2013

Fully secure hidden vector encryption under standard assumptions

Jong Hwan Park  
*Korea University - Korea*

Kwangsu Lee  
*Korea University - Korea*

Willy Susilo  
*University of Wollongong, wsusilo@uow.edu.au*

Dong Hoon Lee  
*Korea University - Korea*

Publication Details
Fully secure hidden vector encryption under standard assumptions

Abstract
Hidden Vector Encryption (HVE) is a special type of predicate encryption that can support conjunctive equality and range searches on encrypted data. All previous HVE schemes were proven to be either selectively secure or weakly attribute-hiding. In this paper, we first construct a new HVE scheme that is fully secure under standard assumptions. Our HVE scheme, which is based on bilinear maps (pairings), provides efficiency advantages in that it requires $O(1)$-sized private keys and $O(1)$ pairing computations for decryption, regardless of both the number of conjunctives and the dimension of vectors. To achieve our goal, we develop a novel technique to realize a tag-based dual system encryption in prime-order groups and show how to hide vector components and compress tag values into one. © 2013 Elsevier Inc. All rights reserved.

Keywords
assumptions, hidden, fully, vector, secure, encryption, under, standard

Disciplines
Engineering | Science and Technology Studies

Publication Details

This journal article is available at Research Online: http://ro.uow.edu.au/eispapers/639
Fully secure hidden vector encryption under standard assumptions

Jong Hwan Park\textsuperscript{a}, Kwangsu Lee\textsuperscript{a}, Willy Susilo\textsuperscript{b}, Dong Hoon Lee\textsuperscript{a,\textsuperscript{*}}

\textsuperscript{a}Center for Information Security and Technologies, Korea University, Seoul, Republic of Korea
\textsuperscript{b}School of Computer Science and Software Engineering, University of Wollongong, Northfields Avenue, Wollongong, Australia

Abstract

Hidden Vector Encryption (HVE) is a special type of predicate encryption that can support conjunctive equality and range searches on encrypted data. All previous HVE schemes were proven to be either selectively secure or weakly attribute-hiding. In this paper, we first construct a new HVE scheme that is fully secure under standard assumptions. Our HVE scheme, which is based on bilinear maps (pairings), provides efficiency advantages in that it requires \(O(1)\)-sized private keys and \(O(1)\) pairing computations for decryption, regardless of both the number of conjunctives and the dimension of vectors. To achieve our goal, we develop a novel technique to realize a tag-based dual system encryption in prime-order groups and show how to hide vector components and compress tag values into one.

1. Introduction

Recently, predicate encryption \cite{28} has received considerable attention as a new vision in public key encryption. In a predicate encryption scheme, an encryptor uses a public key \(PK\) to generate a ciphertext \(CT_{x,M}\), which is an encryption of an arbitrary access control policy \(x \in X\) as well as a message \(M\), and an authority who has a master secret key generates a secret key \(sk_y\) for another access control policy \(y \in Y\). Using \(sk_y\), the ciphertext \(CT_{x,M}\) is successfully decrypted, i.e., the decryption outputs the right message \(M\) if and only if \(P(x, y) = 1\), where \(P\) is a predicate function defined as \(P:X \times Y \rightarrow \{0, 1\}\). A primary security property of predicate encryption is that the ciphertext \(CT_{x,M}\) leaks no information about either \(x\) or \(M\),\footnote{Functional encryption \cite{11} is a broader concept including predicate encryption, which encompasses the case in which \(CT_{x,M}\) does not reveal only information about \(M\) but not about \(x\). The sub-classes of functional encryption with the revelation of \(x\) include Identity-Based Encryption (IBE) \cite{43,9,14,6,47,22,16,2}, Hierarchical IBE \cite{26,23,6,7,21,46,32,33}, Attribute-Based Encryption (ABE) \cite{41,25,37,3,35,4,31}, and Ciphertext-Policy ABE \cite{5,24,30,35,45}. We refer to \cite{11} for more precise definition and classification about functional encryption (including predicate encryption).} Nevertheless, the possibility of computing the predicate \(P(x, y)\) without revealing \(x\) from the ciphertext can provide a good solution for searching encrypted data.

One application of predicate encryption could be an electronic health record system where patients’ sensitive data should be securely encrypted. When patients’ data needs to be accessed by an outside entity, access should be limited to only the minimum necessary amount of data. In the health record system, each doctor has its own public/private key pair, and encrypts a patient’s data \(M\) each time the doctor treats a patient. The data \(M\) is encrypted along with an access policy \(y\) that the doctor generates a private key \(sk_y\) and gives it to the entity as a token. Within the security of predicate encryption, the outside entity is able to access the set of

\* Corresponding author.

\textit{E-mail addresses: decartian@korea.ac.kr} (J.H. Park), guspin@korea.ac.kr (K. Lee), wsusilo@uow.edu.au (W. Susilo), donghlee@korea.ac.kr (D.H. Lee).
ciphertexts \(\{\text{CT}_{x,M}\}\) such that \(P(x, y) = 1\), and not beyond it. This can be an exact realization of the minimum necessary requirement.

Predicate encryption can be realized in a variety of ways, depending on how the predicate function is explored over \(X \times Y\). Until now, there have been three sub-classes of predicate encryption: anonymous Identity-Based Encryption (IBE), Hidden Vector Encryption (HVE), and Inner-Product Encryption (IPE). Anonymous IBE \([1,20,12]\) supports a simple equality predicate and thus gives a simple equality search on encrypted data. HVE \([13]\) provides a conjunctive equality predicate, which can be extended to give a conjunctive combination of equality, comparison, and range searches. IPE \([28]\) employs an inner-product predicate, and this enables more complex access controls such as conjunctions, disjunctions, and polynomial evaluations. The relation between these three primitives forms a hierarchy: anonymous IBE — HVE — IPE, where \(A — B\) signifies that \(B\) implies \(A\).

1.1. Efficiency of HVE

The predicate in an HVE scheme is defined over \(\ell\)-dimensional vectors \(\bar{x} \in X\) and \(\bar{y} \in Y\). Most previous HVE schemes \([13,44,27,28,36,30,18,35,38]\) (including HVE derived from IPE), which are all pairing-based, require not only \(O(\ell)\) pairing computations to perform one decryption, but also a size \(O(\ell)\) of private keys. From the perspective of efficiency, it is desirable that the searching cost per one ciphertext is not proportional to the number \(\ell\), i.e., the cost required to decrypt one ciphertext \(\text{CT}_{x,M}\) using \(sk_y\) becomes \(O(1)\). To search for suitable ciphertexts holding \(P(\bar{x}, \bar{y}) = 1\), the decryptor should perform decryption on all ciphertexts \(\{\text{CT}_{x,M}\}\) in a storage server. This is because the decryptor does not know any information on the stored or incoming ciphertexts in advance and each ciphertext could possibly become the one that matches \(sk_y\). The \(O(\ell)\) pairing computations will become burdensome for the decryptor if the number \(\ell\) increases to deal with more expressive access control, and become seriously problematic if a large number of users can have access to the storage system.

The size of \(sk_y\) becomes an important factor since each \(sk_y\) should be transmitted in a secure channel from the authority to the decryptor. In a broadcast system with a large number of receivers, the transmission can be viewed as a reverse situation of broadcast encryption \([19]\) where a central authority broadcasts encrypted messages to many receivers. Shortening the size of broadcast ciphertexts has long been a central issue in designing broadcast encryption schemes \([10,39]\). Thus, like in broadcast encryption, it is necessary to shorten the transmission size of \(sk_y\) as the number of users increases. Also, this is especially the case when the authority is based on a device with restricted resources like a smart phone. Until now, only a few HVE schemes \([40,29]\) have achieved both \(O(1)\) pairing computations and \(O(1)\) size of private keys in a weaker security model (described below).

1.2. Security of HVE

It is better for an HVE scheme to be fully (or adaptively) secure. Full security means that an adversary is allowed to make both matching and non-matching private key queries for two target pairs \((\bar{x}_b, M_b)\) for \(b = 0, 1\). In other words, any private key query for \(\bar{y}\) is permitted as long as \(P(\bar{x}_b, \bar{y}) = P(\bar{x}_1, \bar{y})\). In fact, this is the complete security notion of HVE that was suggested in \([11]\), but no previous HVE (or even IPE) schemes have achieved full security. Most earlier HVE schemes \([13,44,27,28,36,40,29,38]\) have argued their security in a selective security model (originated from \([15]\)), albeit permitting the two type of key queries. Recently, several constructions \([30,35,18]\) have overcome the barrier of selective security by adapting the technique of dual system encryption \([46]\), but are unfortunately not yet fully secure since their security models allow an adversary to make only non-matching private key queries. This incomplete security is described as ‘weakly attribute-hiding’.

There is a strict difference between weakly attribute-hiding security and full security. In the former case, the adversary is allowed to make only non-matching queries so that it cannot employ queried keys to decrypt a challenge ciphertext that is an encryption of \((\bar{x}_b, M_b)\) for a randomly chosen \(b \in \{0, 1\}\). This ensures that the adversary does not know any information about (the whole of) \(\bar{x}_b\) and \(M_b\), provided that any matching key is not given. In contrast, full security considers an adversary that is able to ask both matching and non-matching queries. Naturally, full security encompasses weakly attributing security by additionally considering the case where an adversary is able to have matching keys. The additional security guarantees that even if the adversary knows information about the message\(^2\) and (partial) \(\bar{x}_b\) that involves the same vector components \(x_{bl} = x_{l,i}\) in \(\bar{x}_b = (x_{0,1}, \ldots, x_{y,b})\) \((b = 0, 1)\), the adversary does not gain any information about the pairwise-distinct vector components in \(\bar{x}_b\) from the ciphertext.

Although we have powerful tools like dual system encryption \([46]\) for achieving adaptive security, the resulting HVE schemes \([30,35,18]\) have been limited to weak attribute hiding. A natural direction of research would be to provide an answer to the open problem by presenting an HVE scheme that can be proven to be fully secure. Another challenge is that it is clearly desirable for HVE security to rely on well-known standard assumptions. Of all suggested HVE schemes (which are all pairing-based), only a few constructions \([27,38,35]\) have demonstrated security under the Decision Bilinear Diffie-Hellman (DBDH) and Decision Linear (DLIN) assumptions. These constructions are all based on prime-order groups and can be instantiated using either symmetric or asymmetric bilinear maps.

\(^2\) It is often called ‘token’, denoted as \(\text{TK}_y\).

\(^3\) In this case, two challenge messages should be equal, i.e., \(M_0 = M_1\).
1.3. Our contribution

We present the first HVE scheme that is fully secure under the DBDH and DLIN assumptions, and additionally achieves $O(1)$ pairing computations and $O(1)$-sized private keys. Table 1 in Section 5 will compare our scheme with previous HVE schemes in terms of efficiency and security. We have developed a new method to realize dual system encryption in prime-order groups. Our method is similar to the first case proof. Prior to our new result, several schemes have been presented to offer full security without random oracles, and only [35] is fully secure under the DLIN assumption. Compared to [35], our anonymous IBE scheme is more efficient in all respects.

2. Preliminaries

2.1. Hidden vector encryption

Let $\Sigma$ be an arbitrary set of attributes, and let $*$ be a wildcard character which is not involved with any attribute. We let $I = \Sigma \cup \{\ast\}$. We then use two $\ell$-dimensional vectors, $x = (x_1, \ldots, x_\ell) \in \Sigma^\ell$ in the encryption phase and $\sigma = (\sigma_1, \ldots, \sigma_\ell) \in I^\ell$ in the decryption phase.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Group order</th>
<th>PK size</th>
<th>Ciphertext size</th>
<th>Token size</th>
<th>Decryption cost</th>
<th>Selective or full</th>
<th>Standard assumptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>BW-HVE [13]</td>
<td>$p_1p_2$</td>
<td>$O(\ell)$</td>
<td>$(2/\ell + 1)G + 1 GT$</td>
<td>$(2/\ell + 1)G$</td>
<td>$(2/\ell + 1)p$</td>
<td>S</td>
<td>No</td>
</tr>
<tr>
<td>KSW-HVE_E [28]</td>
<td>$p_1p_2p_3$</td>
<td>$O(\ell)$</td>
<td>$(2/\ell + 1)G + 1 GT$</td>
<td>$(2/\ell + 1)G$</td>
<td>$(2/\ell + 1)p$</td>
<td>S</td>
<td>No</td>
</tr>
<tr>
<td>SW-HVE [44]$^\dagger$</td>
<td>$p_1p_2p_3$</td>
<td>$O(\ell)$</td>
<td>$(\ell + 3)G + 1 GT$</td>
<td>$(\ell + 3)G$</td>
<td>$(\ell + 3)p$</td>
<td>S</td>
<td>No</td>
</tr>
<tr>
<td>IP-HVE [27]$^\ddagger$</td>
<td>$p_1$</td>
<td>$O(\ell)$</td>
<td>$(2/\ell + 1)G + 1 GT$</td>
<td>$(2/\ell + 1)G$</td>
<td>$(2/\ell + 1)p$</td>
<td>S</td>
<td>No</td>
</tr>
<tr>
<td>OT-HVE [36]$^\ddagger$</td>
<td>$p_1$</td>
<td>$O(\ell)$</td>
<td>$(2/\ell + 1)G + 1 GT$</td>
<td>$(2/\ell + 1)G$</td>
<td>$(2/\ell + 1)p$</td>
<td>S</td>
<td>No</td>
</tr>
<tr>
<td>PL-HVE [40]</td>
<td>$p_1p_2p_3$</td>
<td>$O(\ell)$</td>
<td>$(\ell + 3)G + 1 GT$</td>
<td>$(\ell + 3)G$</td>
<td>$(\ell + 3)p$</td>
<td>S</td>
<td>No</td>
</tr>
<tr>
<td>LL-HVE-1 [29]$^\ddagger$</td>
<td>$p_1$</td>
<td>$O(\ell)$</td>
<td>$(\ell + 3)G + 1 GT$</td>
<td>$(\ell + 3)G$</td>
<td>$(\ell + 3)p$</td>
<td>S</td>
<td>No</td>
</tr>
<tr>
<td>LL-HVE-2 [29]$^\ddagger$</td>
<td>$p_1$</td>
<td>$O(\ell)$</td>
<td>$(\ell + 3)G + 1 GT$</td>
<td>$(\ell + 3)G$</td>
<td>$(\ell + 3)p$</td>
<td>S</td>
<td>No</td>
</tr>
<tr>
<td>LOS-HVE [30]$^\ddagger$</td>
<td>$p_1$</td>
<td>$O(\ell)$</td>
<td>$(\ell + 3)G + 1 GT$</td>
<td>$(\ell + 3)G$</td>
<td>$(\ell + 3)p$</td>
<td>S</td>
<td>No</td>
</tr>
<tr>
<td>DIP-HVE [18]$^\ddagger$</td>
<td>$p_1p_2p_3p_4$</td>
<td>$O(\ell)$</td>
<td>$(\ell + 3)G + 1 GT$</td>
<td>$(\ell + 3)G$</td>
<td>$(\ell + 3)p$</td>
<td>S</td>
<td>No</td>
</tr>
<tr>
<td>OT-HVE [35]$^\ddagger$</td>
<td>$p_1$</td>
<td>$O(\ell)$</td>
<td>$(\ell + 3)G + 1 GT$</td>
<td>$(\ell + 3)G$</td>
<td>$(\ell + 3)p$</td>
<td>S</td>
<td>No</td>
</tr>
<tr>
<td>OT-HVE-1 [34]$^\ddagger$</td>
<td>$p_1$</td>
<td>$O(\ell)$</td>
<td>$(\ell + 3)G + 1 GT$</td>
<td>$(\ell + 3)G$</td>
<td>$(\ell + 3)p$</td>
<td>S</td>
<td>No</td>
</tr>
<tr>
<td>OT-HVE-2 [34]$^\ddagger$</td>
<td>$p_1$</td>
<td>$O(\ell)$</td>
<td>$(\ell + 3)G + 1 GT$</td>
<td>$(\ell + 3)G$</td>
<td>$(\ell + 3)p$</td>
<td>S</td>
<td>No</td>
</tr>
<tr>
<td>Our HVE</td>
<td>$p_1$</td>
<td>$O(\ell)$</td>
<td>$(\ell + 3)G + 1 GT$</td>
<td>$(\ell + 3)G$</td>
<td>$(\ell + 3)p$</td>
<td>S</td>
<td>No</td>
</tr>
</tbody>
</table>

$^\ddagger$ In [27,18], the components of vectors are defined over $\{0,1\}$ in encryption and $\{0,1,*\}$ in token generation.

$^\ddagger$ Weakly attribute-hiding.

$^\ddagger$ [44,36,30,18,35,34] provide delegation mechanism.

$^\ddagger$ The second construction of [29] is based upon asymmetric bilinear maps.
the token generation phase. For the vector $\vec{\sigma} \in \mathcal{I}'$, let $S(\vec{\sigma})$ be the set of indexes $i$ such that $\sigma_i$ is not a wildcard character. We define a predicate function $P_i : \mathcal{I}' \times \mathcal{I}' \to \{0, 1\}$ as follows:

$$P_i(\vec{\sigma} \in \mathcal{I}', \vec{x} \in \mathcal{I}') = \begin{cases} 1 & \text{if for all } i \in S(\vec{\sigma}), x_i = \sigma_i, \\ 0 & \text{otherwise.} \end{cases}$$

In HVE, the sender encrypts a pair $(\vec{x}, M) \in \mathcal{I}' \times \mathcal{M}$ where $\mathcal{M}$ is a message space, and the receiver releases a token for a vector $\vec{\sigma}$. Then, the token can decrypt a ciphertext if and only if $P_i(\vec{\sigma}, \vec{x}) = 1$. With the predicate function described above, we formally define HVE by the following four algorithms:

- **Setup** $(k, \ell)$ takes as input a security parameter $k$ and a dimension $\ell$ of vector consisting of attributes. It outputs a public key $PK$ and a secret key $SK$.
- **Encrypt** $(PK, (\vec{x}, M))$ takes as input the public key $PK$, a vector $\vec{x} \in \mathcal{I}'$ of attributes, and a message $M \in \mathcal{M}$. It outputs a ciphertext $CT$.
- **GenToken** $(SK, \vec{\sigma})$ takes as input the secret key $SK$ and a vector $\vec{\sigma} \in \mathcal{I}'$ of attributes. It outputs a token $TK_{\vec{\sigma}}$.
- **Decrypt** $(TK_{\vec{\sigma}}, CT)$ takes as input the token $TK_{\vec{\sigma}}$ and a ciphertext $CT$. It outputs a message $M$ if $P_i(\vec{\sigma}, \vec{x}) = 1$ and outputs $\perp$ otherwise.

**Correctness.** For all $\vec{x} \in \mathcal{I}'$, all $\vec{\sigma} \in \mathcal{I}'$, and all $M \in \mathcal{M}$, let $(PK, SK) \xleftarrow{\$} \text{Setup}(k, \ell)$, CT $\xleftarrow{\$} \text{Encrypt}(PK, (\vec{x}, M))$, and $TK_{\vec{\sigma}} \xleftarrow{\$} \text{GenToken}(SK, \vec{\sigma})$. If we have $P_i(\vec{\sigma}, \vec{x}) = 1$, $M = \text{Decrypt}(TK_{\vec{\sigma}}, CT)$, otherwise $Pr[\perp = \text{Decrypt}(TK_{\vec{\sigma}}, CT)] > 1 - \epsilon(k)$ where $\epsilon(k)$ is a negligible function.

In the above definition, the message $M$ is a real message that the encryptor wishes to send to recipients. In practice, $M$ can also be used as a symmetric key with which authenticated encryption works to check the validity of the ciphertext.

### 2.2. Security for hidden vector encryption

Following [13,28,11,35], we describe the security for HVE that captures the intuition that the ciphertext $CT$ reveals no information about $(\vec{x}, M)$. The security is defined in the following interaction between an adversary $A$ and a challenger $C$, where $i$ is given to $A$.

**Setup:** $C$ runs the setup algorithm to obtain the public key $PK$ and the secret key $SK$. It gives $PK$ to $A$.

**Query Phase 1:** $A$ adaptively issues a polynomial number of token queries for vectors, $\vec{\sigma}_i$. $C$ responds with the corresponding tokens $TK_{\vec{\sigma}_i} \xleftarrow{\$} \text{GenToken}(SK, \vec{\sigma}_i)$.

**Challenge:** $A$ outputs $\vec{x}_0$, $\vec{x}_1$ and two messages $M_0$, $M_1$ under the two constraints that:

- $P_i(\vec{\sigma}_0, \vec{x}_0) = P_i(\vec{\sigma}_1, \vec{x}_1) = 0$ for all queried vectors, $\vec{\sigma}_i$.
- $P_i(\vec{\sigma}_i, \vec{x}_0) = P_i(\vec{\sigma}_i, \vec{x}_1) = 1$ for at least one queried vector, $\vec{\sigma}_i$, in which case $M_0 = M_1$.

$C$ flips a coin $b \in \{0, 1\}$ and gives $CT^* \leftarrow \text{Encrypt}(PK, (\vec{x}_b, M_b))$ to $A$.

**Query Phase 2:** $A$ adaptively issues additional token queries for vectors, $\vec{\sigma}_i$, subject to the restriction in **Challenge** above. $C$ responds with the corresponding tokens $TK_{\vec{\sigma}_i} \xleftarrow{\$} \text{GenToken}(SK, \vec{\sigma}_i)$.

**Guess:** $A$ outputs a guess $b' \in \{0, 1\}$. $A$ wins if $b' = b$.

The advantage of the adversary $A$ in breaking the HVE scheme is defined as $\text{Adv}_{A}^{\text{HVE}} = |Pr[b' = b] - 1/2|$.

**Definition 1.** We say that a Hidden Vector Encryption (HVE) scheme is (attribute-hiding) secure if for any polynomial time adversaries $A$ attacking the HVE scheme, the advantage $\text{Adv}_{A}^{\text{HVE}}$ is negligible.

### 2.3. Bilinear maps and complexity assumptions

**Bilinear Maps:** We adopt the notation in [9,6]. Let $\mathbb{G}$ and $\mathbb{G}_T$ be two (multiplicative) cyclic groups of prime order $p$. We assume that $g$ is a generator of $\mathbb{G}$. Let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a function that has the following properties:

1. Bilinear: for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}$, we have $e(u^a, v^b) = e(u, v)^{ab}$.
2. Non-degenerate: $e(g, g) \neq 1$.
3. Computable: there is an efficient algorithm to compute the map $e$.

Then, we say that the map $e$ is a bilinear map in $\mathbb{G}$. Note that $e(\cdot)$ is symmetric since $e(g^a, g^b) = e(g, g)^{ab} = e(g^b, g^a)$.

**The Decisional Bilinear Diffie-Hellman (DBDH) Problem:** The DBDH problem [9] is defined as follows: given $(g, g^a, g^b, g^c, g^Z) \in \mathbb{G}^5 \times \mathbb{G}_T$ as input, determine whether $Z = e(g^a, g^b)^{g^c} = Z$ is random in $\mathbb{G}_T$.

**The Decision Linear (DLIN) Problem:** The DLIN problem [8] was originally stated as follows: given $(g, g^a, g^b, g^{a+b}, g^{a+2b}, Z) \in \mathbb{G}^5 \times \mathbb{G}_T$ as input, determine whether $Z = g^{a+b} = Z$ is random in $\mathbb{G}$. We consider an equivalently modified version such as: given $(g, g^a, g^b, g^{a+b}, g^{a+2b}, Z) \in \mathbb{G}^5$ as input, determine whether $Z = g^{a+b} = Z$ is random in $\mathbb{G}$. This was already used in [12].
2. We say that the (DBDH, DLIN) assumption holds in \( G \) if the advantage of any polynomial time algorithm in solving the (DBDH, DLIN) problem is negligible.

**Remark 1.** In the groups equipped with symmetric bilinear maps \( e : G \times G \rightarrow G_T \), we can see that the DBDH assumption is weaker than the DLIN assumption. To show this, let us assume that there is an adversary to solve DBDH problem. If an instance \( (g, g^a, g^{a^2}, g^{a^3}, g^Z) \) as a DLIN problem is given, we first compute \( Z' = e(g^a, Z)/e(g^{a^2}, g^{a^3}) \) and next we give an instance \( (g, g^a, g^{a^2}, g^{a^3}, Z') \) of a DBDH problem to the adversary. Clearly, if \( Z' = e(g, g^{a^2})Z_{a^3} \), then \( Z = g^{a^3}/g^Z \), and otherwise, \( Z \) is random. It seems that the opposite direction does not hold, and also the relation between the \( n \)-DLIN assumption \( (n > 2) \) (which is also weaker than the DLIN assumption) and the DBDH assumption is not clear.

### 3. Fully secure HVE scheme

#### 3.1. Construction

Let \( G \) and \( G_T \) be groups of prime order \( p \), and let \( e : G \times G \rightarrow G_T \) be the bilinear map. We assume that each attribute \( x_i \) belongs to \( \Sigma = \mathbb{Z}_p^* \) and our scheme deals with \( \ell \)-dimensional vector \( \bar{x} = (x_1, \ldots, x_\ell) \in \Sigma^\ell \). If necessary, we can extend our construction to handle arbitrary attributes in \( \{0, 1\}^* \) by first hashing each \( x_i \) using a collision-resistant hash function \( H : \{0, 1\}^* \rightarrow \mathbb{Z}_p^* \). Note that \( I = \mathbb{Z}_p \cup \{\ast\} \).

**Setup (\( \lambda \)):** Given a security parameter \( \lambda \in \mathbb{Z}_p^* \), the setup algorithm runs \( G(\lambda) \) to obtain a tuple \( (p, G, G_T, e) \). The algorithm picks a random generator \( g \in G \), random elements \( \gamma, \tau, \tau_i, \gamma_i \in G \), random exponents \( \Omega, \Omega_i, \{\gamma_i\}_{i=1}^\ell, \{\tau_i\}_{i=1}^\ell \), \( \{f_i\}_{i=1}^\ell \) in \( G_T \). It obtains \( \Omega \neq 0 \in \mathbb{Z}_p^* \) such that \( \Omega \cdot \Omega_i + \Omega_i \cdot \Omega_i = \lambda \Omega \). If \( \Omega = 0 \), the algorithm tries again with new random exponents. It sets \( \Omega \cdot \Omega_i + \Omega_i \cdot \Omega_i = \beta \in \mathbb{Z}_p^* \). The algorithm sets

\[
W_1 = g^{\delta_0}, \quad W_2 = g^{\delta_1}, \quad F_1 = g^{\tau_1}, \quad F_2 = g^{\tau_2}, \quad F_3 = g^{\tau_3},
\]

\[
Y_i = g^{\delta_i} (i = 1, \ldots, \ell), \quad g_2 = g^{\tau_1}, \quad g_3 = g^{\tau_2}, \quad g_4 = g^{\tau_3}, \quad A = e(g_1, g_2).
\]

The public key PK (along with the description of \( (p, G, G_T, e) \)) and the secret key msk are set to be

\[
PK = \left( g, \gamma, \Omega, \tau, \gamma_i, \tau_i, \gamma_i, \Omega_i, \{\gamma_i\}_{i=1}^\ell, \{\tau_i\}_{i=1}^\ell, \{f_i\}_{i=1}^\ell, A \right) \in G^{4\ell+11} \times G_T,
\]

\[
SK = \left( \Omega, \gamma_i, \tau_i, \gamma_i, \{f_i\}_{i=1}^\ell, \Omega_i \right) \in \mathbb{Z}_p^{4\ell+9} \times G.
\]

**Encrypt (PK, (\( \bar{x}, M \)):** Let \( \bar{x} = (x_1, \ldots, x_\ell) \in \Sigma^\ell \). To encrypt a message \( M \in \mathcal{M} \subseteq G_T \) and the vector \( \bar{x} \) under the public key PK, the encryption algorithm picks random exponents \( s_1, s_2, s_3, \{\text{tag}_{i,j}\}_{i=1}^\ell \in \mathbb{Z}_p \) and computes the ciphertext \( CT = (C_1, \ldots, C_6, \{C_{6,i}\}_{i=1}^\ell, C_8, C_9, \{\text{tag}_{i,j}\}_{i=1}^\ell) \in G^{2\ell+6} \times G_T \times \mathbb{Z}_p^{\ell} \) as follows:

\[
C_1 = W_1^{s_1}, \quad C_2 = W_1^{s_2}, \quad C_3 = g_2^{s_1}, \quad C_4 = g_3^{s_2}, \quad C_5 = g_4^{s_3}, \quad C_6 = g^{\delta_0}\gamma\Omega_i\gamma_i.
\]

\[
\{C_{6,i} = (u_i g_2^{\delta_i}v^{\tau_i})^{s_1}\}, \quad C_{7,i} = (\tau_i v^{\tau_i})^{s_2}\}, \quad C_8 = g_4^{\delta_0}, \quad C_9 = A^{s_1} M.
\]

**GenToken (SK, \( \bar{\sigma} \)):** Let \( \bar{\sigma} = (\sigma_1, \sigma_2, \ldots, \sigma_\ell) \in \mathbb{T}^\ell \). Let \( S(\bar{\sigma}) \) be the set of all indexes \( i \) such that \( \sigma_i \neq \ast \). To generate a token TK_{\bar{\sigma}} for the vector \( \bar{\sigma} \), the token generation algorithm picks random exponents \( r_1, r_2, r_3, r_4, \text{tag}_\ast \in \mathbb{Z}_p \) and obtains \( r_5 \in \mathbb{Z}_p \) such that \( (S(\bar{\sigma})(\bar{\ast})(r_5))_3 = \gamma_5 \). The algorithm computes the token \( TK_{\bar{\sigma}} = (K_1, \ldots, K_9, \text{tag}_\ast) \in G^{9 \times \mathbb{Z}_p} \) as follows:

\[
K_1 = g_1^{\tau_1}, \quad K_2 = g_2^{\tau_2}, \quad K_3 = g_3^{\tau_3}, \quad K_4 = g_4^{\tau_4}, \quad K_5 = K_1^{\tau_2}K_2^{\tau_3}K_3^{\tau_4}K_4^{\tau_5}, \quad K_6 = \left( \prod_{i \in S(\bar{\sigma})}(u_i g_2^{\delta_i}v^{\tau_i})^{r_3} \right)^{r_4}, \quad K_7 = g_5^{\tau_2}, \quad K_8 = g^{\tau_4}, \quad K_9 = g^{\tau_5}.
\]

**Decrypt (CT, TK_{\bar{\sigma}}):** To decrypt a ciphertext \( CT = (C_1, \ldots, C_6, \{C_{6,i}\}_{i=1}^\ell, C_8, C_9, \{\text{tag}_{i,j}\}_{i=1}^\ell) \) using a private key \( TK_{\bar{\sigma}} = (K_1, \ldots, K_9, \text{tag}_\ast) \), the decryption algorithm sets

\[
C_9 = \sum_{i : S(\bar{\sigma}) \cap i = \emptyset} \text{tag}_{i,j}\}, \quad C_7 = \sum_{i : S(\bar{\sigma}) \cap i = \emptyset} \text{tag}_{i,j}\},
\]

If \( \text{tag}_\ast \neq \text{tag}_{\ast} \), the decryption algorithm proceeds as follows:

1. Compute \( A_1 = e(C_1, K_1) \cdot e(C_2, K_2) e(C_3, K_3) \cdot e(C_4, K_4) \cdot e(C_5, K_5) \).
2. Compute

\[
A_2 = e(C_9, K_7) \cdot e(C_7, K_8) e(C_5, K_6) \cdot e(C_8, K_9) \}^{1/(\text{tag}_{\ast} - \text{tag}_{\ast})}.
\]
3. Output $M = C_9 \cdot A_1 \cdot A_2$.

**Performance:** Note that a token consists of 9 group elements in $G$ plus 1 group element in $\mathbb{Z}_p$, and the decryption algorithm requires 9 pairing operations. These two efficiency factors are independent of the dimension $\ell$ of the attribute vectors.

### 3.2. Correctness

We first check that $A_1 = \lambda^{-\omega} \cdot \pi g^{-\omega_4} \cdot \pi (g, \varphi)^{-\omega_4}$ as follows:

$$
\frac{e(C_1, K_1) \cdot e(C_2, K_2)}{e(C_3, K_3) \cdot e(C_4, K_4) \cdot e(C_5, K_5)} = \frac{e(g^{\omega_1}, g^{\omega_1}, g^{\omega_1}) \cdot e(g^{\omega_2}, g^{\omega_2}, g^{\omega_2})}{e(g^{\omega_1}, g^{\omega_1}, g^{\omega_1}) \cdot e(g^{\omega_1}, g^{\omega_1}, g^{\omega_1})} = 1
$$

Next, notice that

$$
C_6 = \left( \prod_{i \in S(\tilde{\sigma})} u_i h_i^{x_i} \cdot \nu^{\omega_4} \right)^{y_2} \cdot g^{\sum_{i \in S(\tilde{\sigma})} y_i x_i}, \quad C_7 = \left( \prod_{i \in S(\tilde{\sigma})} \tau_i \cdot \nu^{\omega_4} \right)^{y_2}.
$$

Then, if $P_i(\sigma, \tilde{x}) = 1$, (i.e., $\sigma_i = x_i$ for all $i \in S(\tilde{\sigma})$), we can see that $A_2 = e(g, \nu)^{y_2} \cdot e(g, \varphi)^{y_2}$ by the following computation:

$$
\frac{e(C_6, K_7) \cdot e(C_7, K_9)}{e(C_5, K_6) \cdot e(C_8, K_9)} = e\left( \prod_{i \in S(\tilde{\sigma})} u_i h_i^{x_i} \cdot \nu^{\omega_4} \right)^{y_2} \cdot e\left( \prod_{i \in S(\tilde{\sigma})} \tau_i \cdot \nu^{\omega_4} \right)^{y_2} = e\left( g^{y_2}, \prod_{i \in S(\tilde{\sigma})} \tau_i \cdot \nu^{\omega_4} \right)^{y_2} \cdot e\left( g^{y_2}, g^{x_i} \right)^{y_2} = e\left( \nu^{y_2}, \nu^{y_2} \right)^{y_2} = 1
$$

Finally, the message $M$ is correctly recovered as

$$
C_9 \cdot A_1 \cdot A_2 = A^+ \cdot M \cdot A^{-\omega} \cdot e(g, \nu)^{-\omega_2} \cdot e(g, \varphi)^{-\omega_2} \cdot e(g, \nu)^{y_2} \cdot e(g, \varphi)^{y_2} = M.
$$

Otherwise, if $P_i(\sigma, \tilde{x}) = 0$, this means that there is at least one component $\sigma_i \neq x_i$ for some $i \in S(\tilde{\sigma})$. Let $D$ be the set of indexes $i \in S(\tilde{\sigma})$ such that $\sigma_i \neq x_i$. In this case, the computation above becomes

$$
\frac{e(C_6, K_7) \cdot e(C_7, K_9)}{e(C_5, K_6) \cdot e(C_8, K_9)} = e\left( g^{y_2}, \prod_{i \in D} \tau_i \cdot \nu^{\omega_4} \right)^{y_2} \cdot e\left( g^{y_2}, \nu^{\omega_4}, \nu^{\omega_4} \right)^{y_2} = e\left( g^{y_2}, g^{\nu^{\omega_4}} \right)^{y_2} \cdot e\left( g^{y_2}, \nu^{\omega_4} \right)^{y_2}.
$$

Thus, the final output becomes $M$ if $\Sigma_{i \in D} (\log_2 h_i) / (x_i - \sigma_i) = 0$ in $\mathbb{Z}_p$. However, it is computationally hard to find pairs $(x_i, \sigma_i)$ for $i \in D$ for which such an equality holds. In fact, the probability of a false positive is at most $1/p$ in each decryption.

### 3.3. Fully secure anonymous IBE scheme

Any HVE scheme implies an anonymous IBE scheme if the vectors $\bar{x}$ and $\bar{\sigma}$ are limited to one dimension. Thus, our HVE scheme provides a new anonymous IBE scheme that is fully secure under standard assumptions such as the DLIN and DBDH assumptions. Prior to our result, several works [9,20,12,42,16,2,17] (including all previous HVE and IPE schemes) have been proposed, but until now there were few anonymous IBE schemes [20,30,35] that achieve full security without using random
oracles. Our new scheme is another example, but is fully secure under the standard assumptions. Compared to [35], which has comparable security, our construction is more efficient in all efficiency respects. Table 2 in Section 5 presents the result by simply assigning the dimension to 1. To demonstrate security of any anonymous IBE scheme, testing for weak attribute hiding is sufficient since two target pairs ($D_b, M_b$) for $b = 0, 1$ should be equal as long as at least one matching query is asked. Thus, security of our new anonymous IBE scheme is straightforwardly obtained from the proof of Case 1 (defined in the next section) where only non-matching token queries are permitted.4

The hierarchical extension of our anonymous IBE scheme can not be realized due to the relation $(\Sigma_{i,j} y_i)R_5 = yR_5$. Such a relation plays a key role in the elimination of blinding factors $Y_i = g^{e\alpha_i}$. In constructing our anonymous IBE scheme, only $y_1$ is necessary for one identity and choosing $r_3$ and $r$, satisfying the relation $y_1r_3 = yR_5$ can be easily done by the key generation center who knows the exponents $(y_i)$ and $\gamma$. However, if a key owner (as a parent) wants to generate private keys for its descendants of depth 2, the parent has to select exponents $r_3$ and $r$ such that $y_2r_3 = yR_5$ without knowing $y_2$ and $\gamma$, which is computationally infeasible. Thus, it is still an open problem to construct an anonymous Hierarchical IBE scheme that is fully secure under standard assumptions.

4. Security proof

4.1. Semi-functional algorithms

We now describe the semi-functional ciphertexts and tokens. Their main purpose is to define the structures that will be used in our proof.

4.1.1. Semi-functional ciphertexts

The algorithm first runs the encryption algorithm to generate a normal ciphertext $CT = \left( C_1, \ldots, \{C_{6i}, C_{7i}\}\right)_{i=1}^\ell$, for a vector $X$ and a message $M$. The algorithm selects a random exponent $x \in \mathbb{Z}_p$ and sets

\[
C_1 = C'_1 \cdot g^{i\lambda x}, \quad C_2 = C'_2 \cdot g^{-\lambda x}, \quad C_3 = C'_3, \quad C_4 = C'_4 \cdot g^{(i\lambda - \lambda)2x}, \quad C_5 = C'_5, \quad \left\{ C_{6i} = C'_6, \quad C_{7i} = C'_7 \right\}_{i=1}^\ell, \quad C_8 = C'_8, \quad C_9 = C'_9.
\]

The semi-functional ciphertext is $CT_{sf} = \left( C_1, \ldots, \{C_{6i}, C_{7i}\}\right)_{i=1}^\ell, C_8, C_9, \{tag_{ci}\}_{i=1}^\ell$. If one tries to decrypt the semi-functional ciphertext with a normal token $TK_{\tilde{g}}$ for $\sigma$, then the decryption would be correctly performed. This stems from the fact that

\[
\frac{e(g^{i\lambda x}, K_1) \cdot e(g^{-\lambda x}, K_2)}{e(g^{(i\lambda - \lambda)2x}, K_4)} = \frac{e(g^{i\lambda x} \cdot g^\lambda, g^{\lambda \tau_1}, K_2) \cdot e(g^{-\lambda x} \cdot g^{\lambda \tau_2}, g^{\lambda \tau_2})}{e(g^{(i\lambda - \lambda)2x} \cdot g^{\lambda \tau_1}, g^{\lambda \tau_2})} = 1,
\]

where $K_1, K_2, K_4$ are components of the normal token.

4.1.2. Semi-functional tokens

The algorithm first runs the token generation algorithm to generate a normal token $TK_{\tilde{g}} = (K'_1, \ldots, K'_y, \{tag\})$ for a vector $\tilde{g}$. Next the algorithm picks a random exponent $\lambda \in \mathbb{Z}_p$ and sets

\[
K_1 = K'_1 \cdot g^{-\lambda x}, \quad K_2 = K'_2 \cdot g^{\lambda x}, \quad K_3 = K'_3, \quad K_4 = K'_4, \quad K_5 = K'_5 \cdot g^{(i\lambda - j\lambda)2x}, \quad K_6 = K'_6, \quad K_7 = K'_7, \quad K_8 = K'_8, \quad K_9 = K'_9.
\]

Then, the semi-functional token is $TK'_{\tilde{g}} = (K'_1, \ldots, K'_9, \{tag\})$. Note that the element $K_5$ becomes $K_5 = K'_5 \cdot K'_2 \cdot K_4 \cdot K^\ell \cdot \mu x \cdot \mu x$. If one tries to decrypt a normal ciphertext encrypted under $\tilde{g}$ with the semi-functional token $TK'_{\tilde{g}}$, the decryption would be also correctly performed. This can be checked from the fact that

\[
\frac{e(C_1, g^{-\lambda x}, K'_1) \cdot e(C_2, g^{\lambda x}, K'_2)}{e(C_3, g^{(i\lambda - j\lambda)2x}, K'_4) \cdot e(C_5, K'_5 \cdot g^{(i\lambda - j\lambda)2x}, K'_6)} = 1,
\]

where $C_1, C_2, C_3$ are components of the normal ciphertext.

We note that when a semi-functional token is used to decrypt a semi-functional ciphertext, the semi-functional components in two parts will be computed as follows:

\[
\frac{e(C_1, g^{i\lambda x}, K'_1 \cdot g^{-\lambda x}) \cdot e(C_2, g^{-\lambda x}, K'_2 \cdot g^{\lambda x})}{e(C_3, g^{(i\lambda - j\lambda)2x}, K'_4) \cdot e(C_5, K'_5 \cdot g^{(i\lambda - j\lambda)2x}, K'_6)} = e(g, g)^{-(\lambda x + \lambda x)2x} = e(g, g)^{-2\lambda x},
\]

which is not equal to 1 in $G_1$.

4 The condition should be different when considering security of anonymous Hierarchical IBE, where both matching and non-matching token queries are justified for two identity vectors upon which an adversary wants to challenge.
4.2. Proof of security

In the security game defined in Section 2, the adversary \( A \) outputs two vectors \( \tilde{x}_0 = (x_{0,1}, \ldots, x_{0,s}) \) and \( \tilde{x}_1 = (x_{1,1}, \ldots, x_{1,s}) \in \Sigma^s \) and two messages \( M_0, M_1 \in \mathcal{M} \) as its challenge. The goal of \( A \) is to decide which one of the two pairs \((\tilde{x}_0, M_0)\) and \((\tilde{x}_1, M_1)\) is associated with the challenge ciphertext. All tokens will be normal and the challenge ciphertext will also be normal. This is the real security game \( \text{Game}_{\text{Real}} \). Under the rules of the security game, \( A \) that makes at most \( q \) token queries will behave in one of two different ways:

**Case 1** \( A \) will make token queries for vectors \( \tilde{\sigma}_i \) such that \( P_i(\tilde{\sigma}_i, \tilde{x}_0) = P_i(\tilde{\sigma}_i, \tilde{x}_1) = 0 \) for all \( i = 1, \ldots, q \).

**Case 2** \( A \) will make token queries for vectors \( \tilde{\sigma}_i \) such that \( P_i(\tilde{\sigma}_i, \tilde{x}_0) = P_i(\tilde{\sigma}_i, \tilde{x}_1) = 1 \) for at least one \( i \in \{1, \ldots, q\} \). In this case, it should be the case that \( M_0 = M_1 \).

In our security proof, the simulator needs to guess which case it will be in by flipping a coin. If the guess is wrong, the simulator aborts the simulation and outputs a random bit as its answer. Since the simulator’s guess will be independent of which case \( A \) behaves in, the simulation is able to proceed with probability \( 1/2 \). Depending on the case by guess, the simulator prepares its simulation differently. We describe the simulator’s strategy in two cases.

**Case 1:** (Proof idea) We first give an idea behind the security proof in Case 1. Since \( A \) cannot make any matching token query, we can adapt a similar proof strategy to one in the Waters’ original dual system encryption [46]. That is, we create a sequence of hybrid games, where the challenge ciphertext and all tokens are changed into semi-functional ones, and we can then change the message \( M_b \) for a random bit \( b \in \{0, 1\} \) into a random message. This is the same as in [46], but the difference is that we create an additional sequence of hybrid games, based on the result of randomizing \( M_b \), in order to change each component of the vector \( \tilde{x}_0 \) into a random one. During these two sequences of hybrid games, tag values are crucially used to solve the paradox that happens inevitably when proving full security.

The simulator considers a sequence of hybrid games as follows:

- **Game\textsubscript{Real}**: This is the actual HVE security game in Case 1. All tokens will be normal and the challenge ciphertext will be a normal challenge ciphertext on a pair \((\tilde{x}_0, M_b)\), where \( b \in \{0,1\} \) is a random bit.
- **Game\textsubscript{0}**: All tokens will be normal, but the challenge ciphertext will be a semi-functional ciphertext on a pair \((\tilde{x}_0, M_b)\).
  
  ...  
  
- **Game\textsubscript{1}**: The first \( k \) token queries will return semi-functional tokens, and the rest of the tokens will be normal. The challenge ciphertext will be a semi-functional ciphertext on a pair \((\tilde{x}_0, M_b)\).
  
  ...  
  
- **Game\textsubscript{2}**: All tokens will be semi-functional, and the challenge ciphertext will be a semi-functional ciphertext on a pair \((\tilde{x}_0, M_b)\).
- **Game\textsubscript{3}**: All tokens will be semi-functional, and the challenge ciphertext will be a semi-functional ciphertext on a pair \((\tilde{x}_1, R)\), where \( R \) is a random message from \( \mathcal{M} \).
- **Game\textsubscript{4}**: All tokens will be semi-functional, and the challenge ciphertext will be a semi-functional ciphertext on a pair \((\{r_1, x_{1,2}, \ldots, x_{1,s}\}, R)\), where \( r_1 \) is a random element from \( \Sigma \).
  
  ...  
  
- **Game\textsubscript{5}**: All tokens will be semi-functional, and the challenge ciphertext will be a semi-functional ciphertext on a pair \((\{r_1, r_2, \ldots, r_k\}, R)\), where \( r_i \) for \( i = 1, \ldots, k \) are random elements from \( \Sigma \).

In **Game\textsubscript{Real}**, the normal challenge ciphertext corresponding to \((\tilde{x}_0, M_b)\) is given to the adversary. On the other hand, in **Game\textsubscript{2}**, the challenge ciphertext given to the adversary is a semi-functional ciphertext corresponding to \((\{r_1, \ldots, r_k\}, R)\) that leaks no information about \((\tilde{x}_0, M_b)\). We will show that no polynomial time adversary is able to distinguish between **Game\textsubscript{Real}** and **Game\textsubscript{2}**, by proving that the transitions between the sequence of games above are all computationally indistinguishable under the DLIN and DBDH assumptions.

**Lemma 1.** Suppose that the DLIN assumption holds. Then no polynomial time adversary \( A \) can distinguish between **Game\textsubscript{Real}** and **Game\textsubscript{1}** with non-negligible advantage.
Proof. Suppose that there exists an adversary $A$ which can attack our HVE scheme with non-negligible advantage $\epsilon$. We describe an algorithm $B$ which uses $A$ to solve the DLIN problem with advantage $\epsilon$. On input $(g, g^{z_1}, g^{z_2}, g^{z_3}, Z) \in \mathbb{G}_p^3$, $B$'s goal is to output $1$ if $Z = g^{z_2(z_1 + z_4)}$ and $0$ otherwise. $B$ interacts with $A$ as follows:

**Setup** $B$ selects random exponents $\Omega, \gamma, \{y_i, \mu_i, t_i, z_i\}_{i=1}^\ell$, $\{\gamma_i\}_{i=1}^\ell$, $\{w_i, \phi_i, \phi_{i}^{2}\}_{i=1}^\ell$, $\{f_i\}_{i=1}^\ell$ in $\mathbb{Z}_p$, such that $w_1 \phi_1 + w_2 \phi_2 = \Omega$. $B$ sets

$$W_1 = (g^{z_3})^{z_1}(g^{z_1})^{w_1}, \quad W_2 = (g^{z_2})^{-\phi_2}(g^{z_1})^{w_2},$$

$$F_1 = (g^{z_2})^{\phi_1}, \quad F_2 = (g^{z_2})^{-\phi_1}, \quad F_3 = (g^{z_2})^{\phi_1 + \phi_2} g^{f_3},$$

$$g_2 = g^{z_1}, \quad g_3 = (g^{z_2})^{\phi_1 + \phi_2} (g^{z_1})^{\phi_1 + w_2} g^{f_3}, \quad g_4 = g^{z_2}.$$  

$$Y_i = g^{z_i}, \quad u_i = g^{y_i}, \quad h_i = g^{\mu_i}, \quad \tau_i = g^{z_i} \quad (i = 1, \ldots, \ell).$$

$$v = g^{z_2}, \quad \varphi = g^{z_1}, \quad A = e(g^{z_1}, g^{z_2}).$$

$B$ (implicitly) sets

$$\tilde{w}_1 = \delta_1 z_2 + w_1 z_1, \quad \tilde{w}_2 = -\delta_1 z_2 + w_2 z_1, \quad \tilde{f}_1 = \delta_1 z_2 + f_1, \quad \tilde{f}_2 = -\delta_1 z_2 + f_2$$

$$\tilde{f}_3 = (\delta_2 \phi_1 - \delta_1 \phi_2) z_2 + f_3, \quad \beta = (\delta_2 \phi_1 - \delta_1 \phi_2) z_2 + (w_1 \phi_1 + w_2 \phi_2) z_1, \quad \alpha = z_1, \quad g_1 = g^{z_1}.$$  

Notice that each public key element is independently and uniformly distributed as in the actual construction. Also, we can see that

$$\tilde{w}_1 \phi_1 + \tilde{w}_2 \phi_2 = (\delta_2 z_2 + w_1 z_1) \phi_1 + (\delta_1 z_2 + w_2 z_1) \phi_2 = (\delta_2 \phi_1 - \delta_1 \phi_2) z_2 + (w_1 \phi_1 + w_2 \phi_2) z_1 = \beta.$$  

**Key Generation Phases 1 and 2** $A$ issues token queries for vectors $\{\tilde{t}, \tilde{r}\}$. For any queried vector $\tilde{t}_i$, it is easy for $B$ to generate a normal token $TK_{\tilde{t}_i}$, since it knows exponents $\Omega, \gamma, \{y_i\}_{i=1}^\ell, \{\delta_1, \phi_i\}_{i=1}^\ell$. It selects random $r_1, r_2, r_3, r_4, r_5$ (subject to the equation $(\sum_{i=\tilde{t}_i}^{\tilde{r}} y_i) r_5 = r_5^\ast$, $\tilde{r}_a \in \mathbb{Z}_p$ and computes a normal token.

**Challenge Ciphertext** $A$ outputs two vectors $x_0^\ast, x_1^\ast$ and two messages $M_0, M_1$. $B$ flips a random coin $b \in \{0, 1\}$ and picks random $s_1, \{\tag{\tilde{t}_i}\}_{i=1}^\ell$ in $\mathbb{Z}_p$. To generate a challenge ciphertext for $(x_0^\ast = (x_0^\ast_1, \ldots, x_0^\ast_\ell), M_b)$, $B$ implicitly sets $s_1 = z_2$ and $s_2 = z_4$. $B$ computes $C_3, C_5, \{C_{6, i}, C_{7, i}\}_{i=1}^\ell, C_8$, and $C_9$ elements as

$$C_3 = g^{r_5 x_0^\ast}, \quad C_5 = g^{r_4 x_0^\ast},$$

$$C_{6,i} = (g^{z_2})^{r_1 x_0^\ast} (g^{z_1})^{y_i (r_2 + r_3 + r_4)} g^{r_i x_0^\ast} = (u_i h_i) (g^{3 x_0^\ast})^{r_i x_0^\ast},$$

$$C_{7,i} = (g^{z_2})^{z_1} (g^{z_1})^{y_i (r_2 + r_3 + r_4)} = (\tau_i) (g^{3 x_0^\ast})^{z_1},$$

$$C_8 = g^{r_5 x_0^\ast} = g_4^{r_5}, \quad C_9 = e(g^{z_1}, g^{z_2}) M_b = e(g_1, g_2)^{r_5} M_b.$$  

Next, $B$ computes $C_1, C_2,$ and $C_4$ elements as follows:

$$C_1 = (g^{z_2})^{r_5} (g^{z_2})^{r_5 z_2},$$

$$C_2 = (g^{z_2})^{r_5} (g^{z_2})^{r_5 z_2},$$

$$C_4 = (g^{z_2})^{r_5} (g^{z_2})^{r_5 z_2} = C_3.$$  

If $Z = g^{z_2(z_1 + z_4)}$, then we have that

$$C_1 = (g^{z_2})^{r_5} (g^{z_2})^{r_5 z_2} (g^{z_2(z_1 + z_4)})^{r_5} = g^{r_5 z_2 + w_1 z_1 z_4} g^{(z_1 + z_2) z_4} = W_1 F_1,$$

$$C_2 = (g^{z_2})^{r_5} (g^{z_2})^{r_5 z_2} = g^{r_5 z_2 + w_1 z_1 z_4} g^{((z_1 + z_2) z_4) + z_2} = W_2 F_2,$$

$$C_4 = (g^{z_2})^{r_5} (g^{z_2})^{r_5 z_2} (g^{z_2(z_1 + z_4)})^{r_5} = g^{(z_1 + z_2) z_4} g^{z_1 + z_2} z_2 z_4 = g_4^{z_1 + z_2} z_4.$$  

In this case, the ciphertext will have the same distribution as a normal ciphertext. Thus, $B$ is playing $\text{Game}_{\text{Real}}$ with $A$. On the other hand, if $Z = g^{z_2(z_1 + z_4)}$, then $A$ is playing $\text{Game}_{\text{Fake}}$ with $B$. For $z_1, z_2, z_4 \in \mathbb{Z}_p^+$, we have

$$W_1 F_1 + W_2 F_2 + C_4.$$
where the exponent $x$ plays the role of $x$. In this case, the ciphertext will have the same distribution as a semi-functional ciphertext. Thus, $B$ is playing Game$_6^1$ with $\mathcal{A}$.

**Guess** $B$ receives a bit $b' \in \{0, 1\}$ and outputs $0$ if $b' = b$.

**Analysis** As mentioned above, if $Z = g^{z_1(\delta_1 + z_4)}$ the challenge ciphertext is distributed exactly as in Game$^\delta_{\text{Real}}$ whereas if $Z = g^{z_2(\delta_1 + z_4)}g^z$ the challenge ciphertext is distributed exactly as in Game$_6^0$. It follows that under the DLIN assumption, these two games are indistinguishable. □

Let $k = 1, \ldots, q$.

**Lemma 2.** Suppose that the DLIN assumption holds. Then no polynomial time adversary $\mathcal{A}$ can distinguish between Game$_{k-1}^1$ and Game$_k^1$ with non-negligible advantage.

**Proof.** Suppose that there exists an adversary $\mathcal{A}$ which can attack our HVE scheme with non-negligible advantage $\varepsilon$. We describe an algorithm $B$ which uses $\mathcal{A}$ to solve the DLIN problem with advantage $\varepsilon$. On input $(g, g^a, \ldots, g^t, Z) \in \mathbb{G}$, $B$ interacts with $\mathcal{A}$ as follows:

**Setup** $B$ selects random exponents $\iota$, $\{A_i, B_i, y_i, \mu_i, t_i, \sigma_i\}_{i=1}^\ell, \gamma, \gamma', \{j_i\}_{i=1}^3, \{W_i, \delta_i, \phi_i\}_{i=1}^3, \{f_i\}_{i=1}^3$ in $\mathbb{Z}_p$, such that $w_1\delta_1 + w_2\delta_2 = \xi$. (Here, we can exclude the unlikely event that $w_1, w_2 = 0$ in $\mathbb{Z}_p$.)

$B$ sets $w_1\phi_1 + w_2\phi_2 = \beta$ and

$W_1 = g^r, \ W_2 = g^{y_2},$

$F_1 = g^h, \ F_2 = g^{h_2}, \ F_3 = (g^{j_1})^{\frac{1}{2}},$

$B_2 = g^g, \ B_3 = g^g, \ B_4 = g^t,$

$Y_1 = g^u, \ u_i = (g^{\iota_i})^{-A}g^{\mu_i}, \ h_i = (g^{\iota_i})^{-B}g^{\mu_i}, \ t_i = g^{\iota_i} \ i = 1, \ldots, \ell),$

$\nu = g^{j_1}g^{j_2}, \ \varphi = g^{j_1}, \ A = e(g^{j_1}, g^{j_2}).$

$B$ sets

$\tilde{\delta}_1 = -w_2z_2 + \tilde{\delta}_1, \ \tilde{\delta}_2 = w_1z_2 + \tilde{\delta}_2, \ \tilde{\phi}_1 = -w_2z_2 + \tilde{\phi}_1, \ \tilde{\phi}_2 = w_1z_2 + \tilde{\phi}_2, \ \tilde{f}_3 = f_3z_1, \ \Omega = z_1, \ g_1 = g^{z_1}.$

Notice that each public key element is independently and uniformly distributed as in the actual construction. Also, we can see that

$w_1\tilde{\delta}_1 + w_2\tilde{\delta}_2 = w_1(-w_2z_2 + \tilde{\delta}_1z_1) + w_2(w_1z_2 + \tilde{\delta}_2z_1) = (w_1\delta_1 + w_2\delta_2)z_1 = z\Omega.$

$w_1\tilde{\phi}_1 + w_2\tilde{\phi}_2 = w_1(-w_2z_2 + \tilde{\phi}_1z_1) + w_2(w_1z_2 + \tilde{\phi}_2z_1) = w_1\phi_1 + w_2\phi_2 = \beta.$

**Key Generation Phases** $\mathcal{A}$ issues token queries for vectors $\{\tilde{\delta}_i\}$. $B$ breaks the token generation phases into three cases. Consider ith query issued by $\mathcal{A}$.

**Case I:** $i > k$.

$B$ generates a normal token for the requested vector $\tilde{\delta}_i$. $B$ picks random exponents $r_1, r_2, r_3, r_4, r_5$ (subject to the equation $(\Sigma_{l \in I(\tilde{\delta}_i, j)}r_l = \gamma r_5)$), $tag_6 \in \mathbb{Z}_p$ and performs the usual token generation procedures. Note that even though $\tilde{\delta}_1, \tilde{\delta}_2, \tilde{\phi}_1, \tilde{\phi}_2, \Omega$, and $f_3$ are unknown to $B$, it is easy to compute a normal token.

**Case II:** $i < k$.

$B$ generates a semi-functional token for the requested vector $\tilde{\delta}_i$. $B$ picks random exponents $r_1, r_2, r_3, r_4, r_5$ (subject to the equation $(\Sigma_{l \in I(\tilde{\delta}_i, j)}r_l = \gamma r_5)$), $tag_6, \lambda \in \mathbb{Z}_p$ and generates the semi-functional token generation procedures. Since $B$ knows exponents $w_1, w_2, f_1, f_2$, it is easy to compute a semi-functional token.

**Case III:** $i = k$.

For the requested vector $\tilde{\delta}_i$, $B$ picks random exponents $r_3, r_4 \in \mathbb{Z}_p$ and sets

$tag_6 = \Sigma_{l \in I(\tilde{\delta}_i)}(A_l + B_l - \sigma_{l,j}).$

where $\sigma_{l,j}$ is a non-wildcard component in $\tilde{\delta}_i$. It (implicitly) sets $\tilde{r}_1 = z_1, \tilde{r}_2 = z_4, \tilde{r}_3 = f_3z_4 + r_3$, and $\tilde{r}_5 = (\Sigma_{l \in I(\tilde{\delta}_i, j)})(f_4z_4 + r_3)/\gamma$. Note that the equation $(\Sigma_{l \in I(\tilde{\delta}_i, j)}\tilde{r}_l = \gamma \tilde{r}_5$ is satisfied. $B$ generates $K_3, K_4, K_6, K_7, K_8$, and $K_9$ elements as follows:
$K_3 = g^{g_i} g^{z_i z_3} = g_{1}^{g_{3}}, \quad K_4 = g^{z_4} = g^{z_2}$,

$K_6 = (g^{z_4})^{([z_5 z_6][z_7 z_8])} g^{z_7 z_9} = \left(\prod_{j \in \mathcal{S}(d_i)} u_{i}^{h_{i}} v_{\lambda_{26}}\right)^{T_{i}}$, 

$K_7 = (g^{z_4})^{g_{1}^{z_1}} = g^{z_1}, \quad K_8 = g^{z_4}$,

$K_9 = (g^{z_4})^{([z_5 z_6][z_7 z_8])} g^{z_7 z_9} = g^{z_4}$.

Next, $\mathcal{B}$ generates $K_1$, $K_2$, and $K_5$ elements as

$K_1 = Z^{-w_1} (g^{z_2 z_4})^{h_1} (g^{z_4})^{h_1}, \quad K_2 = Z^{w_1} (g^{z_2 z_4})^{h_1} (g^{z_4})^{h_1}$,

$K_5 = K_1^i K_2^j (g^{z_1})^{g_{1}^{z_1} g^{z_2 z_4} g^{z_7 z_9}} = K_1^i K_2^j (g^{z_1})^{g_{1}^{z_1} g^{z_2 z_4} g^{z_7 z_9}}$.

If $Z = g^{z_2 z_4}$, then $\mathcal{B}$ has that

$K_1 = (g^{z_2 z_4})^{z_1} (g^{z_4})^{z_1} = g^{-w_2 z_2 z_4} g^{-w_2 z_2} = g^{z_1} g^{z_2}$,

$K_2 = (g^{z_2 z_4})^{w_1} (g^{z_4})^{w_1} = g^{w_2 z_2} g^{z_4}$.

In this case, $\mathcal{B}$ generates the $i$th token as a normal token, so $\mathcal{B}$ plays Game$_{k-1}$ with $\mathcal{A}$. On the other hand, if $Z = g^{z_2 z_4}$ for some (non-zero) $\pi \in \mathbb{Z}_p$, then $\mathcal{B}$ has that

$K_1 = (g^{z_2 z_4})^{\pi} (g^{z_4})^{\pi} = g^{z_1} g^{z_2}$,

$K_2 = (g^{z_2 z_4})^{\pi} (g^{z_4})^{\pi} = g^{z_1} g^{z_2}$.

where $\pi$ plays a random exponent $i$ in $\mathbb{Z}_p$. In this case, $\mathcal{B}$ generates the $i$th token as a semi-functional token, so $\mathcal{B}$ plays Game$_k$ with $\mathcal{A}$.

Challenge Ciphertext $\mathcal{A}$ outputs two vectors $\vec{x}_{\mathcal{B}}, \vec{x}_{\mathcal{A}}$ and two messages $M_0, M_1$. $\mathcal{B}$ picks a random bit $b \in \{0, 1\}$ and random exponents $\ell_1, \ell_2, \ell_3, x \in \mathbb{Z}_p$. $\mathcal{B}$ (implicitly) sets $\vec{s}_2 = f_{2}^{x} x z_2 + s_2$ and $\vec{a}_{2}^{i} = A_{i} + B_{i} \cdot x_{\mathcal{B}, i} \in \mathbb{Z}_p$ for $i = 1, \ldots, \ell$, where $\vec{x}_{\mathcal{B}} = (x_{\mathcal{B}, 1}, \ldots, x_{\mathcal{B}, \ell})$. $\mathcal{B}$ computes a semi-functional ciphertext for $(\vec{x}_{\mathcal{B}}, M_b)$ as follows:

$C_1 = g^{w_{1} x_1} (g^{z_2})^{f_{1} \cdot x_2} g^{f_{2}} (g^{z_2})^{w_{1} x} (g^{z_2})^{\ell_1} x = W_{1} F_{1} \cdot g^{z_{1} x}$,

$C_2 = g^{w_{1} x_2} (g^{z_2})^{f_{1} \cdot x_2} g^{f_{2}} (g^{z_2})^{w_{1} x} (g^{z_2})^{\ell_1} x = W_{1} F_{1} \cdot g^{z_{1} x}$,

$C_3 = g^{w_{1} x_1} = g^{z_{1} x}$,

$C_4 = g^{w_{1} x_2} (g^{z_2})^{f_{1} \cdot x_2} g^{f_{2}} (g^{z_2})^{w_{1} x} (g^{z_2})^{\ell_1} x = W_{1} F_{1} \cdot g^{z_{1} x}$,

$C_5 = (g^{z_2})^{f_{1} \cdot x_2} g^{f_{2}} = g^{z_{1} x}$,

$C_6 = (g^{z_2})^{f_{1} \cdot x_2} g^{f_{2}} = (g^{z_2})^{w_{1} x} (g^{z_2})^{w_{2} z_2} (g^{z_2})^{f_{1} \cdot x_2} = (g^{z_2})^{w_{1} x} (g^{z_2})^{w_{2} z_2} (g^{z_2})^{f_{1} \cdot x_2} = (g^{z_2})^{w_{1} x} (g^{z_2})^{w_{2} z_2} (g^{z_2})^{f_{1} \cdot x_2}$,

$C_7 = (g^{z_2})^{(f_{1} \cdot x_2) \cdot x_2} (g^{z_2})^{(f_{1} \cdot x_2) \cdot x_2} = (g^{z_2})^{(f_{1} \cdot x_2) \cdot x_2}$,

$C_8 = g^{z_{1} x} = g^{z_{1} x}$,

$C_9 = g^{z_{1} x} = g_{1}^{z_{1} x}$,

$C_{10} = e(g_{1}, g_{2})^{(z_{1} x)} M_b = e(g_{1}, g_{2})^{(z_{1} x)} M_b$.

In computing $C_4$, we have that

$\tilde{\phi}_{1} \cdot \tilde{\phi}_{2} = (W_{1} z_{2} + \tilde{\phi}_{1} - \tilde{\phi}_{2}) (W_{2} z_{2} + \tilde{\phi}_{2}) = (W_{1} z_{2} + \tilde{\phi}_{1} - \tilde{\phi}_{2}) (W_{2} z_{2} + \tilde{\phi}_{2}) = (\tilde{\phi}_{1} \cdot \tilde{\phi}_{2}) (W_{1} z_{2} + \tilde{\phi}_{2}) - (\tilde{\phi}_{1} \cdot \tilde{\phi}_{2}) z_{2} = z_{2} (W_{1} z_{2} + \tilde{\phi}_{2})$. 

where $z = W_{1} \cdot z_{2} + W_{2} \cdot z_{2}$.
Suppose that the DBDH assumption holds. Then no polynomial time adversary under the DLIN assumption, these two games are indistinguishable.

Proof. Suppose that there exists an adversary \( A \) which can attack our HVE scheme with non-negligible advantage \( \epsilon \). We describe an algorithm \( B \) which uses \( A \) to solve the DBDH problem with advantage \( \epsilon \). On input \( (g, g^a, g^b, g^z) \in \mathbb{G}^4 \times \mathbb{G}_T \), \( B \)'s goal is to output 1 if \( z = e(g, g)^{abc} \) and 0 otherwise. \( B \) interacts with \( A \) as follows:

**Setup** \( B \) selects random exponents \( \alpha, \Omega, \gamma, \{y_i, \mu_i, t_i, c_i\}_{i=1}^\ell, \{\gamma_i\}_{i=1}^\ell, \{w_i, \delta_i, \phi_i\}_{i=1}^\ell, \{f_i\}_{i=1}^\ell \in \mathbb{Z}_p \), such that \( w_1 \delta_1 + w_2 \delta_2 = x \Omega \) (where \( \Omega \neq 0 \)).

\( B \) sets

\[
W_1 = (g^b)^\gamma g^{w_1}, \quad W_2 = (g^b)^\gamma g^{w_2}, \quad F_1 = g^f_1, \quad F_2 = g^f_2, \quad F_3 = g^f_3,
\]

\[
g_2 = g^2, \quad g_3 = (g^b)^{\frac{1}{2}} g^{y_1 - y_2} g^{w_1 - w_2} g_2, \quad g_4 = g^j,
\]

\[
Y_1 = g^{2^j}, \quad u_i = g^i, \quad h_i = g^i, \quad \tau_i = g^{\tau_i} (i = 1, \ldots, \ell),
\]

\[
u = g^{2^j}, \quad \phi = g^{2^j}, \quad A = e(g^a, g^b)^{\Omega} \cdot e(g, g)^{\gamma},
\]

\( B \) (implicitly) sets

\[
\tilde{w}_1 = \delta_2 \tilde{b} + w_1, \quad \tilde{w}_2 = -\delta_1 \tilde{b} + w_2, \quad \beta = (\delta_2 \phi_1 - \delta_1 \phi_2) \tilde{b} + (w_1 \phi_1 + w_2 \phi_2),
\]

\[
g_1 = g^{abc} g^{\gamma_1}.
\]

Notice that each public key element is independently and uniformly distributed as in the actual construction. Also, we can see that

\[
\tilde{w}_1 \delta_1 + \tilde{w}_2 \delta_2 = (\delta_2 \tilde{b} + w_1) \delta_1 + (-\delta_1 \tilde{b} + w_2) \delta_2 = w_1 \delta_1 + w_2 \delta_2 = x \Omega,
\]

\[
\tilde{w}_1 \phi_1 + \tilde{w}_2 \phi_2 = (\delta_2 \tilde{b} + w_1) \phi_1 + (-\delta_1 \tilde{b} + w_2) \phi_2 = (\delta_2 \phi_1 - \delta_1 \phi_2) \tilde{b} + (w_1 \phi_1 + w_2 \phi_2) = \beta.
\]

**Key Generation Phases 1 and 2** \( A \) issues token queries for vectors. For any queried vector \( \tilde{\sigma}_i, B \) generates a semi-functional token \( TK^\alpha_{\tilde{\sigma}_i} \). It selects random \( r_1, r_2, r_3, r_4, r_5 \) (subject to the equation \( (\Sigma_{i \in \mathbb{G}(\tilde{\sigma}_i)} r_i = \gamma r_5) \), tags: \( \lambda \in \mathbb{Z}_p \). \( B \) implicitly sets

\[
\tilde{r}_1 = -ab + r_1, \quad \tilde{\lambda} = a + \lambda.
\]

It computes the token as follows:

\[
K_1 = (g^b)^{\lambda_1} (g^a)^{-w_2} g^{t_1} r_1, \quad K_2 = (g^b)^{\lambda_2} (g^a)^{w_1} g^{t_2} r_2, \quad K_3 = g^j, \quad K_4 = g^s, \quad K_5 = K_1 K_2 K_4 f g^s, \quad K_6 = \left( \prod_{i \in \mathbb{G}(\tilde{\sigma}_i)} u_i h_i^{j_i} g^{\phi_i} \right)^{r_1}, \quad K_7 = g^t, \quad K_8 = g^\alpha, \quad K_9 = g^{\gamma_1}.
\]

The validity of \( K_1, K_2, K_3, K_5, \) and \( K_6 \) elements can be checked as follows:

\[
K_1 = (g^b)^{\lambda_1} (g^a)^{-w_2} g^{t_1} r_1, \quad g^{\phi_i} g^{w_2} = g^{(\phi_i - \lambda_1 b + w_2)} (\alpha + \lambda) = g^{(\phi_i - \lambda_1 b + w_2)} g^{w_2},
\]

\[
K_2 = (g^b)^{\lambda_2} (g^a)^{w_1} g^{t_2} r_2, \quad g^{\phi_i} g^{w_2} = g^{(\phi_i + \lambda_1 b + w_2)} (\alpha + \lambda) = g^{(\phi_i + \lambda_1 b + w_2)} g^{w_2},
\]

\[
K_3 = g^j, \quad K_4 = g^s, \quad K_5 = g^\alpha, \quad K_9 = g^{\gamma_1}.
\]
Observe that the unknown term $g^{ab}$ is canceled out in $K_1$, $K_s$, and $K_3$ elements, respectively.

**Challenge Ciphertext** $A$ outputs two vectors $x_i^c, x_i^c$ and two messages $M_0, M_1$. $B$ then flips a random coin $\beta \in \{0, 1\}$. $B$ picks random exponents $(\text{tag}, \gamma, \mu, \lambda, \nu, \tau, \phi, \psi, \theta, \omega, \chi, \alpha_0, \alpha_1) \in \mathbb{Z}_p^*$, self-implicitly sets $s_1 = c$ and $s = -bc + x$. $B$ computes a semi-functional ciphertext under $(x_i^c, x_i^c, M_\beta)$ as follows:

\[
C_1 = (g^{c})^{\beta_1} \cdot g^{f_1 x_1}, \quad C_2 = (g^{c})^{\beta_2} \cdot g^{f_2 x_2}, \quad C_3 = (g^{c})^{\beta_3} = g^{\tilde{s}_3},
\]

\[
C_4 = (g^{c})^{\beta_4} \cdot g^{f_2 x_2}, \quad C_5 = g^{\tilde{s}_5},
\]

\[
C_{6, i} = \left(u_i^{h_i^{\gamma_i}} \cdot g^{\nu_i \nu_i} \right) \cdot g^{\mu_i \mu_i}, \quad C_{7, i} = \left(\nu_i^{g^{\phi_i \phi_i}} \right) \cdot g^{\lambda_i \lambda_i}, \quad C_8 = g^{s_4},
\]

\[
C_9 = Z^{q} \cdot e(g, g^{c})^{\gamma_i} \cdot C_{M_{\beta}}.
\]

The validity of elements $C_1, C_2, C_3$ can be checked as follows:

\[
C_1 = (g^{c})^{\beta_1} \cdot g^{f_1 x_1} = g^{(\beta_1 + f_1 x_1)} = W_1 \cdot g^{\tilde{s}_1},
\]

\[
C_2 = (g^{c})^{\beta_2} \cdot g^{f_2 x_2} = g^{(\beta_2 + f_2 x_2)} = W_2 \cdot g^{\tilde{s}_2},
\]

\[
C_3 = (g^{c})^{\beta_3} \cdot g^{f_2 x_2} = g^{(\beta_3 + f_2 x_2)} = W_3 \cdot g^{\tilde{s}_3},
\]

If $Z = e(g, g)^{abc}$, then the element $C_9$ can be computed as follows:

\[
C_9 = Z^{q} \cdot e(g, g^{c})^{\gamma_i} \cdot C_{M_{\beta}} = e(g^{1}, g^{2})^{s_i} \cdot C_{M_{\beta}}.
\]

In this case, the challenge ciphertext is a valid encryption under $(x_i^c, M_s)$. Thus, $B$ is playing $\text{Game}_q$ with $A$. On the other hand, if $Z$ is random, $C_9$ is randomly distributed. Thus, $B$ is playing $\text{Game}_\text{final}$ with $A$.

**Guess** $B$ receives a bit $\beta' \in \{0, 1\}$ and outputs $0$ if $\beta' = \beta$.

**Analysis** As mentioned above, if $Z = e(g, g)^{abc}$, the challenge ciphertext is distributed as in $\text{Game}_q$, whereas if $Z$ is random, the challenge ciphertext is distributed as in $\text{Game}_\text{final}$. It follows that under the DBDH assumption, these two games are indistinguishable. □

Let $k = 1, \ldots, \ell$ and let $\text{Game}_M = \text{Game}_M^1$.

**Lemma 4.** Suppose that the DLIN assumption holds. Then no polynomial time adversary $A$ can distinguish between $\text{Game}_M^1$ and $\text{Game}_M^2$ with non-negligible advantage.

**Proof.** Suppose that there exists an adversary $A$ which can attack our HVE scheme with non-negligible advantage $e$. We describe an algorithm $B$ which uses $A$ to solve the DLIN problem with advantage $e$. On input $(g, g^{\gamma_1}, g^{\gamma_2}, g^{\gamma_3}, g^{\gamma_4}, Z) \in \mathbb{Z}_p^6$, $B$ interacts with $A$ as follows:

**Setup** $B$ selects random exponents $\gamma, (A_i, \gamma_i, \phi_i, \psi_i, \tau_i, \omega_i, \chi_i, \alpha_0, \alpha_1)$ in $\mathbb{Z}_p^*$. $B$ sets

\[
W_1 = g^{\gamma_1} \cdot g^{\mu_1}, \quad W_2 = g^{\mu_2}, \quad F_1 = g^{h_1}, \quad F_2 = g^{h_2}, \quad F_3 = (g^{\gamma_3})^{\lambda_3},
\]

\[
g_2 = g^{\gamma_2}, \quad g_3 = (g^{\gamma_3})^{\phi_1 + \mu_2} \cdot g^{\mu_1 \mu_2}, \quad g_4 = g^{\gamma_4},
\]

\[
Y_i = (g^{\gamma_i})^{\lambda_i} \quad (i = 1, \ldots, k), \quad Y_k = (g^{\gamma_3})^{\lambda_3},
\]

\[
u_i = g^{h_3} \quad (i = 1, \ldots, k), \quad \nu_k = (g^{\gamma_3})^{\lambda_3} g^{h_3},
\]

\[
s_i = g^{h_i} \quad (i = 1, \ldots, k), \quad s_k = (g^{\gamma_3})^{\lambda_3} g^{h_3},
\]

Note the values $\{A_i\}_{i=1}^\ell$ are information-theoretically hidden. $B$ (implicitly) sets

\[
\gamma = \gamma_1, \quad \gamma_i = \gamma_i \gamma_1 \quad (i = 1, \ldots, k), \quad \gamma_k = \gamma_k \gamma_1, \quad \gamma = (e(g, g)^{\gamma_3}),
\]

\[
\tilde{f}_3 = f_3 \cdot \gamma_3, \quad \tilde{w}_1 = \tilde{w}_1 + \nu_1, \quad \tilde{\phi}_2 = \tilde{\phi}_2 + \phi_2, \quad \tilde{\omega} = (\tilde{\omega} + \omega_1 \phi_1 + \omega_2 \phi_2) / \gamma,
\]

\[
\beta = (\phi_1 + \nu_1) \gamma_1 + (\nu_1 \phi_1 + \nu_2 \phi_2), \quad g_1 = g^{s_1}.
\]

Notice that each public key element is independently and uniformly distributed as in the actual construction. Also, we can see that
\[\begin{align*}
\tilde{w}_1\delta_1 + w_2\phi_2 = (z_1 + w_1)\delta_1 + w_2\phi_2 = \delta_1z_1 + w_1\delta_1 + w_2\phi_2 = z\Omega, \\
\tilde{w}_1\phi_1 + w_2\phi_2 = (z_1 + w_1)\phi_1 + w_2(\phi_1 + \phi_2) = (\phi_1 + w_2)z_1 + (w_1\phi_1 + w_2\phi_2) = \beta.
\end{align*}\]

**Key Generation Phases** A issues token queries for vectors. B generates a semi-functional token for the requested vector \(\vec{\alpha}\). B handles this in one of two ways.

**Case I:** \(k \neq S(\vec{\alpha})\).

B picks random exponents \(r_1, \ r_2, \ r_3, \ r_4, \ \lambda, \ \tag_k(\neq \Sigma_{j \in S(\vec{\alpha})}A_j)\) in \(\mathbb{Z}_p\). It implicitly sets
\[r_3 = r_2z_1, \quad \tilde{r}_4 = r_3(\Sigma_{j \in S(\vec{\alpha})}y_jz_1).
\]

Note that the equation
\[(\Sigma_{j \in S(\vec{\alpha})}y_j)r_3 = (\Sigma_{j \in S(\vec{\alpha})}y_j)r_2z_1 = z_1 \cdot r_3(\Sigma_{j \in S(\vec{\alpha})}y_jz_1) = \gamma \tilde{r}_5
\]
is satisfied. B generates the semi-functional token as follows:
\[\begin{align*}
K_1 &= g^{\lambda r_1}g^{\phi_1 r_2 - w_2}g = g^{\lambda r_1}g^{\phi_1 r_2} \cdot g^{-w_2}g, \\
K_2 &= g^{g^{y_1 r_1}g^{y_2 r_2} + w_2}g = g^{g^{y_1 r_1}g^{y_2 r_2 + w_2}} \cdot g^{g^{r_2 - w_1}}g = g^{g^{y_1 r_1}g^{y_2 r_2} \cdot g^{w_1}}, \\
K_3 &= g^{g^{y_1 r_1}g^{y_2 r_2} + w_2}g^{w_1 r_2}g^{r_2} = g_1 g^{\Omega r_1}, \\
K_4 &= g^{z_2}, \\
K_5 &= K_1 K_2^2 (g^{z_2})^{-r_1} (g^{y_2})^{-r_2} = K_1^2 K_2^2 (g^{z_2})^{-r_1} (g^{y_2})^{-r_2}, \\
K_6 &= (g^{z_2})^{-r_1} \left( \prod_{j \in S(\vec{\alpha})} g^{y_j r_1}g^{y_j r_2} \cdot g^{y_1 r_2} \right) \left( \prod_{j \in S(\vec{\alpha})} (g^{y_j r_1}g^{y_j r_2})^{r_2} \right) \left( \prod_{j \in S(\vec{\alpha})} \tau_j \cdot g^{a_j} \right), \\
K_7 &= (g^{z_2})^{-r_1} = g^{z_2}, \quad K_8 = g^{r_4}, \\
K_9 &= (g^{z_2})^{-r_1} = g^{z_2}.
\end{align*}\]

Note that B can generate a normal token\(^5\) for vectors in Case I, even though B computes the semi-functional token. This is because B can simply set the exponent \(r_4\) to not include the unknown terms \(z_1\) or \(z_2\). However, the semi-functionality of tokens is necessary for handling queries in the next case.

**Case II:** \(k \in S(\vec{\alpha})\).

For notational convenience, we define \(\tilde{S} = S(\vec{\alpha})/(k)\). B picks random exponents \(r_1, \ r_2, \ r_3, \ r_4, \ \lambda, \ \tag_k(\neq \Sigma_{j \in S(\vec{\alpha})}A_j)\) in \(\mathbb{Z}_p\). It implicitly sets
\[\begin{align*}
\lambda &= \lambda - \frac{r_3 y_k z_2}{r_3(\Sigma_{j \in S(\vec{\alpha})}y_j - \tag_k)}, \\
\tilde{r}_2 &= r_2 + \frac{r_3 y_k z_2}{r_3(\Sigma_{j \in S(\vec{\alpha})}y_j - \tag_k)}, \\
\tilde{r}_3 &= r_3 z_1, \quad \tilde{r}_4 = r_4 + \frac{r_3 y_k z_1}{\Sigma_{j \in S(\vec{\alpha})}y_j - \tag_k}, \quad \tilde{r}_5 = r_3 \left( y_k z_2 + \Sigma_{j \in S(\vec{\alpha})}y_j z_1 \right).
\end{align*}\]

Note that the equation
\[(\Sigma_{j \in S(\vec{\alpha})}y_j)r_3 = (y_k z_2 + \Sigma_{j \in S(\vec{\alpha})}y_j z_1) r_3 z_1 = z_1 \cdot r_3 \left( y_k z_2 + \Sigma_{j \in S(\vec{\alpha})}y_j z_1 \right) = \gamma \tilde{r}_5,
\]
is satisfied. For notational convenience, we let \(\Phi = y_k/(\Sigma_{j \in S(\vec{\alpha})}y_j - \tag_k)\). B generates the semi-functional token as follows:

---

\(^5\) This fact will be used in proving Claim 2.
$K_1 = g^{h_1 r_1} \cdot g^{(r_1 p_1 + r_1 w_2) / f_1} = g^{h_1 r_1} \cdot g^{(r_1 p_1 + r_1 w_2) / f_1} = g^{h_1 r_1} \cdot g^{(r_1 p_1 + r_1 w_2) / f_1}$

$K_2 = g^{h_2 r_2} \cdot g^{(r_2 p_2 + r_2 w_4) / f_2} = g^{h_2 r_2} \cdot g^{(r_2 p_2 + r_2 w_4) / f_2} = g^{h_2 r_2} \cdot g^{(r_2 p_2 + r_2 w_4) / f_2}$

$K_3 = g^{h_3 r_3} \cdot g^{(r_3 p_3 + r_3 w_5) / f_3} = g^{h_3 r_3} \cdot g^{(r_3 p_3 + r_3 w_5) / f_3}$

$K_4 = g^{h_4 r_4} \cdot g^{(r_4 p_4 + r_4 w_6) / f_4} = g^{h_4 r_4}$

$K_5 = K_1 K_2 (g^{e^1 r_1} \cdot g^{(e^1 p_1 + e^1 w_2) / f_1}) = K_1 K_2 (g^{e^1 r_1} \cdot g^{(e^1 p_1 + e^1 w_2) / f_1})$

$K_6 = (g^{e^2 r_2})^{\gamma r_2 + (e^2 p_2 + e^2 w_4) / f_2} = (g^{e^2 r_2})^{\gamma r_2 + (e^2 p_2 + e^2 w_4) / f_2}$

$B$ cannot force $B$ to generate a normal token for vectors in Case II, which is because $B$ is forced to set the exponent $r_4$ to include $z_4$. From this starting point, $B$ eventually has to use the additional terms for semi-functional token in generating the component $K_2$.

**Challenge Ciphertext** $A$ outputs two vectors $x^0_i, x^1_i$ and two messages $M_0, M_1$. $B$ picks a random bit $b \in \{0, 1\}$. $B$ selects random $s_1, x \in \mathbb{Z}_p$, random $R_1, \ldots, R_{k-1} \in \mathbb{G}$, and random $R_t \in \mathbb{G}_t$, and it sets $\text{tag}_{1}^i = A_i$ for $i = 1, \ldots, \ell$. $B$ implicitly sets $S_2 = z_4, S_3 = z_3, X = x + f_4 z_4 / \delta_1$.

$B$ computes a semi-functional ciphertext $CT^b$ for $(\{r_1, \ldots, r_{k-1}, r_k \text{ or } x^0_h, x^0_{k+1}, \ldots, x^0_{\ell}\}, M_b)$ as follows:

$C_1 = (g^{e_1 r_1})^{\gamma r_1 + (e_1 p_1 + e_1 w_2) / f_1} = g^{e_1 r_1} \cdot g^{(e_1 p_1 + e_1 w_2) / f_1} = g^{e_1 r_1} \cdot g^{(e_1 p_1 + e_1 w_2) / f_1}$

$C_2 = g^{e_2 p_2} \cdot g^{(e_2 p_2 + e_2 w_4) / f_2} = g^{e_2 p_2} \cdot g^{(e_2 p_2 + e_2 w_4) / f_2}$

$C_3 = g^{e_3 r_3} = g^{e_3}$

$C_4 = (g^{e_4 r_4})^{\gamma r_4 + (e_4 p_4 + e_4 w_6) / f_4} = (g^{e_4 r_4})^{\gamma r_4 + (e_4 p_4 + e_4 w_6) / f_4}$

$C_5 = g^{e_5} = g^{e_5}$

$C_{6,i} = R_i \quad (i = 1, \ldots, k - 1), \quad C_{6,k} = Z^{\ell k} \cdot (g^{e_4 r_4})^{\gamma r_4 + (e_4 p_4 + e_4 w_6) / f_4}$

$C_{7,i} = (g^{e_7 r_7})^{\gamma r_7 + (e_7 p_7 + e_7 w_9) / f_7} = (g^{e_7 r_7})^{\gamma r_7 + (e_7 p_7 + e_7 w_9) / f_7}$

$C_{8} = g^{e_8 r_8} = g^{e_8}$, \quad $C_{9} = R_7$.

---

6 This fact will be used in proving Claim 2.
$B$ cannot compute a normal ciphertext as a challenge, because it is required to use the additional term $g^{(z_3 + z_4)}$ for semi-functional ciphertext in order to remove $g^{z_3}$ derived from $F_3$. At first glance, the term $g^{z_3}$ can also be canceled out by making the exponent $s_i$ include $z_4$, but in that case, the component $C_1 = W_1 F_1 = g^{(z_3 + z_4)} g^{z_3}$ has the term $g^{z_3}$.

As a result, $B$ has to generate the challenge ciphertext in a semi-functional form. Notice that $G_{ij}$ for $i = 1, \ldots, k - 1$ can also be generated in the right form\(^7\) as in $C_{ij}$, for $i = k + 1, \ldots, \ell$, but by the hybrid argument these elements are replaced with random group elements in $G$. In case of $G_{ij}$, $B$ can generate the element in the right form\(^8\), but it can also be replaced with a random element in $G$.

If $Z = g^{z_3(z_3 + z_4)}$, then $C_{Gk}$ is computed as follows:

$$C_{Gk} = (g^{z_3(z_3 + z_4)})^{y_k} \cdot (g^{z_3})^{\nu_k} = (g^{z_3})^{y_k + \nu_k} = (g^{z_3})^{g^{\nu_k} \cdot g^{y_k}} = (g^{z_3})^{(h_k)_{256}^2 \cdot g^{y_k}}.$$  

In this case, $B$ generates the semi-functional ciphertext for the vector $(r_1, \ldots, r_{k - 1}, x_{k - 1}^2, \ldots, x_{k - 1}^k)$, so $B$ plays $Game_{final'}$ with $A$. On the other hand, if $Z = g^{z_3(z_3 + z_4)} g^{v}$ for some (non-zero) $v \in \mathbb{Z}_p$, then $B$ has that

$$C_{Gk} = (g^{z_3(z_3 + z_4)})^{y_k} \cdot (g^{z_3})^{\nu_k} = (g^{z_3})^{\nu_k + z_3 \cdot y_k} = (g^{z_3})^{\nu_k + \nu_k} = (g^{z_3})^{(h_k)_{256}^2 \cdot g^{y_k}}.$$  

In this case, $B$ generates the semi-functional ciphertext for the vector $(r_1, \ldots, r_{k - 1}, x_{k - 1}^2, x_{k - 1}^3)$, where $r_{k - 1}$ is a discrete logarithm satisfying $g^{y_k} = g^{z_3 r_{k - 1}} = (h_k)_{256}^2$. Since $\pi$ is uniformly distributed at random, so is $x_{k - 1}^3$. This means that $B$ plays $Game_{final'}$ with $A$.

**Guess** $B$ receives a bit $b' \in \{0, 1\}$ and outputs 0 if $b' = b$.

**Analysis** As mentioned above, if $Z = g^{z_3(z_3 + z_4)} B$ is in $Game_{final'}$, whereas if $Z = g^{z_3(z_3 + z_4)} g^{v} B$ is in $Game_{final'}$. It follows that under the Decision Linear assumption, these two games are indistinguishable. \(\square\)

By combining the results of Lemmas 1–4, we obtain the following security result:

**Theorem 1** (Case 1). Assume the DLIN and DBDH assumptions hold in $G$. Then, our HVE scheme is (attribute-hiding) secure in Case 1.

**Case 2:** (Proof idea) We give a key idea behind the security proof in Case 2. When given the challenge ciphertext, the adversary aims to decide which one of the two pairs $(\hat{x}_0^k, M)$ and $(\hat{x}_1^k, M)$ is associated with the challenge ciphertext. As in Case 1, the basic step is to change the challenge ciphertexts into semi-functional ones, but the difference is that tokens are changed from normal to semi-functional ones or vice versa during the security game. The main obstacle comes from the fact that the adversary can query matching tokens for both the challenge vectors $\hat{x}_0^k$ and $\hat{x}_1^k$ simultaneously. This means that the adversary can use the matching tokens to decrypt the challenge ciphertext that would be an encryption of $\hat{x}_0^k$ for a random bit $b \in \{0, 1\}$. However, under the fair rule of the security game, we know that the matching tokens are associated with the vector components such that $x_{0i}^k = x_{1i}^k$ for $i \in \{1, \ldots, \ell\}$. Also, for the other pairwise-distinct vector components, tokens cannot be matching for $\hat{x}_0^k$ and $\hat{x}_1^k$ simultaneously, and thus they are not helpful to the adversary when trying to decrypt the challenge ciphertext. This observation shows that when tokens are generated with at least one pairwise-distinct component, we can generate the resulting tokens in a semi-functional form.

At each position $k \in \{1, \ldots, \ell\}$, we consider two cases depending on $x_{0k}^k = x_{1k}^k$ or not. In any case, all tokens including any $k$th component can be semi-functional and the other tokens not including any $k$th component are normal until the challenge phase. In the case where $x_{0k}^k \neq x_{1k}^k$, there should be no matching token including $k$th component as we observed above. Also, even if the challenge ciphertext is semi-functional, other matching tokens not involving $k$th component (if possible) will decrypt the challenge ciphertext correctly. Thus, we can change the $k$-th component $x_{0k}^k$ in the challenge ciphertext into a random one by the similar simulation to the one of Case 1, and then we move onto the next position $k + 1$. In the other case where $x_{0k}^k = x_{1k}^k$, we will perform the same process in generating tokens until the challenge phase. However, when the adversary outputs $(\hat{x}_0^k, M)$ and $(\hat{x}_1^k, M)$ where $x_{0k}^k = x_{1k}^k$, then we cannot construct the semi-functional challenge ciphertext since the adversary can be already given semi-functional tokens associated with the $k$-th component $x_{0k}^k = x_{1k}^k$. Fortunately, in that case, we can move onto the next position $k + 1$ without generating the challenge ciphertext at position $k$, which is because the challenge ciphertext at position $k$ has exactly the same distribution as that in the previous position $k - 1$ (or further previous position). Before moving onto the next position $k + 1$, we return all semi-functional tokens into normal ones to prepare for generating tokens in the next position. In this way, we can proceed to the last vector component.

The simulator considers a sequence of hybrid games as follows:

- $Game_{20}^2$: This is the actual HVE security game in Case 2. All tokens will be normal and the challenge ciphertext will be a normal challenge ciphertext on a pair $(\hat{x}_b^k, M)$, where $b \in \{0, 1\}$ is a random bit and $M = M_0 = M_1$.
- $Game_{21}^2$: All tokens will be normal, but the challenge ciphertext will be a semi-functional ciphertext on a pair $(\hat{x}_b^k, M)$.

---

\(^7\) This fact will be used in proving Claim 2.

\(^8\) This fact will be used in proving Claim 2.
Game$^2$: All tokens will be normal, and the challenge ciphertext will be a semi-functional ciphertext on a pair $(\langle r_1, x_{b_1}, \ldots, x_{b_2}\rangle, M)$, where $r_1$ is a random element from $\Sigma$ if $x_{0,1} \neq x_{1,1}$ and otherwise $r_1 = x_{0,1}$.

... ...

Game$^2$: All tokens will be normal, and the challenge ciphertext will be a semi-functional ciphertext on a pair $(\langle r_i, r_2, \ldots, r_i \rangle, M)$, where for $i = 1, \ldots, t$, $r_i$ is a random element from $\Sigma$ if $x_{0,i} \neq x_{1,i}$ and otherwise $r_i = x_{0,i}$.

In Game$^2_{\text{Real}}$, the normal challenge ciphertext on a pair $(\tilde{x}_0, M)$ is given to the adversary. On the other hand, in Game$^2$, the challenge ciphertext given to the adversary is a semi-functional ciphertext corresponding to $(\langle r_1, \ldots, r_t \rangle, M)$ that leaks no information about $\tilde{x}_0$. Note that $r_i$ is random for each $i$ such that $x_{0,i} \neq x_{1,i}$, so that the final vector $(r_1, r_2, \ldots, r_t)$ does not help the adversary to tell between $\tilde{x}_0$ and $\tilde{x}_1$. We will show that no polynomial time adversary is able to distinguish between Game$^2_{\text{Real}}$ and Game$^2$ by proving that the transitions between the sequence of games above are all computationally indistinguishable under the DLIN assumption.

**Lemma 5.** Suppose that the DLIN assumption holds. Then no polynomial time adversary $A$ can distinguish between Game$^2_{\text{Real}}$ and Game$^2$ with non-negligible advantage.

**Proof.** The proof of Lemma 5 is identical to that of Lemma 1. □

Next, in order to prove that distinguishing between two games Game$^2_{k-1}$ and Game$^2_k$ for $k = 1, \ldots, t$ is computationally hard, we consider two cases (for each $k$) depending on whether $x_{0,k} \neq x_{1,k}$ or $x_{0,k} = x_{1,k}$ when $A$ outputs two challenge vectors $\tilde{x}_0 = (x_{0,1}, \ldots, x_{0,t})$ and $\tilde{x}_1 = (x_{1,1}, \ldots, x_{1,t})$. In the first case, $A$ should not issue token queries for $\tilde{\sigma}_i = (\sigma_{i,1}, \ldots, \sigma_{i,k}, \ldots, \sigma_{i,t})$ such that $k \in S(\tilde{\sigma}_i)$ and $P_i(\tilde{\sigma}_i, \tilde{x}_0) = P_i(\tilde{\sigma}_i, \tilde{x}_1) = 1$. This means that tokens including $k$-th component $\sigma_{ik}$ as non-wildcard should not be matching queries on either $\tilde{x}_0$ or $\tilde{x}_1$. Thus, the tokens including $\sigma_{ik}$ do not help $A$ decrypt the challenge ciphertext correctly. The impossibility of decryption allows the simulator to change the tokens from normal to semi-functional even if the challenge ciphertext will be semi-functional. However, other tokens remain normal so that $A$ is able to use the other normal tokens to decrypt the challenge ciphertext successfully. Once the tokens including the $k$-th component $\sigma_{ik}$ and the challenge ciphertext will be semi-functional, the $k$-th component $x_{0,k}$ for a random bit $b \in \{0,1\}$ is replaced with a random element. This is performed in a game defined (below) as Game$^2_{k-1,k}$, and then the tokens return back to the normal type necessary for the next intermediate game Game$^2_k$. On the other hand, the second case when $x_{0,k} = x_{1,k}$ makes two intermediate games equal, i.e., Game$^2_{k-1} = Game^2_k$. Thus, we can naturally move onto the next intermediate games in Case 2.

Now, it remains to show how the simulator behaves in the case where $x_{0,k} \neq x_{1,k}$. Let $q_k$ be the number of token queries for vectors $\{\tilde{\sigma}_i\}_{i=1}^k$ such that $k \in S(\tilde{\sigma}_i)$. Then the simulator considers a sequence of hybrid games as follows:

Game$^2_{k-1}$: All tokens will be normal, and the challenge ciphertext will be a semi-functional ciphertext on a pair $(\langle r_1, \ldots, r_{k-1}, x_{b_1}, \ldots, x_{b_2}\rangle, M)$, where the elements $\{r_i\}$ for $i = 1, \ldots, k-1$ are random from $\Sigma$ if $x_{0,i} \neq x_{1,i}$ and otherwise $r_i = x_{0,i}$.

... ...

Game$^2_{k-1,j}$: The first $j$ tokens in $\{\tilde{\sigma}_i\}_{i=1}^{q_j}$ will be semi-functional and the other tokens will be normal, and the challenge ciphertext will be a semi-functional ciphertext on a pair $(\langle r_1, \ldots, r_{k-1}, x_{b_1}, \ldots, x_{b_2}\rangle, M)$.

... ...

Game$^2_{k-1,q_k}$: All tokens in $\{\tilde{\sigma}_i\}_{i=1}^{q_k}$ will be semi-functional and the other tokens will be normal, and the challenge ciphertext will be a semi-functional ciphertext on a pair $(\langle r_1, \ldots, r_{k-1}, x_{b_1}, \ldots, x_{b_2}\rangle, M)$.

... ...

Game$^2_{k-1,q_k-1}$: All tokens in $\{\tilde{\sigma}_i\}_{i=1}^{q_k-1}$ will be semi-functional and the other tokens will be normal, and the challenge ciphertext will be a semi-functional ciphertext on a pair $(\langle r_1, \ldots, r_k, x_{b_1}, \ldots, x_{b_2}\rangle, M)$, where $r_k$ is random.
The following claims show that all these hybrid games are indistinguishable under the DLIN assumption, so that distinguishing between \( \text{Game}_{k-1,j}^2 \) and \( \text{Game}_{k-1,j}^2 \) is computationally infeasible.

Let \( j = 1, \ldots, q_k \) and \( \text{Game}_{k-1}^{2} = \text{Game}_{k-1,0}^{2} \).

**Claim 1.** Suppose that the DLIN assumption holds. Then no polynomial time adversary \( \mathcal{A} \) can distinguish between \( \text{Game}_{k-1,j-1}^{2} \) and \( \text{Game}_{k-1,j}^{2} \) with non-negligible advantage.

**Proof.** The proof of Claim 1 is almost identical to that of Lemma 2. The difference is that the first \( j - 1 \) tokens in \( \{ \tilde{s}_i \}_{i=1}^{q_{k-1}} \) are generated as semi-functional ones, and the \( j \)-th token is generated using the target element \( Z \) of a DLIN problem, and the other tokens are generated as normal ones. \( \square \)

**Claim 2.** Suppose that the DLIN assumption holds. Then no polynomial time adversary \( \mathcal{A} \) can distinguish between \( \text{Game}_{k-1,q_k}^{2} \) and \( \text{Game}_{k-1,\ell} \) with non-negligible advantage.

**Proof.** The proof of Claim 2 is almost identical to that of Lemma 4. The difference is that all tokens in \( \{ \tilde{s}_i \}_{i=1}^{q_{k-1}} \) are generated as semi-functional ones and the other tokens are generated as normal ones. In constructing the challenge ciphertext, the elements \( \{ c_i \}_{i=1}^{q_{k-1}} \) are generated in the right form if \( x_{0,i} = x_{1,i} \) for \( i = 1, \ldots, k - 1 \) and \( c_0 \) is also generated in the right form. \( \square \)

Let \( j = q_k, \ldots, 1 \), \( \text{Game}_{k-1,\ell} = \text{Game}_{k-1,j}^{2} \), and \( \text{Game}_{k-1} = \text{Game}_{k-1,0}^{2} \).

**Claim 3.** Suppose that the DLIN assumption holds. Then no polynomial time adversary \( \mathcal{A} \) can distinguish between \( \text{Game}_{k-1,j}^{2} \) and \( \text{Game}_{k-1,j}^{2} \) with non-negligible advantage.

**Proof.** The proof of Claim 3 is identical to that of Claim 1 (Lemma 2), except that the \( k \)-th component \( x_{0,k} \) is replaced with a random element in generating the challenge ciphertext. \( \square \)

By putting the results of claims all together, we obtain the security result of Lemma 6. Let \( k = 1, \ldots, \ell \).

**Lemma 6.** Suppose that the DLIN assumption holds. Then no polynomial time adversary \( \mathcal{A} \) can distinguish between \( \text{Game}_{k-1}^{2} \) and \( \text{Game}_{k-1}^{2} \) with non-negligible advantage.

By combining the results of Lemmas 5 and 6, we obtain the following security result of Case 2:

**Theorem 2.** (Case 2) Assume the DLIN assumption holds in \( G \). Then, our HVE scheme is (attribute-hiding) secure in Case 2.

5. **Comparison to other HVE schemes**

Table 1 gives a comparison of the different features in previous HVE schemes and ours when encrypting \( \ell \)-dimensional vectors. Any IPE scheme can be straightforwardly transformed into an HVE scheme by expanding the dimension of vectors from \( \ell \) to \( 2\ell \) [28], so we consider the previous IPE schemes [28,36,30,38,35,34] handling \( 2\ell \)-dimensional vectors as HVE schemes handling \( \ell \)-dimensional vectors. We point out that [34] is an independent work that has recently suggested a fully secure IPE scheme. In terms of achieving full security, [30,18,35] are weakly attribute-hiding in the sense that an adversary can make token queries such that \( P_{i}(\tilde{x}_0, \tilde{s}_i) = P_{i}(\tilde{x}_1, \tilde{s}_i) = 0 \) for all queried vectors \( \{ \tilde{s}_i \} \), where \( \tilde{x}_0 \) and \( \tilde{x}_1 \) are vectors challenged by the adversary. [18] suggested another HVE scheme which is secure in the opposite sense of security modelling where \( P_{i}(\tilde{x}_0, \tilde{s}_i) = P_{i}(\tilde{x}_1, \tilde{s}_i) = 1 \) for all queried vectors \( \{ \tilde{s}_i \} \). In any case, our HVE scheme (as well as the independent work [34]) is the first one that is fully secure in the security model where both matching and non-matching token queries are validly considered in a single security game.

Regarding efficiency, the schemes in [40,29,34] and ours have the property that both token size and the number of pairing computations (necessary for decryption) do not depend on the dimension \( \ell \) of attribute vectors. As mentioned before, these schemes are desirable in applications where \( \ell \) increases to deal with more expressive access control. Table 1 simply shows the comparison of \( \ell \) conjunctive equality predicates, but when we consider access control along with conjunctive...
combination of comparison and subset predicates, the efficiency impact is stronger. For instance, if a subset predicate is defined over a set of \( n \) elements, one subset predicate leads to a token of size \( O(n) \) and pairing computations of size \( O(n) \). Thus, if an access control is a conjunctive combination that consists of \( \ell_1 \) equality, \( \ell_2 \) comparison, and \( \ell_3 \) subset predicates, the token size and pairing computations for such an access control increase with \( O(\ell_1 + \ell_2 + \ell_3) \) in other HVE schemes. However, in [40,29,34] and ours, these two factors remain \( O(1) \) regardless of the numbers of conjunctions.

6. Conclusion

We presented the first HVE scheme that is fully secure under the DBDH and DLIN assumptions. Our HVE scheme required \( O(1) \)-sized private keys and \( O(1) \) pairing computations for decryption, regardless of the dimension of vectors. These advantages are attractive to the query server as the dimension increases to support more expressed access control. This was achieved by first constructing a novel type of (tag-based) dual system encryption. New techniques were then applied to both conceal vector components from ciphertexts and compress tag values into one. Our HVE scheme also yielded an anonymous IBE scheme that is fully secure under the standard assumptions.

It was difficult to extend our HVE (and anonymous IBE) scheme to support a hierarchical delegation mechanism. Thus, it is still an open problem to construct an HVE scheme supporting delegation, while preserving full security under standard assumptions. Another interesting open problem is to create an IPE scheme that is fully secure under standard assumptions in a way that both matching and non-matching key queries are allowed. It would also be interesting to show how to reduce the number of pairing computations to \( O(1) \) in an IPE scheme, which has seemed difficult to achieve.

Acknowledgements

The authors thank the reviewers for their helpful comments and suggestions for this paper. Jong Hwan Park and Dong Hoon Lee were supported by the National Research Foundation of Korea (NRF) Grant funded by the Korea government (MEST) (No. 2012-0008697). Willy Susilo was supported by ARC Future Fellowship FT0991397.

References