i,k_i}} = g_2^{a_i/t'_{i,k_i}} = \bar{T}_{i,k_i}^{a_i}.$$

Instead if $k_i \neq z_i$ then

$$R_i = g_2^{a'_i/t'_{i,k_i}} = g_2^{u_1 a'_i/u_1 t'_{i,k_i}} = g_2^{a_i/u_1 t'_{i,k_i}} = \bar{T}_{i,k_i}^{a_i}.$$

Similarly, we have in both cases

$$W_i = \bar{V}_{i,k_i}^{a_i}.$$

Furthermore, we have that if $j \in S_{\vec{k}}$ then

$$R_j = \hat{U}_1^{a'_j/t'_{j,k_j}} = g_2^{u_1 a'_j/t'_{j,k_j}} = g_2^{u_1 u_2 a'_j/u_2 t'_{j,k_j}} = g_2^{a_j/u_2 t'_{j,k_j}} = \bar{T}_{j,k_j}^{a_j}.$$

and similarly we have that $W_j = \bar{V}_{j,k_j}^{a_i}$. Finally, we have

$$R_h = \hat{U}_2^{(1-a'_j)/t'_{h,k_h}} g_2^{-a'/t'_{h,k_h}} = g_2^{(u_2 - u_2 a_j - a')/t'_{h,k_h}} = g_2^{u_1(u_2-u_2a_j-a')/u_1 t'_{h,k_h}}$$

$$= g_2^{a_h/u_1 t'_{h,k_h}} = \bar{T}_{h,k_h}^{a_h}.$$

and similarly we have that $W_h = \bar{V}_{h,k_h}^{a_h}$.

We can thus conclude that $\tilde{K} = \mathsf{KeyGen}(\mathsf{SK}, \vec{k}; (a_i)_{i \in S_{\vec{k}}})$. Moreover, the a_i's are independently and randomly chosen in $\mathbb{Z}_p$ under the constraint that their sum is $u_1 u_2 = y$. Hence $\tilde{K}$ is distributed according to $\mathsf{KeyGen}(\mathsf{SK}, \vec{k})$.

Case 2: $x_j \neq k_j$ In this case, for $i \in S_{\vec{k}} \setminus \{j\}$ $\mathcal{B}$ chooses random $a'_i \in \mathbb{Z}_p$ and sets $a' = \sum_{i \in S_{\vec{k}} \setminus \{j\}} a'_i$ and sets R_i and W_i as in the previous case. To compute R_j and W_j, $\mathcal{B}$ sets

$$R_j = \hat{U}_2^{1/t'_{j,k_j}} \cdot g_2^{-a'/t'_{j,k_j}} \quad \text{and} \quad W_j = \hat{U}_2^{1/v'_{j,k_j}} \cdot g_2^{-a'/v'_{j,k_j}}.$$

If we set $a_j = u_1 u_2 - u_1 a'$ and, for $i \in S_{\vec{k}} \setminus \{j\}$, $a_i = u_1 a'_i$, we have that

$$R_j = \hat{U}_2^{1/t'_{j,k_j}} \cdot g_2^{-a'/t'_{j,k_j}} = g_2^{(u_2-a')/t'_{j,k_j}} = g_2^{u_1(u_2-a')/u_1 t'_{j,k_j}} = g_2^{a_j/u_1 t'_{j,k_j}} = \bar{T}_{j,k_j}^{a_j}.$$

Similarly, we have $W_j = \bar{V}_{j,k_j}^{a_j}$. Furthermore, like in the previous case, for $i \in S_{\vec{k}} \setminus \{j\}$ we have that $R_i = \bar{T}_{i,k_i}^{a_i}$ and $W_i = \bar{V}_{i,k_i}^{a_i}$.We thus conclude that $\tilde{K} = \mathsf{KeyGen}(\mathsf{SK}, \vec{k}; (a_i)_{i \in S_{\vec{k}}})$. Moreover, the a_i's are independently and randomly chosen in $\mathbb{Z}_p$ under the constraint that their sum is $u_1 u_2 = y$. Hence, also in this case, $\tilde{K}$ is distributed according to $\mathsf{KeyGen}(\mathsf{SK}, \vec{k})$.

When $\mathcal{B}$ is asked to provide encrypted attribute vector for $\vec{z}$, $\mathcal{B}$ chooses random $\Omega \in \mathbb{G}_T$ and, for $j \leq i \leq \ell$, chooses random $s_i \in \mathbb{Z}_p$.

$\mathcal{B}$ then constructs the tuple $\tilde{X} = (\Omega, (X_i, Z_i)_{i=1}^{\ell})$ in the following way, For $i < j$, X_i and Z_i are chosen uniformly from $\mathbb{G}_1$. For $i > j$, $\mathcal{B}$ computes X_i and Z_i as

$$X_i = U^{t'_{i,z_i}} g_1^{-t'_{i,z_i} s_i} \quad \text{and} \quad Z_i = g_1^{v'_{i,z_i} s_i}$$

Finally, X_j and Y_j are computed as

$$X_j = Z^{t'_{j,z_j}} \quad \text{and} \quad Z_j = U_{13}^{v'_{j,z_j}}$$

Suppose that $Z = g_1^{u_2(u-u_3)}$ and set $s = u$ and $s_j = u_3$. Then, it is easy to verify that $\tilde{X} = \mathsf{SemanticExp}(1^n, 1^\ell, \vec{z}, j-1; s, s_j, \cdots, s_\ell)$. Moreover s and the s_i's are random in $\mathbb{Z}_p$ and thus we can conclude that $\tilde{X}$ is distributed as in $\mathsf{SemanticExp}(1^n, 1^\ell, \vec{z}, j+1)$.

Suppose instead that Z is random in $\mathbb{G}_1$. Then X_j and Y_j are also random and it is easy to verify that $\tilde{X} = \mathsf{SemanticExp}(1^n, 1^\ell, \vec{z}, j; s, s_{j+1}, \cdots, s_\ell)$. Since s and the s_i's are random in $\mathbb{Z}_p$, we can conclude that the challenge received by $\mathcal{A}$ is distributed as in $\mathsf{SemanticExp}(1^n, 1^\ell, \vec{z}, j+1)$.

Finally $\mathcal{B}$ returns $\mathcal{A}$'s output.

By the observations above, we can say that if $Z = g_1^{u_2(u-u_3)}$ then $\mathcal{A}$'s view is the same as in $\mathsf{SemanticExp}(1^n, 1^\ell, \vec{z}, j)$ and if Z is randomly and uniformly distributed in $\mathbb{G}_1$ then $\mathcal{A}$'s view is the same as in $\mathsf{SemanticExp}(1^n, 1^\ell, \vec{z}, j+1)$. This contradicts the DLinear assumption. □

Combining Lemma 1 and Lemma 2 and by noticing that DLinear implies BDDH, we have the following lemma.

Lemma 3 *Assume DLinear. Then predicate encryption* (MasterKeyGen, Enc, KeyGen, Test) *is semantically secure.*

6 Proof of Key Confidentiality

In this section, we prove the construction of Section 4 is key secure, under Assumption Q. We use the following experiments for $\eta \in \{0,1\}$.

$\mathsf{KeyExp}_{\mathcal{A}}(1^n, 1^\ell, \vec{z}_0, \vec{z}_1, \eta)$

1. Key-Generation Phase. The secret key SK is generated by the $\mathsf{MasterKeyGen}$ procedure.
2. Query Phase I. $\mathcal{A}$ can make any number of key and encryption query.
   A key query for pattern $\vec{k}$ is answered by returning $\mathsf{KeyGen}(\mathsf{SK}, \vec{k})$.
   An encryption query for attribute vector $\vec{x}$ is answered as follows.
   If $\mathsf{Match}(\vec{x}, \vec{z}_0) = \mathsf{Match}(\vec{x}, \vec{z}_1) = 0$ then $\mathcal{A}$ receives $\mathsf{Enc}(\mathsf{SK}, \vec{x})$. Otherwise, $\mathcal{A}$ receives $\perp$.
3. Challenge construction.
   $\mathcal{A}$ receives $\mathsf{KeyGen}(\mathsf{SK}, \vec{z}_\eta)$.
4. Query Phase II. Identical to Query Phase I.
5. Output Phase. $\mathcal{A}$ returns η'.

We denote by $p_{\mathcal{A}}(\vec{z}_0, \vec{z}_1, \eta)$ the probability that $\mathsf{KeyExp}_{\mathcal{A}}(1^n, 1^\ell, \vec{z}_0, \vec{z}_1, \eta)$ returns η. In the next lemma, we prove that, if $\vec{z}_0$ and $\vec{z}_1$ have no $\star$-entry and they differ in exactly m positions then the (ℓ, m)-Q assumption implies that

$$|p_{\mathcal{A}}(\vec{z}_0, \vec{z}_1, 0) - p_{\mathcal{A}}(\vec{z}_0, \vec{z}_1, 1)|$$

is negligible for all probabilistic poly-time adversaries. A similar (omitted) proof shows that, if $\vec{z}_0$ and $\vec{z}_1$ contain k $\star$'s in the same positions and differ in exactly m positions then the $(\ell - k, m)$-Q assumption implies that

$$|p_{\mathcal{A}}(\vec{z}_0, \vec{z}_1, 0) - p_{\mathcal{A}}(\vec{z}_0, \vec{z}_1, 1)|$$

is negligible.

Lemma 4 *Assume Assumption (ℓ, m)-Q holds. Then, for all probabilistic poly-time adversaries $\mathcal{A}$ and for all vectors $\vec{z}_0, \vec{z}_1 \in \{0,1\}^\ell$ which differ in exactly m positions, we have that*

$$|p_{\mathcal{A}}(\vec{z}_0, \vec{z}_1, 0) - p_{\mathcal{A}}(\vec{z}_0, \vec{z}_1, 1)|$$

is negligible.

Proof. Write $\vec{z}_0 = \langle z_{0,1}, \ldots, z_{0,\ell} \rangle$ and $\vec{z}_1 = \langle z_{1,1}, \ldots, z_{1,\ell} \rangle$ and assume, without loss of generality, that $\vec{z}_0$ and $\vec{z}_1$ differ in exactly the first m positions and that $\vec{z}_0 = 0^m \cdot 0^{\ell - m}$ and $\vec{z}_1 = 1^m \cdot 0^{\ell - m}$.

We proceed by contradiction. We assume that the lemma does not hold for some probabilistic poly-time adversary $\mathcal{A}$, and prove that there exists a probabilistic poly-time distinguisher $\mathcal{B}$ that has a non-negligible advantage for Assumption (ℓ, m)-Q.

We now describe $\mathcal{B}$. $\mathcal{B}$ takes as input a challenge $\mathsf{ChOutput}$ for Assumption (ℓ, m)-Q, simulates $\mathsf{KeyExp}_{\mathcal{A}}$ with parameters $(1^n, 1^\ell, \vec{z}_0, \vec{z}_1, \eta)$ for $\mathcal{A}$ and uses $\mathcal{A}$'s output to obtain non-negligible advantage in the game of Assumption (ℓ, m)-Q.

Initialization Phase. $\mathcal{B}$ starts by choosing random $y \in \mathbb{Z}_p$ and by setting $Y = \mathsf{e}(g_1, g_2)^y$. Define $JH = \{(j,h) | 1 \leq j \leq m,\ 1 \leq h \leq m,\ j \neq h \text{ or } j = h,\ m+1 \leq j \leq d\}$. For $(j,h) \in JH$, $\mathcal{B}$ sets[2]

$$G_{j,h} = \mathsf{e}(\mathsf{A_{j,h}}[1,j], C_{j,1}).$$

Throughout the simulation we will consider secret key $\mathsf{SK} = (\mathcal{I}, Y, y, \mathsf{SK}_1, \ldots, \mathsf{SK}_\ell)$ *implicitly* defined by $\mathsf{ChOutput}$, with $\mathsf{SK}_i = (T_{i,0}, T_{i,1}, V_{i,0}, V_{i,1}, \bar{T}_{i,0}, \bar{T}_{i,1}, \bar{V}_{i,0}, \bar{V}_{i,1})$, for $i = 1, \ldots, \ell$, where, for $i = 1, \ldots, \ell$ and $b = 0, 1$,

$$\begin{array}{ll} T_{i,b} = g_1^{\hat{t}_{i,b}}, & V_{i,b} = g_1^{\hat{v}_{i,b}}, \\ \bar{T}_{i,b} = C_{i,b}, & \bar{T}_{i,1} = D_{i,1}. \end{array}$$

[2]Hereafter, we assume that $\mathsf{A_{j,h}}$'s ($\mathsf{B_{j,h}}$'s) rows are indexed by 0 and 1.

This implies that, for $i = 1, \ldots, \ell$ and $b = 0, 1$,

$$t_{i,b} = \hat{t}_{i,b} \text{ and } \quad v_{i,b} = \hat{v}_{i,b}.$$

Since, for $i = 1, \ldots, \ell$, and for $b = 0, 1$ the values $\hat{t}_{i,b}, \hat{v}_{i,b}$ are random from $\mathbb{Z}_p$, the key SK is uniformly distributed as the output of $\mathsf{MasterKeyGen}$. We stress that $\mathcal{B}$ only has indirect access to SK through $\mathsf{ChOutput}$ and in what follows we show that this is sufficient for simulating KeyExp.

Answering encryption queries. To answer queries to the Enc oracle for attribute vectors $\vec{x} = \langle x_1, \ldots, x_\ell \rangle$, we distinguish two cases.

Case 1. The vector $\vec{x}$ is such that there exists and index $j \geq m+1$ such that $x_j = 1$. $\mathcal{B}$ chooses $s', s'_1, \ldots, s'_\ell$ at random in $\mathbb{Z}_p$, sets $\Omega = G_{j,j}^{-ys'}$ and, for $i = 1, \ldots, \ell$, sets

$$X_i = (\mathsf{A_{j,j}}[x_i, i])^{s'-s'_i} \quad \text{and} \quad Z_i = (\mathsf{B_{j,j}}[x_i, i])^{s'_i}.$$

$\mathcal{B}$ returns $\tilde{X} = (\Omega, (X_i, Z_i)_{i=1}^{\ell})$ as output of the query.

Case 2. The vector $\vec{x}$ is such that $x_j = 0$ for $m+1 \leq j \leq \ell$. Since $\mathsf{Match}(\vec{x}, \vec{z}_0) = \mathsf{Match}(\vec{x}, \vec{z}_1)$, then there exist two indices j and h such that $x_j = 1$ and $x_h = 0$. $\mathcal{B}$ chooses $s', s'_1, \ldots, s'_\ell$ at random in $\mathbb{Z}_p$, sets $\Omega = G_{j,h}^{-ys'}$ and, for $i = 1, \ldots, \ell$, sets

$$X_i = (\mathsf{A_{j,h}}[x_i, i])^{s'-s'_i} \quad \text{and} \quad Z_i = (\mathsf{B_{j,h}}[x_i, i])^{s'_i}.$$

$\mathcal{B}$ returns $\tilde{X} = (\Omega, (X_i, Z_i)_{i=1}^{\ell})$ as output of the query.

We notice that, in both above described cases, $\mathcal{B}$ can perform the computation as it has access to the needed values from $\mathsf{ChOutput}$ and from the initialization phase. Let us now argue that the output returned by $\mathcal{B}$ has the same distribution as in KeyExp. By setting, in **Case 1**, $s = s'\hat{s}_{(j,j)}$ and $s_i = s'_i\hat{s}_{(j,j)}$, for $i = 1, \ldots, \ell$; and, in **Case 2**, $s = s'\hat{s}_{(j,h)}$ and $s_i = s'_i\hat{s}_{(j,h)}$, for $i = 1, \ldots, \ell$, we have that $X_i = T_{i,x_i}^{s-s_i}$ and $Z_i = V_{i,x_i}^{s_i}$. Thus, $\tilde{X} = \mathsf{Enc}(\mathsf{SK}, \vec{x}; s, s_1, \ldots, s_\ell)$. Moreover, since s and the s_i's are random and independently chosen from $\mathbb{Z}_p$ we can conclude that $\tilde{X}$ has the same distribution as the answers obtained by $\mathcal{A}$ in $\mathsf{KeyExp}_{\mathcal{A}}$.

Answering key queries. To answer to the queries to the KeyGen oracle for attribute vector $\vec{k} = \langle k_1, \ldots, k_\ell \rangle$, $\mathcal{B}$, for $i \in S_{\vec{k}}$, chooses random $a_i \in \mathbb{Z}_p$ such that their sum is y and sets

$$R_i = C_{i,k_i}^{a_i} \quad \text{and} \quad W_i = D_{i,k_i}^{a_i}.$$

$\mathcal{B}$ returns $\tilde{K} = (R_i, W_i)_{i \in S_{\vec{k}}}$. Notice that, for $i = 1, \ldots, \ell$, we have $C_{i,k_i} = \bar{T}_{i,k_i}$ and $D_{i,k_i} = \bar{V}_{i,k_i}$. Therefore, we can conclude that $\tilde{K} = \mathsf{KeyGen}(\mathsf{SK}, \vec{k};\ (a_i)_{i \in S_{\vec{k}}})$. Since the a_i are random in $\mathbb{Z}_p$ under the constraint that their sum is y, we can conclude that that $\tilde{K}$ has the same distribution as the answers obtained by $\mathcal{A}$ in $\mathsf{KeyExp}_{\mathcal{A}}$.

Challenge construction. We describe how $\mathcal{B}$ prepares the challenge for $\mathcal{A}$.
$\mathcal{B}$ chooses, for $i = m+1, \ldots, \ell$, random $b'_i \in \mathbb{Z}_p$ under the constraint that their sum is y and returns $\tilde{K} = ((R_1, W_1), \ldots, (R_\ell, W_\ell))$ computed as follows. For $i = 1, \ldots, m$, $\mathcal{B}$ sets

$$R_i = E_i \qquad \text{and} \qquad W_i = F_i;$$

while, for $i = m+1, \ldots, \ell$, $\mathcal{B}$ sets

$$R_i = E_i \cdot C_{i,0}^{b'_i} \qquad \text{and} \qquad W_i = F_i \cdot D_{i,0}^{b'_i}.$$

Notice that, for $i = m+1, \ldots, \ell$, we have $R_i = \bar{T}_{i,0}^{a_i}$ and $W_i = \bar{V}_{i,0}^{a_i}$ where $a_i = \hat{a}_i + b'_i$. In addition, for $i = 1, \ldots, m$, we have $R_i = \bar{T}_{i,z_{\eta_i}}^{a_i}$ and $W_i = \bar{V}_{i,z_{\eta_i}}^{a_i}$ where $a_i = \hat{a}_i$. Therefore, we can conclude that $\tilde{K} = \mathsf{KeyGen}(\mathsf{SK}, \vec{z}_\eta, (a_1, \ldots, a_\ell))$. Finally, we observe that the a_i's are random in $\mathbb{Z}_p$ under the constraint that their sum is y. Thus, $\tilde{K}$ is distributed as in $\mathsf{KeyExp}_{\mathcal{A}}(1^n, 1^\ell, \vec{z}_0, \vec{z}_1, \eta)$.

Finally, when $\mathcal{A}$ halts and returns η', $\mathcal{B}$ halts and returns η'.

Since the simulation provided by $\mathcal{B}$ is perfect, by our assumption on $\mathcal{A}$'s advantage, we can conclude that the advantage of $\mathcal{B}$ is also non-negligible thus contradicting Assumption (d,m)-Q. $\square$

We thus have the following lemma.

Lemma 5 *Under Assumptions (d,m)-Q predicate encryption scheme (*$\mathsf{MasterKeyGen}$*,*Enc*,*KeyGen*,*Test*) is key secure.*

Combining Lemma 3 and Lemma 5 we have the main result of this paper.

Theorem 2 *Under Assumptions (d,m)-Q and Decision Linear predicate encryption scheme (*$\mathsf{MasterKeyGen}$*,*Enc*,*KeyGen*,*Test*) is secure* HVE.

7 Match Concealing

In this section, we show that, under the Double Decision Linear Assumption, the scheme presented in Section 4 actually enjoys a stronger notion of semantic security in which the adversary $\mathcal{A}$ is allowed to make queries for keys associated to any pattern $\vec{k}$ provided only that $\mathsf{Match}(\vec{z}_0,\vec{k}) = \mathsf{Match}(\vec{z}_1,\vec{k})$. We call this notion *match concealing*. In the notion presented in the main body of the paper, $\mathcal{A}$ is restricted to queries for patterns $\vec{k}$ such that $\mathsf{Match}(\vec{z}_0,\vec{k}) = \mathsf{Match}(\vec{z}_1,\vec{k}) = 0$. This latter notion is called *match revealing* (see [SBC+07]).

We now present the Double Decision Linear Assumption by means of the following experiment $\mathsf{DDLExp}_{\mathcal{A}}$.

```
DDLExp_A(1^n)
01. Choose instance $\mathcal{I} = [p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, \mathsf{e}]$ with security parameter $1^n$.
02. Choose $u_1, u_2, u_3, u_4, u_5, u \in \mathbb{Z}_p$ at random.
03. Choose $\eta \in \{0,1\}$ at random.
04. If $\eta = 1$, then
05.     set $Z = g_1^{u_2(u-u_3)}$ and $Z_0 = g_1^{u_1 u_3}$;
06.     else, set $Z = g_1^{u_5(u-u_3)}$ and $Z_0 = g_1^{u_4 u_3}$.
07. Set $U_1 = g_1^{u_1}, \hat{U}_1 = g_2^{u_1}, U_2 = g_1^{u_2}, U_4 = g_1^{u_4}, U_5 = g_1^{u_5}, U_{245} = g_2^{u_2 u_4 u_5}$.
08. Set $U_{145} = g_2^{u_1 u_4 u_5}, U_{125} = g_2^{u_1 u_2 u_5}, U_{124} = g_2^{u_1 u_2 u_4}, U = g_1^{u}$.
09. Let $\eta' = \mathcal{A}(\mathcal{I}, U_1, \hat{U}_1, U_2, U_4, U_5, U_{245}, U_{145}, U_{125}, U_{124}, U, Z, Z_0)$.
10. If $\eta = \eta'$ then return 1 else return 0,
```

Assumption 4 (Double Decision Linear (DDLinear)) *For all probabilistic poly-time algorithms $\mathcal{A}$,*

$$|\mathrm{Prob}[\mathsf{DDLExp}_{\mathcal{A}}(1^n) = 1] - 1/2|$$

is negligible in n.

Suppose that $\vec{z}_0, \vec{z}_1$ are two attribute vectors in $\{0,1\}^\ell$ which differ only in position j. Consider the following experiments.

$\mathsf{SemanticExp}_{\mathcal{A}}(1^n, 1^\ell, \vec{z}_0, \vec{z}_1, \eta)$

1. Key-generation Phase. Compute $\mathsf{SK} = (\mathcal{I}, y, \mathsf{SK}_1, \cdots, \mathsf{SK}_\ell)$ by executing $\mathsf{MasterKeyGen}(1^n, 1^\ell)$.
2. Query Phase I. Answer Enc queries for attribute vectors $\vec{x}$ by using secret key SK. Answer KeyGen queries for pattern vectors $\vec{k}$ such that $\mathsf{Match}(\vec{z}_0,\vec{k}) = \mathsf{Match}(\vec{z}_1,\vec{k})$ using secret key SK.
3. Challenge Construction.
    1. Choose random $s, s_1, \ldots, s_\ell \in \mathbb{Z}_p$ and set $\Omega = \mathsf{e}(g_1, g_2)^{ys}$.

2. For $1 \le i \ne j \le \ell$
   set $X_i = g_1^{t_{i,z_{0,i}}(s-s_i)}$ and $Z_i = g_1^{s_i v_{i,z_{0,i}}}$.
3. set $X_j = g_1^{t_{j,z_{\eta,i}}(s-s_j)}$ and $Z_j = g_1^{s_j v_{j,z_{\eta,j}}}$.
4. Set $\tilde{X} = (\Omega, (X_i, Z_i)_{i=1}^{\ell})$.

5. Query Phase II. Identical to Query Phase I.
6. **return** $\mathcal{A}(\tilde{X})$.

We will use the writing $\mathsf{SemanticExp}(1^n, 1^\ell, \vec{z}_0, \vec{z}_1, \eta; s, s_1, \ldots, s_\ell)$ to denote the tuple $\tilde{X}$ computed by $\mathsf{SemanticExp}(1^n, 1^\ell, \vec{z}_0, \vec{z}_1, \eta)$ using $s, s_1, \ldots, s_\ell$ as random values. We will denote by $p_\eta^{\mathcal{A}}(\vec{z}_0, \vec{z}_1)$ the probability that experiment $\mathsf{SemanticExp}_{\mathcal{A}}(1^n, 1^\ell, \vec{z}_0, \vec{z}_1, \eta)$ returns η. Notice that, since $\vec{z}_0$ and $\vec{z}_1$ differ only in position j, then in $\mathsf{SemanticExp}_{\mathcal{A}}(1^n, 1^\ell, \vec{z}_0, \vec{z}_1, 0)$ adversary $\mathcal{A}$ receives a valid encrypted attribute vector $\tilde{X}$ for attribute vector $\vec{z}_0$ whereas in $\mathsf{SemanticExp}_{\mathcal{A}}(1^n, 1^\ell, \vec{z}_0, \vec{z}_1, 1)$ adversary $\mathcal{A}$ receives $\tilde{X}$ for attribute vector $\vec{z}_1$. Next we prove that, under the Double Decision Linear assumption, for all attribute vectors $\vec{z}_0, \vec{z}_1$ which differ only in position j, the difference $|p_0^{\mathcal{A}}(\vec{z}_0, \vec{z}_1) - p_1 \ell^{\mathcal{A}}(\vec{z}_0, \vec{z}_1)|$ is negligible. This implies the match concealing semantic security of the scheme.

Lemma 6 *Assume DDLinear holds. Then, for any j, for any attribute strings $\vec{z}_0$ and $\vec{z}_1$ which differ only in position j, and for any adversary $\mathcal{A}$,*

$$|p_0^{\mathcal{A}}(\vec{z}_0, \vec{z}_1) - p_1^{\mathcal{A}}(\vec{z}_0, \vec{z}_1)|$$

is negligible.

PROOF. Suppose that there exist PPT adversary $\mathcal{A}$ and attribute vector $\vec{z}_0, \vec{z}_1$ for which $|p_0^{\mathcal{A}}(\vec{z}_0, \vec{z}_1) - p_1^{\mathcal{A}}(\vec{z}_0, \vec{z}_1)|$ is non-negligible. We assume without loss of generality that, for $i \ne j$, we have $z_{0,i} = z_{1,i} = 0$ and that $z_{0,j} = 0$ and $z_{1,j} = 1$. We next construct a PPT adversary $\mathcal{B}$ for the experiment DDLExp. $\mathcal{B}$ takes in input $[\mathcal{I}, U_1 = g_1^{u_1}, \hat{U}_1 = g_2^{u_1}, U_2 = g_1^{u_2}, U_4 = g_1^{u_4}, U_5 = g_1^{u_5}, U_{245} = g_2^{u_2 u_4 u_5}, U_{145} = g_2^{u_1 u_4 u_5}, U_{125} = g_2^{u_1 u_2 u_5}, U_{124} = g_2^{u_1 u_2 u_4}, U = g_1^u, Z, Z_0]$, and depending on whether $Z = g_1^{u_2(u-u_3)}$ and $Z_0 = g_1^{u_1 u_3}$ or $Z = g_1^{u_5(u-u_3)}, Z_0 = g_1^{u_4 u_3}$, simulates experiment $\mathsf{SemanticExp}(1^n, 1^\ell, \vec{z}_0, \vec{z}_1, 0)$ or $\mathsf{SemanticExp}(1^n, 1^\ell, \vec{z}, 1)$ for $\mathcal{A}$. We next describe algorithm $\mathcal{B}$.

Initialization Phase. $\mathcal{B}$ simulates the key-generation phase by choosing random $y' \in \mathbb{Z}_p$ and sets $Y = \mathsf{e}(U_1^{y'}, g_2)$. This implicitly sets $y = u_1 y'$. $\mathcal{B}$ chooses random $t'_{i,0}, v'_{i,0}, t'_{i,1}, v'_{i,1} \in \mathbb{Z}_p$, for $i \ne j$, and then computes values $T_{i,0}, T_{i,1}, V_{i,0}$, and $V_{i,1}$ as follows.

$$T_{i,0} = g_1^{t'_{i,0}}, T_{i,1} = U_1^{t'_{i,1}}, V_{i,0} = g_1^{v'_{i,0}}, \text{ and } V_{i,1} = U_1^{v'_{i,1}}.$$

These settings implicitly define $t_{i,0} = t'_{i,0}$, $t_{i,1} = u_1 \cdot t'_{i,1}$, $v_{i,0} = v'_{i,0}$, and $v_{j,1} = u_1 \cdot v'_{i,1}$ which in turn define values $\bar{T}_{i,0}, \bar{T}_{i,1}, \bar{V}_{i,0}$, and $\bar{V}_{i,1}$. Then, $\mathcal{B}$ computes $T_{j,0}, T_{j,1}, V_{j,0}$, and $V_{j,1}$ by setting

$$T_{j,0} = U_2, T_{j,1} = U_5, V_{j,0} = U_1, \text{ and } V_{j,1} = U_4,$$

thus implicitly setting $t_{j,0} = u_2$, $t_{j,1} = u_5$, $v_{j,0} = u_1$, and $v_{j,1} = u_4$ which in turn define values $\bar{T}_{j,0}, \bar{T}_{j,1}, \bar{V}_{j,0}$ and $\bar{V}_{j,1}$.

After this step key $\mathsf{SK} = (\mathcal{I}, Y, y, \mathsf{SK}_1, \ldots, \mathsf{SK}_\ell)$ with $\mathsf{SK}_i = (T_{i,0}, T_{i,1}, V_{i,0}, V_{i,1}, \bar{T}_{i,0}, \bar{T}_{i,1}, \bar{V}_{i,0}, \bar{V}_{i,1})$ is implicitly defined even though $\mathcal{B}$ does not completely know SK. Notice that SK has the same distribution as a key given in output by $\mathsf{MasterKeyGen}$.

Answering Queries. $\mathcal{B}$ answers $\mathcal{A}$'s Enc queries for vector $\vec{x}$ by executing procedure Enc. Notice that Enc only needs values $T_{i,b}$'s and $V_{i,b}$'s which are known to $\mathcal{B}$ from the previous step. To describe how $\mathcal{B}$ answers $\mathcal{A}$'s KeyGen queries for vector $\vec{k}$, we distinguish the following cases.

Case 1: $k_j \ne \star$. In this case there exists index $h \in S_{\vec{k}}$ such that $k_h = 1$, for otherwise we would have $\mathsf{Match}(\vec{z}_0, \vec{k}) \ne \mathsf{Match}(\vec{z}_1, \vec{k})$. Then, for $i \in S_{\vec{k}}$, B chooses random values $a'_i \in \mathbb{Z}_p$, and sets $a' = \sum_{i \in S_{\vec{k}} \setminus \{j,h\}} a'_i$. For $i \in S_{\vec{k}} \setminus \{j, h\}$, $\mathcal{B}$ computes R_i and W_i as follows. If $k_i = 0$, then $\mathcal{B}$ sets

$$R_i = \hat{U}_1^{a'_i / t'_{i,k_i}} \quad \text{and} \quad W_i = \hat{U}_1^{a'_i / v'_{i,k_i}}$$

else $\mathcal{B}$ sets
$$R_i = g_2^{a'_i/t'_{i,k_i}} \quad \text{and} \quad W_i = g_2^{a'_i/v'_{i,k_i}}.$$
$\mathcal{B}$ then computes R_j and W_j as follows. If $k_j = 0$, then $\mathcal{B}$ sets
$$R_j = U_{145}^{a'_j} \quad \text{and} \quad W_j = U_{245}^{a'_j},$$
else $\mathcal{B}$ sets
$$R_j = U_{124}^{a'_j} \quad \text{and} \quad W_j = U_{125}^{a'_j}.$$
Finally, $\mathcal{B}$ sets
$$R_h = U_{245}^{-a'_j/t'_{h,k_h}} g_2^{(y'-a')/t'_{h,k_h}} \quad \text{and} \quad W_h = U_{245}^{-a'_j/v'_{h,k_h}} g_2^{(y'-a')/v'_{h,k_h}}.$$
$\mathcal{B}$ returns $\tilde{K} = (R_i, W_i)_{i \in S_{\vec{k}}}$.
We next show that, even though $\mathcal{B}$ does not have complete access to SK, $\tilde{K}$ has the same distribution of the output of the KeyGen procedure on input SK and $\vec{k}$.
Set $a_i = u_1 a'_i$, for $i \in S_{\vec{k}} \backslash \{h, j\}$, $a_j = u_1 u_2 u_4 u_5 a'_j$, and $a_h = u_1 y' - u_1 u_2 u_4 u_5 a'_j - u_1 a'$. Then, for $i \in S_{\vec{k}} \backslash \{j, h\}$ such that $k_i = 0$ we have
$$R_i = \hat{U}_1^{a'_i/t'_{i,k_i}} = g_2^{u_1 a'_i/t'_{i,k_i}} = g_2^{a_i/t'_{i,k_i}} = \bar{T}_{i,0}^{a_i}.$$
Similarly, for $i \in S_{\vec{k}} \setminus \{j, h\}$ such that $k_i = 1$,
$$R_i = g_2^{a'_i/t'_{i,k_i}} = g_2^{u_1 a'_i/u_1 t'_{i,k_i}} = g_2^{a_i/u_1 t'_{i,k_i}} = \bar{T}_{i,1}^{a_i}.$$
Similarly, we have in both cases that $W_i = \bar{V}_{i,k_i}^{a_i}$. Furthermore, if $k_j = 0$ we have
$$R_j = U_{145}^{a'_j} = g_2^{u_1 u_4 u_5 a'_j} = g_2^{u_1 u_2 u_4 u_5 a'_j/u_2} = g_2^{a_j/u_2} = \bar{T}_{j,0}^{a_j}.$$
Similarly, for $k_j = 1$ and for W_j. Finally, we have
$$\begin{aligned} R_h &= U_{245}^{-a'_j/t'_{h,1}} g_2^{(y'-a')/t'_{h,1}} \\ &= g_2^{(-u_2 u_4 u_5 a'_j + y' - a')/t'_{h,1}} \\ &= g_2^{u_1(-u_2 u_4 u_5 a'_j + y' - a')/t_{h,1}} \\ &= g_2^{a_h/t_{h,1}} \\ &= \bar{T}_{h,1}^{a_h}. \end{aligned}$$
To conclude notice that the a_i's are random under the constraint that their sum is $u_1 y' = y$ and thus the simulation is perfect.

Case 2: $k_j = \star$. In this case, for $i \in S_{\vec{k}}$, $\mathcal{B}$ chooses random values $a'_i \in \mathbb{Z}_p$ which sum up to y', and computes R_i and W_i as follows. If $k_i = 0$, then $\mathcal{B}$ sets
$$R_i = \hat{U}_1^{a'_i/t'_{i,k_i}} \quad \text{and} \quad W_i = \hat{U}_1^{a'_i/v'_{i,k_i}}$$
else $\mathcal{B}$ sets
$$R_i = g_2^{a'_i/t'_{i,k_i}} \quad \text{and} \quad W_i = g_2^{a'_i/v'_{i,k_i}}.$$
If we set, for $i \in S_{\vec{k}}$, $a_i = u_1 a'_i$, we have that if $k_i = 0$ then
$$R_i = \hat{U}_1^{a'_i/t'_{i,k_i}} = g_2^{u_1 a'_i/t'_{i,k_i}} = g_2^{a_i/t'_{i,k_i}} = g_2^{a_i/t_{i,k_i}} = \bar{T}_{i,0}^{a_i},$$
and if $k_i = 1$ then
$$R_i = g_2^{a'_i/t'_{i,k_i}} = g_2^{u_1 a'_i/u_1 t'_{i,k_i}} = g_2^{a_i/t_{i,k_i}} = \bar{T}_{i,1}^{a_i}.$$

Similarly, we have $W_i = \bar{V}_{i,k_i}^{a_i}$. We thus conclude that $\tilde{K} = \mathsf{KeyGen}(\mathsf{SK}, \vec{k}; (a_i)_{i \in S_{\vec{k}}})$. Moreover, the a_i's are independently and randomly chosen in $\mathbb{Z}_p$ under the constraint that their sum is $u_1 y' = y$. Hence, also in this case, $\tilde{K}$ is distributed according to $\mathsf{KeyGen}(\mathsf{SK}, \vec{k})$.

Challenge construction. When $\mathcal{B}$ is asked to provide encrypted attribute vector for $\vec{z}_0$ or $\vec{z}_1$, $\mathcal{B}$ constructs the tuple $\tilde{X} = (\Omega, (X_i, Z_i)_{i=1}^{\ell})$ in the following way. $\mathcal{B}$ sets $\Omega = \mathsf{e}(U, \hat{U}_1)^{-y'}$, thus implicitly setting $s = u$. For $i \neq j$, $\mathcal{B}$ chooses random $s_i \in \mathbb{Z}_p$ and computes X_i and Z_i as

$$X_i = U^{t'_{i,0}} g_1^{-t'_{i,0} s_i} \quad \text{and} \quad Z_i = g_1^{v'_{i,0} s_i}.$$

Notice that the above settings implies

$$X_i = U^{t'_{i,0}} g_1^{-t'_{i,0} s_i} = g_1^{u t'_{i,0}} T_{i,0}^{-s_i} = T_{i,0}^{s - s_i} \quad \text{and} \quad Z_i = g_1^{v'_{i,0} s_i} = V_{i,0}^{s_i}.$$

Finally, X_j and Y_j are computed as

$$X_j = Z \quad \text{and} \quad Z_j = Z_0.$$

Finally $\mathcal{B}$ returns $\mathcal{A}$'s output.

Suppose that $Z = g_1^{u_2(u-u_3)}$, $Z_0 = g_1^{u_1 u_3}$ and $s_j = u_3$. Then, we have

$$X_j = U_2^{u-u_3} = T_{j,0}^{u-u_3} = T_{j,0}^{s-s_3} \quad \text{and} \quad Z_j = U_1^{u_3} = V_{j,0}^{u_3} = V_{j,0}^{s_3}$$

and thus $\tilde{X} = \mathsf{SemanticExp}(1^n, 1^{\ell}, \vec{z}_0, \vec{z}_1, 0; s, s_1, \ldots, s_{\ell})$. Moreover s and the s_i's are random in $\mathbb{Z}_p$ and thus we can conclude that $\tilde{X}$ is distributed as in $\mathsf{SemanticExp}(1^n, 1^{\ell}, \vec{z}_0, \vec{z}_1, 1)$.

Suppose instead that $Z = g_1^{u_5(u-u_3)}$ and $Z_0 = g_1^{u_4 u_3}$, and sets $s_j = u_3$ as before. Then we have

$$X_j = U_5^{u-u_3} = T_{j,1}^{u-u_3} = T_{j,1}^{s-s_3} \quad \text{and} \quad Z_j = U_4^{u_3} = V_{j,1}^{u_3} = V_{j,1}^{s_3}$$

and thus $\tilde{X} = \mathsf{SemanticExp}(1^n, 1^{\ell}, \vec{z}_0, \vec{z}_1, 1; s, s_1, \ldots, s_{\ell})$. Since s and the s_i's are random in $\mathbb{Z}_p$, we can conclude that the challenge received by $\mathcal{A}$ is distributed as in $\mathsf{SemanticExp}(1^n, 1^{\ell}, \vec{z}, 1)$. Furthermore notice that setting $s = u$ and $y = u_1 y'$ then Ω has the correct distribution.

By the observations above, we can say that if $Z = g_1^{u_2(u-u_3)}$ and $Z_0 = g_1^{u_1 u_3}$, then $\mathcal{A}$'s view is the same as in $\mathsf{SemanticExp}(1^n, 1^{\ell}, \vec{z}_0, \vec{z}_1, 0)$; whereas, if $Z = g_1^{u_5(u-u_3)}$ and $Z_0 = g_1^{u_4 u_3}$, then $\mathcal{A}$'s view is the same as in $\mathsf{SemanticExp}(1^n, 1^{\ell}, \vec{z}_0, \vec{z}_1, 1)$. This contradicts the DDLinear assumption. □

Simple hybrid arguments can extend the lemma to arbitrary $\vec{z}_0$ and $\vec{z}_1$ (and not just for vectors differing in one position).

Lemma 7 *Assume DDLinear. Then predicate encryption* (MasterKeyGen, Enc, KeyGen, Test) *is match concealing semantically secure.*

8 Larger alphabets

Our constructions have been presented for binary attribute vectors. The extension to larger alphabets is straightforward. Specifically, for an alphabet Σ of size s we would have a master secret key consisting of an instance $\mathcal{I}$ and of one element of $\mathbb{G}_T$, $2 \cdot \ell \cdot s$ elements of $\mathbb{G}_1$, and $2 \cdot \ell \cdot s$ elements of $\mathbb{G}_2$. The length of the encrypted attribute vectors and of the keys are independent of the size of Σ and only depend on ℓ. We can make the length of the secret key SK independent from the size of Σ by employing a pseudo-random function $\mathbb{F}$. Specifically, we randomly select a k-bit string R and set $t_{i,\sigma} = \mathbb{F}_R(i||\sigma)$ and $v_{i,\sigma} = \mathbb{F}_R(i||\sigma)$ for $i = 1, \ldots, \ell$ and $\sigma \in \Sigma$.

Acknowledgements

The work of the authors has been supported in part by the European Commission through the EU ICT program under Contract ICT-2007-216646 ECRYPT II and through the FP6 program under contract FP6-1596 AEOLUS.

References

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 440–456, Aarhus, Denmark, May 22–26, 2005. Springer-Verlag, Berlin, Germany.

[BDOP04] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 506–522, Interlaken, Switzerland, May 2–6, 2004. Springer-Verlag, Berlin, Germany.

[Boy08] Xavier Boyen. The uber-assumption family – a unified complexity framework for bilinear groups. In Steven D. Galbraith and Kenneth G. Paterson, editors, *Pairing-Based Cryptography - Pairing 2008, Second International Conference. Prooceedings*, volume 5209 of *LNCS*, pages 39–56, Egham, UK, September 1–3, 2008. Springer-Verlag, Berlin, Germany.

[BW06] Xavier Boyen and Brent Waters. Anonymous Hierarchical Identity-Based Encryption (Without Random Oracles). In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 290–307, Santa Barbara, CA, USA, August 20–24, 2006. Springer-Verlag, Berlin, Germany.

[BW07] Dan Boneh and Brent Waters. Conjunctive, subset and range queries on encrypted data. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 535–554, Amsterdam, The Netherlands, February 21–24, 2007. Springer-Verlag, Berlin, Germany.

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-Based Encryption for Fine-Grained Access Control for Encrypted Data. In *ACM CCS 06*, pages 89–98, Alexandria, VA, USA, October 30 - November 3, 2006. ACM Press.

[IP08] Vincenzo Iovino and Giuseppe Persiano. Hidden-vector encryption with groups of prime order. In Steven D. Galbraith and Kenneth G. Paterson, editors, *Pairing-Based Cryptography - Pairing 2008, Second International Conference. Prooceedings*, volume 5209 of *LNCS*, pages 75–88, Egham, UK, September 1–3, 2008. Springer-Verlag, Berlin, Germany.

[KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate Encryption Supporting Disjunction, Polynomial Equations, and Inner Products. In Nigel Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 146–162, Istanbul, Turkey, April 13–17, 2008. Springer-Verlag, Berlin, Germany.

[Nao03] Moni Naor. On cryptographic assumptions and challenges (invited talk). In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 96–109, Santa Barbara, CA, USA, August 17–21, 2003. Springer-Verlag, Berlin, Germany.

[SBC$^+$07] Elaine Shi, John Bethencourt, Hubert Chan, Dawn Song, and Adrian Perrig. Multi-Dimensional Range Query over Encrypted Data. In *2007 IEEE Symposium on Security and Privacy*, Oakland, CA, 2007. IEEE Computer Society Press.

[SSW09] Emily Shen, Elaine Shi, and Brent Waters. Predicate privacy in encryption systems. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 457–473, San Francisco, CA, USA, 2009. Springer-Verlag, Berlin, Germany.