Hidden Vector Encryption Fully Secure Against Unrestricted Queries

No query left unanswered

Angelo De Caro Vincenzo Iovino
Giuseppe Persiano

Dipartimento di Informatica ed Applicazioni,
Università di Salerno, 84084 Fisciano (SA), Italy.
{decaro,iovino,giuper}@dia.unisa.it.

Abstract

Predicate encryption is an important cryptographic primitive (see [3, 6, 11, 14]) that enables fine-grained control on the decryption keys. Roughly speaking, in a predicate encryption scheme the owner of the master secret key Msk can derive secret key Sk_P, for any predicate P from a specified class of predicates $\mathbb{P}$. 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 Sk_P such that $P(\vec{x}) = 1$.

Our main contribution is the *first* construction of a predicate encryption scheme that can be proved *fully* secure against *unrestricted* queries by probabilistic polynomial-time adversaries under non-interactive constant sized (that is, independent of the length ℓ of the attribute vectors) hardness assumptions on bilinear groups of composite order.

Specifically, we consider *hidden vector encryption* (HVE in short), a notable case of predicate encryption introduced by Boneh and Waters [6] and further developed in [24, 13, 22]. In a HVE scheme, the ciphertext attributes are vectors $\vec{x} = \langle x_1, \ldots, x_\ell \rangle$ of length ℓ over alphabet Σ, keys are associated with vectors $\vec{y} = \langle y_1, \ldots, y_\ell \rangle$ of length ℓ over alphabet $\Sigma \cup \{\star\}$ and we consider the $\mathsf{Match}(\vec{x}, \vec{y})$ predicate which is true if and only if, for all i, $y_i \neq \star$ implies $x_i = y_i$. Previous constructions restricted the proof of security to adversaries that could ask only *non-matching* queries; that is, for challenge attribute vectors $\vec{x}_0$ and $\vec{x}_1$, the adversary could ask only for keys of vectors $\vec{y}$ for which $\mathsf{Match}(\vec{x}_0, \vec{y}) = \mathsf{Match}(\vec{x}_1, \vec{y}) = \text{false}$.

Our proof employs the dual system methodology of Waters [26], that gave one of the first fully secure construction in this area, blended with a careful design of intermediate security games that keep into account the relationship between challenge ciphertexts and key queries.

Keywords: predicate encryption, full security, pairing-based cryptography.

1 Introduction and related work

Predicate encryption is an important cryptographic primitive (see [3, 6, 11, 14]) that enables fine-grained control on the decryption keys. Roughly speaking, in a predicate encryption scheme for a class $\mathbb{P}$ of ℓ-ary predicates, the owner of the master secret key Msk can derive secret key Sk_P for any predicate $P \in \mathbb{P}$. In encrypting a message M, the sender can specify an *attribute* vector $\vec{x}$ of length ℓ and the resulting ciphertext X can be decrypted only by using keys Sk_P such that $P(\vec{x}) = 1$. Thus a predicate encryption scheme enables the owner of the master secret key to delegate the decryption of different types of ciphertexts to different entities by releasing the appropriate key.

Our main contribution is the *first* construction of a predicate encryption scheme that can be proved *fully* secure against *unrestricted* queries from probabilistic polynomial-time adversaries under non-interactive constant sized (that is independent of ℓ) hardness assumptions on bilinear groups of composite order.

More specifically, we consider *hidden vector encryption* (HVE in short), a notable case of predicate encryption introduced by [6]. In a HVE scheme, the ciphertext attributes are vectors $\vec{x} = \langle x_1, \ldots, x_\ell \rangle$ of length ℓ over alphabet Σ and predicates are described by vectors $\vec{y} = \langle y_1, \ldots, y_\ell \rangle$ of length ℓ over alphabet $\Sigma \cup \{\star\}$. The class $\mathbb{P}$ of predicates for HVE consists of all predicates $\mathsf{Match}_{\vec{y}}$ defined as follows: $\mathsf{Match}_{\vec{y}}(\vec{x})$ is true if and only if, for all i, $y_i \neq \star$ implies $x_i = y_i$. In the rest of the paper we will adopt the writing $\mathsf{Match}(\vec{x}, \vec{y})$ instead of $\mathsf{Match}_{\vec{y}}(\vec{x})$.

We model our security notion by means of a game between a challenger $\mathcal{C}$ and a PPT adversary $\mathcal{A}$ that sees the public key (thus we depart from the *selective* model of security), is allowed to ask for keys of vectors $\vec{y}$ of his choice and gives two *challenge vectors* $\vec{x}_0$ and $\vec{x}_1$. $\mathcal{A}$ then receives a *challenge ciphertext* (an encryption of a randomly chosen challenge vectors) and has to guess which of the two challenge vectors has been encrypted. The adversary $\mathcal{A}$ is allowed to ask queries even after seeing the challenge ciphertext. Unlike previous work, we only require the adversary $\mathcal{A}$ to ask for keys of vectors $\vec{y}$ that do not discriminate the two challenge vectors; that is, for which $\mathsf{Match}(\vec{x}_0, \vec{y}) = \mathsf{Match}(\vec{x}_1, \vec{y})$. It can be readily seen that this condition is necessary. Previous constructions restricted the proof of security to adversaries that could ask only *non-matching* queries; that is, ask for keys of vectors $\vec{y}$ such that $\mathsf{Match}(\vec{x}_0, \vec{y}) = \mathsf{Match}(\vec{x}_1, \vec{y}) = 0$. Thus our construction is the first to be proved fully secure against *unrestricted* PPT adversaries $\mathcal{A}$.

Besides being one of the first predicates for which constructions have been given, HVE can be used as building block for several other predicates. Specifically in [6], it is shown that HVE implies predicate encryption schemes for conjunctions, comparison, range queries and subset queries. For completeness, in Appendix H, we describe also constructions of secure predicate encryption for Boolean predicates that can be expressed as k-CNF and k-DNF (for any constant k).

We also stress that the two computational assumptions on which we base our proof of security are very natural. Specifically, our two assumptions posit the difficulty of a subgroup decision problem and of a problem that can be seen as the generalization of Decision Diffie-Hellman to groups of composite order.

Related Work. The first implementation of HVE is due to [6] that proved the security of their construction under assumptions on bilinear groups of composite order in the selective model. In this security model (introduced by [7] in the context of IBE), the adversary must commit to its challenge vectors before seeing the public key of the HVE scheme. In a recent series of papers Waters [26] and Lewko and Waters [18] introduced the concept of a dual system encryption scheme that was used to construct efficient and fully secure Identity Based Encryption (IBE) and Hierarchical IBE from simple assumptions. Previous fully secure construction of these primitives either used a

partitioning strategy (see [2],[25]) or used complexity assumptions of non-constant size (see [9],[10]). Partitioning strategy and the approaches of [9] and [10] do not seem to be helpful in proving full security of more complex primitives like HVE.

For HVE, fully secure constructions of HVE can can be derived, via the reduction given in [14], from the fully secure constructions for inner-product encryption given by [17, 19]. The resulting constructions are proved secure against adversaries that are allowed to ask only *non-matching* key queries; that is, key queries for vectors $\vec{y}$ such that $\mathsf{Match}(\vec{x}_0, \vec{y}) = \mathsf{Match}(\vec{x}_1, \vec{y}) = 0$. We call queries for vectors $\vec{y}$ such that $\mathsf{Match}(\vec{x}_0, \vec{y}) = \mathsf{Match}(\vec{x}_1, \vec{y}) = 1$ *matching*. Our construction poses no restriction on the queries that adversaries can ask. The limitation of having security against non-matching adversaries was already pointed out in [14] and [23] described a scheme secure against non-matching adversaries that was not secure against unrestricted adversaries. Our security notion is game-based. Recently, Boneh et al. [5] and O'Neill [20] investigated simulation-based security notions for IBE and predicate encryption.

Proof technique. Our result is based on the dual system encryption methodology introduced by Waters [26] and gives extra evidence of the power of this proof technique. However, to overcome the difficulty of having to deal also with matching queries, we have to carefully look at the space of matching queries and at how they relate to the challenge vectors. This enables us to craft a new security game in which the challenge ciphertext is constructed in a way that guarantees that keys obtained by the adversary give the expected result when tested against the challenge ciphertext and, at the same time, the challenge ciphertext is independent from the challenge vector used to construct it. Then we show, by means of a sequence of intermediate security games, that the real security game is computationally indistinguishable from this new game. It is not immediate to obtain a similar relation between matching queries and challenge vectors for other predicates (e.g., inner product) that would make our approach viable.

2 Hidden Vector Encryption

In this section we give formal definitions for Hidden Vector Encryption (HVE) and its security properties. For sake of simplicity, we present predicate-only definitions and constructions for HVE instead of full-fledged ones. In Appendix G we will briefly discuss how to extend our scheme to the full-fledged version. For the same reason, we give our definitions and constructions for binary alphabets. In Appendix C we discuss how to extend our work to general alphabets.

Following standard terminology, we call a function $\nu(\lambda)$ *negligible* if for all constants $c > 0$ and sufficiently large λ, $\nu(\lambda) < 1/\lambda^c$ and denote by $[n]$ the set of integers $\{1, \ldots, n\}$. Moreover the writing "$a \leftarrow A$", for a finite set A, denotes that a is randomly and uniformly selected from A.

2.1 Hidden Vector Encryption

Let $\vec{x}$ be a binary vector of length ℓ and $\vec{y}$ a vector of the same length over $\{0, 1, \star\}$. We remind that predicate $\mathsf{Match}(\vec{x}, \vec{y})$ is defined to be true if and only if the two vectors agree in all positions i where $y_j \neq \star$. A Hidden Vector Encryption scheme is a tuple of four efficient probabilistic algorithms (Setup, $\mathsf{Encrypt}$, KeyGen, Test) with the following semantics.

$\mathsf{Setup}(1^\lambda, 1^\ell)$: takes as input a security parameter λ and a length parameter ℓ (given in unary), and outputs the public parameters Pk and the master secret key Msk.

$\mathsf{KeyGen}(\mathsf{Msk}, \vec{y})$: takes as input the master secret key Msk and a vector $\vec{y} \in \{0,1,\star\}^\ell$, and outputs a secret key $\mathsf{Sk}_{\vec{y}}$.

$\mathsf{Encrypt}(\mathsf{Pk}, \vec{x})$: takes as input the public parameters Pk and a vector $\vec{x} \in \{0,1\}^\ell$ and outputs a ciphertext Ct.

$\mathsf{Test}(\mathsf{Pk}, \mathsf{Ct}, \mathsf{Sk}_{\vec{y}})$: takes as input the public parameters Pk, a ciphertext Ct encrypting $\vec{x}$ and a secret key $\mathsf{Sk}_{\vec{y}}$ and outputs $\mathsf{Match}(\vec{x}, \vec{y})$.

For correctness we require that, for pairs $(\mathsf{Pk}, \mathsf{Msk})$ output by $\mathsf{Setup}(1^\lambda, 1^\ell)$, it holds that for all vectors $\vec{x} \in \{0,1\}^\ell$ and $\vec{y} \in \{0,1,\star\}^\ell$, we have that $\mathsf{Test}(\mathsf{Pk}, \mathsf{Encrypt}(\mathsf{Pk}, \vec{x}), \mathsf{KeyGen}(\mathsf{Msk}, \vec{y})) = \mathsf{Match}(\vec{x}, \vec{y})$ except with negligible in λ probability.

2.2 Security definitions for HVE

In this section we formalize our security requirement by means of a security game GReal between a probabilistic polynomial time adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. GReal consists of a Setup phase and of a Query Answering phase. In the Query Answering phase, the adversary can issue a polynomial number of Key Queries and one Challenge Construction query and at the end of this phase $\mathcal{A}$ outputs a guess. We stress that key queries can be issued by $\mathcal{A}$ even after he has received the challenge from $\mathcal{C}$. In GReal the adversary is restricted to queries for vectors $\vec{y}$ such that $\mathsf{Match}(\vec{y}, x_0) = \mathsf{Match}(\vec{y}, x_1)$.

More precisely, we define game GReal in the following way.

Setup. $\mathcal{C}$ runs the Setup algorithm on input the security parameter λ and the length parameter ℓ (given in unary) to generate public parameters Pk and master secret key Msk. $\mathcal{C}$ starts the interaction with $\mathcal{A}$ on input Pk.

Key Query Answering$(\vec{y})$. $\mathcal{C}$ returns the key for $\vec{y}$ computed by executing $\mathsf{KeyGen}(\mathsf{Msk}, \vec{y})$.

Challenge Query Answering$(\vec{x}_0, \vec{x}_1)$. $\mathcal{C}$ picks random $\eta \in \{0,1\}$ and returns the challenge ciphertext computed by executing $\mathsf{Encrypt}(\mathsf{Pk}, \vec{x}_\eta)$.

Winning Condition. Let η' be $\mathcal{A}$'s output. We say that $\mathcal{A}$ *wins* the game if $\eta = \eta'$ and for all $\vec{y}$ for which $\mathcal{A}$ has issued a Key Query, it holds $\mathsf{Match}(\vec{x}_0, \vec{y}) = \mathsf{Match}(\vec{x}_1, \vec{y})$.

We define the advantage $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{HVE}}(\lambda)$ of $\mathcal{A}$ in GReal to be the probability of winning minus $1/2$.

Definition 1. *An Hidden Vector Encryption scheme is* secure *if for all probabilistic polynomial time adversaries $\mathcal{A}$, we have that $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{HVE}}(\lambda)$ is a negligible function of λ.*

It is trivial to observe that an adversary that possesses a secret key for a vector $\vec{y}$ such that $\mathsf{Match}(\vec{y}, \vec{x}_0) \neq \mathsf{Match}(\vec{y}, \vec{x}_1)$ has probability 1 of guessing η.

3 Complexity Assumptions

We work with *symmetric* bilinear groups of composite order. Our construction can be adapted to the asymmetric setting in a straightforward way. Composite order bilinear groups were first used in Cryptography by [4] (see also [1]). We suppose the existence of an efficient group generator algorithm $\mathcal{G}$ which takes as input the security parameter λ and outputs a description $\mathcal{I} = (N, \mathbb{G}, \mathbb{G}_T, \mathbf{e})$ of a bilinear setting, where $\mathbb{G}$ and $\mathbb{G}_T$ are cyclic groups of order N, and $\mathbf{e} : \mathbb{G}^2 \to \mathbb{G}_T$ is a map with the following properties:

1. (Bilinearity) $\forall\ g, h \in \mathbb{G}$ and $a, b \in \mathbb{Z}_N$ it holds that $\mathbf{e}(g^a, h^b) = \mathbf{e}(g, h)^{ab}$.

2. (Non-degeneracy) $\exists\ g \in \mathbb{G}$ such that $\mathbf{e}(g,g)$ has order N in $\mathbb{G}_T$.

We assume that the group descriptions of $\mathbb{G}$ and $\mathbb{G}_T$ include generators of the respective cyclic subgroups. We require that the group operations in $\mathbb{G}$ and $\mathbb{G}_T$ as well as the bilinear map $\mathbf{e}$ are computable in deterministic polynomial time in λ. In our construction we will make hardness assumptions for bilinear settings whose order N is product of four distinct primes each of length $\Theta(\lambda)$. For an integer m dividing N, we let $\mathbb{G}_m$ denote the subgroup of $\mathbb{G}$ of order m. From the fact that the group is cyclic, it is easy to verify that if g and h are group elements of co-prime orders then $\mathbf{e}(g,h) = 1$. This is called the *orthogonality* property and is a crucial tool in our constructions.

We are now ready to give our complexity assumptions.

Assumption 1**.** The first assumption is a subgroup-decision type assumption for bilinear settings. Specifically, Assumption 1 posits the difficulty of deciding whether an element belongs to one of two specified subgroups, even when generators of some of the subgroups of the bilinear group are given. More formally, we have the following definition.

First pick a random bilinear setting $\mathcal{I} = (N = p_1p_2p_3p_4, \mathbb{G}, \mathbb{G}_T, \mathbf{e}) \leftarrow \mathcal{G}(1^\lambda)$ and then pick $A_3 \leftarrow \mathbb{G}_{p_3},\ A_{13} \leftarrow \mathbb{G}_{p_1p_3},\ A_{12} \leftarrow \mathbb{G}_{p_1p_2},\ A_4 \leftarrow\in \mathbb{G}_{p_4},\ T_1 \leftarrow \mathbb{G}_{p_1p_3},\ T_2 \leftarrow \mathbb{G}_{p_2p_3}$, and set $D = (\mathcal{I}, A_3, A_4, A_{13}, A_{12})$. We define the advantage of an algorithm $\mathcal{A}$ in breaking Assumption 1 to be

$$\mathsf{Adv}_1^{\mathcal{A}}(\lambda) = |\mathrm{Prob}[\mathcal{A}(D,T_1) = 1] - \mathrm{Prob}[\mathcal{A}(D,T_2) = 1]|$$

Assumption 1. *We say that Assumption* 1 *holds for generator $\mathcal{G}$ if for all probabilistic polynomial-time algorithms $\mathcal{A}$, $\mathsf{Adv}_1^{\mathcal{A}}(\lambda)$ is a negligible function of λ.*

Assumption 2**.** Our second assumption can be seen as the Decision Diffie-Hellman Assumption for composite order groups. Specifically, Assumption 2 posits the difficult of deciding if a triple of elements constitute a Diffie-Hellman triplet with respect to one of the factors of the order of the group, even when given, for each prime divisor p of the group order, a generator of the subgroup of order p. Notice that for bilinear groups of prime order the Diffie-Hellman assumption does not hold. More formally, we have the following definition.

First pick a random bilinear setting $\mathcal{I} = (N = p_1p_2p_3p_4, \mathbb{G}, \mathbb{G}_T, \mathbf{e}) \leftarrow \mathcal{G}(1^\lambda)$ and then pick $A_1 \leftarrow \mathbb{G}_{p_1}, A_2 \leftarrow \mathbb{G}_{p_2}, A_3 \leftarrow \mathbb{G}_{p_3}, A_4, B_4, C_4, D_4 \leftarrow \mathbb{G}_{p_4}, \alpha, \beta \leftarrow \mathbb{Z}_{p_1}, T_2 \leftarrow \mathbb{G}_{p_1p_4}$, and set $T_1 = A_1^{\alpha\beta} \cdot D_4$ and $D = (\mathcal{I}, A_1, A_2, A_3, A_4, A_1^{\alpha} \cdot B_4, A_1^{\beta} \cdot C_4)$. We define the advantage of an algorithm $\mathcal{A}$ in breaking Assumption 2 to be

$$\mathsf{Adv}_2^{\mathcal{A}}(\lambda) = |\mathrm{Prob}[\mathcal{A}(D,T_1) = 1] - \mathrm{Prob}[\mathcal{A}(D,T_2) = 1]|$$

Assumption 2. *We say that Assumption* 2 *holds for generator $\mathcal{G}$ if for all probabilistic polynomial-time algorithms $\mathcal{A}$, $\mathsf{Adv}_2^{\mathcal{A}}(\lambda)$ is a negligible function of λ.*

In Appendix E, we prove that Assumption 1 and 2 hold in the generic group model.

4 Constructing HVE

In this section we describe our HVE scheme. To make our description and proofs simpler, we add to all vectors $\vec{x}$ and $\vec{y}$ two dummy components and set both of them equal to 0. We can thus assume that all vectors have at least two non-star positions.

$\mathsf{Setup}(1^\lambda, 1^\ell)$: The setup algorithm chooses a description of a bilinear group $\mathcal{I} = (N = p_1p_2p_3p_4, \mathbb{G}, \mathbb{G}_T, \mathbf{e})$ with known factorization by running a generator algorithm $\mathcal{G}$ on input 1^λ. The setup algorithm chooses random $g_1 \in \mathbb{G}_{p_1}$, $g_2 \in \mathbb{G}_{p_2}$, $g_3 \in \mathbb{G}_{p_3}$, $g_4 \in \mathbb{G}_{p_4}$, and, for $i \in [\ell]$ and $b \in \{0,1\}$, random $t_{i,b} \in Z_N$ and random $R_{i,b} \in \mathbb{G}_{p_3}$ and sets $T_{i,b} = g_1^{t_{i,b}} \cdot R_{i,b}$.

The public parameters are $\mathsf{Pk} = [N, g_3, (T_{i,b})_{i\in[\ell],b\in\{0,1\}}]$ and the master secret key is $\mathsf{Msk} = [g_{12}, g_4, (t_{i,b})_{i\in[\ell],b\in\{0,1\}}]$, where $g_{12} = g_1 \cdot g_2$.

$\mathsf{KeyGen}(\mathsf{Msk}, \vec{y})$: Let $S_{\vec{y}}$ be the set of indices i such that $y_i \neq \star$. The key generation algorithm chooses random $a_i \in \mathbb{Z}_N$ for $i \in S_{\vec{y}}$ under the constraint that $\sum_{i\in S_{\vec{y}}} a_i = 0$. For $i \in S_{\vec{y}}$, the algorithm chooses random $W_i \in \mathbb{G}_{p_4}$ and sets

$$Y_i = g_{12}^{a_i/t_{i,y_i}} \cdot W_i.$$

The algorithm returns the tuple $(Y_i)_{i\in S_{\vec{y}}}$. Here we use the fact that $S_{\vec{y}}$ has size at least 2.

$\mathsf{Encrypt}(\mathsf{Pk}, \vec{x})$: The encryption algorithm chooses random $s \in \mathbb{Z}_N$. For $i \in [\ell]$, the algorithm chooses random $Z_i \in \mathbb{G}_{p_3}$ and sets

$$X_i = T_{i,x_i}^s \cdot Z_i,$$

and returns the tuple $(X_i)_{i\in[\ell]}$.

$\mathsf{Test}(\mathsf{Ct}, \mathsf{Sk}_{\vec{y}})$: The test algorithm computes $T = \prod_{i\in S_{\vec{y}}} \mathbf{e}(X_i, Y_i)$. It returns TRUE if $T = 1$, FALSE otherwise.

Correctness In Appendix B we prove that the above scheme is correct.

5 Security of our HVE scheme

We start by giving an informal description of the ideas behind our proof of security and show how we overcome the main technical difficulty of having to deal with adversaries that possess keys that match the challenge ciphertext.

The first step of our proof strategy consists in projecting the public key (and thus the ciphertexts the adversary constructs by himself) to a different subgroup from the one of the challenge ciphertext. Specifically, we defined a new security game GPK in which the $t_{i,b}$'s are encoded in the $\mathbb{G}_{p_2}$ part of the $T_{i,b}$'s from the public key (instead of the the $\mathbb{G}_{p_1}$ part as in the real game). The challenge ciphertext and the answers to the key queries are instead constructed as in the real security game GReal. Thus, ciphertexts constructed by the adversary are completely independent from the challenge ciphertext (as they encode information in two different subgroups). We observe that since keys are constructed as in the real security game, they carry information about $\vec{y}$ both in the $\mathbb{G}_{p_1}$ and $\mathbb{G}_{p_2}$ parts. Thus when the adversary tests a ciphertext he has constructed by using the public key against a key obtained by means of a query, he obtains the expected result because of the information encoded in the $\mathbb{G}_{p_2}$ part of the key and of the ciphertext. The same holds for the challenge ciphertext but in this case thanks to the $\mathbb{G}_{p_1}$ part of the key. The only difference between the two games is in the public key but, under Assumption 1 (a natural subgroup decision hardness assumption), we can prove that the two games are indistinguishable.

The second step proves that the keys obtained from queries do not help the adversary. Since the challenge ciphertext carries information about the randomly selected challenge vector $\vec{x}_\eta$ in its $\mathbb{G}_{p_1}$

Game $\mathsf{GPK}(\lambda, \ell)$

Setup. $\mathcal{C}$ chooses a description of a bilinear group $\mathcal{I} = (N = p_1p_2p_3p_4, \mathbb{G}, \mathbb{G}_T, \mathbf{e})$ with known factorization by running a generator algorithm $\mathcal{G}$ on input 1^λ. $\mathcal{C}$ chooses random $g_1 \in \mathbb{G}_{p_1}, g_2 \in \mathbb{G}_{p_2}, g_3 \in \mathbb{G}_{p_3}, g_4 \in \mathbb{G}_{p_4}$ and sets $g_{12} = g_1 \cdot g_2$. For each $i \in [\ell]$ and $b \in \{0,1\}$, $\mathcal{C}$ chooses random $t_{i,b} \in Z_N$ and $R_{i,b} \in \mathbb{G}_{p_3}$ and sets $T'_{i,b} = g_1^{t_{i,b}} \cdot R_{i,b}$ and $T_{i,b} = g_2^{t_{i,b}} \cdot R_{i,b}$. Then $\mathcal{C}$ sets $\mathsf{Pk} = [N, g_3, (T_{i,b})_{i\in[\ell],b\in\{0,1\}}]$, $\mathsf{Pk}' = [N, g_3, (T'_{i,b})_{i\in[\ell],b\in\{0,1\}}]$, and $\mathsf{Msk} = [g_{12}, g_4, (t_{i,b})_{i\in[\ell],b\in\{0,1\}}]$. Finally, $\mathcal{C}$ starts the interaction with $\mathcal{A}$ on input Pk.

Key Query Answering$(\vec{y})$. $\mathcal{C}$ returns the output of $\mathsf{KeyGen}(\mathsf{Msk}, \vec{y})$.

Challenge Query Answering$(\vec{x}_0, \vec{x}_1)$. Upon receiving the pair $(\vec{x}_0, \vec{x}_1)$ of challenge vectors, $\mathcal{C}$ picks random $\eta \in \{0,1\}$ and returns the output of $\mathsf{Encrypt}(\mathsf{Pk}', \vec{x}_\eta)$.

Winning Condition. Like in $\mathsf{GReal}(\lambda, \ell)$.

Figure 1: A formal description of GPK.

part, in this informal discussion when we refer to key we mean its $\mathbb{G}_{p_1}$ part. The $\mathbb{G}_{p_2}$ parts of the keys are always correctly computed.

In our construction, testing a ciphertext against a non-matching key gives a random value (from the target group) whereas testing it against a matching key returns a specified value (the identity of the target group). If we had to prove security against an adversary that asked only non-matching queries we could consider the experiment in which key queries were replied by returning a key with random $\mathbb{G}_{p_1}$ parts. Such a game can be proved indistinguishable from GPK (under an appropriate complexity assumption) and it is easy to prove that it gives no advantage to an adversary. This approach fails for matching queries as such a key will return the wrong answer with high probability when tested against the challenge ciphertext. Instead we modify the construction of the challenge ciphertext in the following way: the challenge ciphertext is well-formed in all the positions where the two challenge vectors are equal and random in all the other positions. We observe that testing such a challenge ciphertext against matching and non-matching keys always gives the correct answer and that no adversary (even an all powerful one) can guess which of the two challenge vectors has been used to construct the challenge ciphertext (see the discussion in Section 5.2.1).

5.1 The first step of the proof

We start by defining game $\mathsf{GPK}(\lambda, \ell)$ (see Figure 1) that differs from $\mathsf{GReal}(\lambda, \ell)$ as in the Setup phase, $\mathcal{C}$ prepares two sets of public parameters, Pk and Pk', and one master secret key Msk. Pk is given as input to $\mathcal{A}$, Msk is used to answer $\mathcal{A}$'s key queries and Pk' is used to construct the challenge ciphertext.

The next lemma shows that, the advantages of an adversary in $\mathsf{GReal}(\lambda, \ell)$ and $\mathsf{GPK}(\lambda, \ell)$ are the same, up to a negligible factor.

Lemma 2. *If Assumption* 1 *holds, then for any PPT adversary* $\mathcal{A}$, $\left|\mathsf{Adv}^{\mathcal{A}}[\mathsf{GReal}(\lambda, \ell)] - \mathsf{Adv}^{\mathcal{A}}[\mathsf{GPK}(\lambda, \ell)]\right|$ *is negligible.*

The proof is found in Appendix F.1.

Game $\mathsf{GBadQ}(f,k)$

Setup. Like in GPK. That is, $\mathcal{C}$ chooses a description of a bilinear group $\mathcal{I} = (N = p_1p_2p_3p_4, \mathbb{G}, \mathbb{G}_T, \mathbf{e})$ with known factorization by running a generator algorithm $\mathcal{G}$ on input 1^λ. $\mathcal{C}$ chooses random $g_1 \in \mathbb{G}_{p_1}, g_2 \in \mathbb{G}_{p_2}, g_3 \in \mathbb{G}_{p_3}, g_4 \in \mathbb{G}_{p_4}$ and sets $g_{12} = g_1 \cdot g_2$. For each $i \in [\ell]$ and $b \in \{0,1\}$, $\mathcal{C}$ chooses random $t_{i,b} \in Z_N$ and $R_{i,b} \in \mathbb{G}_{p_3}$ and sets $T'_{i,b} = g_1^{t_{i,b}} \cdot R_{i,b}$ and $T_{i,b} = g_2^{t_{i,b}} \cdot R_{i,b}$. Then $\mathcal{C}$ sets $\mathsf{Pk} = [N, g_3, (T_{i,b})_{i\in[\ell],b\in\{0,1\}}]$, $\mathsf{Pk}' = [N, g_3, (T'_{i,b})_{i\in[\ell],b\in\{0,1\}}]$, and $\mathsf{Msk} = [g_{12}, g_4, (t_{i,b})_{i\in[\ell],b\in\{0,1\}}]$. Finally, $\mathcal{C}$ starts the interaction with $\mathcal{A}$ on input Pk.

Answering Key-Query for $\vec{y} = \langle y_1, \ldots, y_\ell \rangle$.
$\mathcal{C}$ answers the first k queries in the following way.

- If $y_f \neq \star$, $\mathcal{C}$ returns a key whose $\mathbb{G}_{p_1}$ parts is random.
  More specifically, $\mathcal{C}$ chooses, for each $i \in S_{\vec{y}}$, random $W_i \in \mathbb{G}_{p_4}$, random $C_i \in \mathbb{G}_{p_1}$ and random $a_i \in \mathbb{Z}_N$ under the constraint that $\sum_{i\in S_y} a_i = 0$ and sets $Y_i = C_i \cdot g_2^{a_i/t_{i,y_i}} \cdot W_i$.
- If $y_f = \star$ then $\mathcal{C}$ returns the output of $\mathsf{KeyGen}(\vec{y}, \mathsf{Msk})$.

The remaining $q-k$ queries are answered by running $\mathsf{KeyGen}(\vec{y}, \mathsf{Msk})$.

Answering Challenge Query for $(\vec{x}_0, \vec{x}_1)$. $\mathcal{C}$ chooses random $s \in \mathbb{Z}_N$ and $\eta \in \{0,1\}$ and sets $\vec{x} = \vec{x}_\eta$. For each $i \in [f-1]$ such that $\vec{x}_{0,i} \neq \vec{x}_{1,i}$, $\mathcal{C}$ chooses random $X_i \in \mathbb{G}_{p_1p_3}$. Then, for each remaining i, $\mathcal{C}$ chooses random $Z_i \in \mathbb{G}_{p_3}$ and sets $X_i = T'^{s}_{i,x_i} \cdot Z_i$. $\mathcal{C}$ returns the tuple $(X_i)_{i\in[\ell]}$.

Winning Condition. Like in GReal.

Figure 2: A formal description of game $\mathsf{GBadQ}(f,k)$.

5.2 The second step of the proof

We start the second step of the proof by describing in Figure 2, for $1 \leq f \leq \ell+1$ and $0 \leq k \leq q$, game $\mathsf{GBadQ}(f,k)$ between the challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ that asks q queries. Not to overburden our notation, we omitted λ and ℓ from the name of the games. $\mathsf{GBadQ}(f,k)$ differs from GPK both in the way in which key queries are answered and in the way in which the challenge ciphertext is constructed. Specifically, in $\mathsf{GBadQ}(f,k)$ the first k key queries are answered by distinguishing two cases. Queries for $\vec{y}$ such that $y_f = \star$ are answered by running $\mathsf{KeyGen}(\mathsf{Msk}, \vec{y})$. Instead queries for $\vec{y}$ such that $y_f \neq \star$ are answered by returning keys whose $\mathbb{G}_{p_1}$ part is random for all components. Moreover, in $\mathsf{GBadQ}(f,k)$, the $\mathbb{G}_{p_1}$ part of the first $f-1$ components of the challenge ciphertext corresponding to positions in which the two challenges differ are random.

In the proofs, we will use the shorthand $\mathsf{GBadCh}(f)$ to denote the game $\mathsf{GBadQ}(f,0)$ in which only the challenge ciphertext is modified whereas all the replies to the key queries are correctly computed. Moreover, we define $\mathsf{GBadQ2}(f,k)$, for $1 \leq f \leq \ell$ and $0 \leq k \leq q$, as a game in which the setup phase is like in $\mathsf{GBadQ}(f,k)$, key queries are answered like in $\mathsf{GBadQ}(f,k)$ and the challenge ciphertext is constructed like in $\mathsf{GBadQ}(f+1,k)$.

5.2.1 Some simple observations about GBadQ and GBadQ2

Observation 3. $\mathsf{GPK} = \mathsf{GBadQ}(1,0) = \mathsf{GBadCh}(1)$.

Straightforward from the definitions of the games.

Observation 4. $\mathsf{GBadQ}(f,q) = \mathsf{GBadQ2}(f,q)$ *for* $f = 1,\ldots,\ell$.
From the definitions of the two games, it is clear that all key queries are answered in the same way in both the games and all components X_i for $i \neq f$ of the challenge ciphertext are computed in the same way. Let us now look at X_f and more precisely to its $\mathbb{G}_{p_1}$ part. In $\mathsf{GBadQ}(f,q)$, the $\mathbb{G}_{p_1}$ part of X_f is computed as T'^{s}_{f,x_f} which is exactly how it is computed in $\mathsf{GBadQ2}(f,q)$ when $x_{0,f} = x_{1,f}$. On the other hand, when $x_{0,f} \neq x_{1,f}$, the $\mathbb{G}_{p_1}$ part of X_f is chosen at random. However, observe that exponents $t_{f,0} \mod p_1$ and $t_{f,1} \mod p_1$ have not appeared in the answers to key queries since every query has either a $\star$ in position f (in which case position f of the answer is empty) or a non-$\star$ value in position f (in which case the $\mathbb{G}_{p_1}$ part of the position f of the answer is random since $k = q$). Therefore, we can conclude that the $\mathbb{G}_{p_1}$ part of the component X_f of the answer to the challenge query is also random in $\mathbb{G}_{p_1}$.

Observation 5. $\mathsf{GBadQ2}(f,0) = \mathsf{GBadQ}(f+1,0)$ *for* $f = 1,\ldots,\ell-1$.
Indeed, in both games all key queries are answered correctly, and the challenge query in $\mathsf{GBadQ2}(f,0)$ is by definition answered in the same way as in $\mathsf{GBadQ}(f+1,0)$.

Observation 6. *For* $f = 1,\ldots,\ell-1$*, if* $x_{0,f} = x_{1,f}$*,* $\mathsf{GBadCh}(f) = \mathsf{GBadCh}(f+1)$.
By definition, in $\mathsf{GBadCh}(f) = \mathsf{GBadQ}(f,0)$ the f-th component of the challenge ciphertext is well formed, namely $X_f = T'^{s}_{f,x_f} \cdot Z_f$. This is the same in $\mathsf{GBadCh}(f+1) = \mathsf{GBadQ}(f+1,0)$ under the condition that $x_{0,f} = x_{1,f}$.

Observation 7. *All adversaries have no advantage in* $\mathsf{GBadCh}(\ell+1) = \mathsf{GBadQ}(\ell+1,0)$.
This follows from the fact that, for positions i such that $x_{0,i} \neq x_{1,i}$, the $\mathbb{G}_{p_1}$ part of X_i is random. Thus the challenge ciphertext of $\mathsf{GBadCh}(\ell+1)$ is independent from η.

Overview of the second step of the proof. Consider the sequence

$$\mathsf{GPK} = \mathsf{GBadCh}(1), \mathsf{GBadCh}(2), \ldots, \mathsf{GBadCh}(\ell), \mathsf{GBadCh}(\ell+1)$$

of $\ell+1$ experiments. By Observation 7, if an adversary $\mathcal{A}$ has a non negligible advantage in GPK then it must be the case that there exists $1 \leq f \leq \ell$ such that the difference between $\mathcal{A}$'s advantages in $\mathsf{GBadCh}(f)$ and $\mathsf{GBadCh}(f+1)$ is non-negligible. Moreover, by Observation 6, for this to happen it must be the case that $\mathcal{A}$ has non-negligible probability to output two challenges that differ in the f-th component. Then, if $\mathcal{A}$ makes q key queries, consider the following sequence

$$\begin{array}{l} \mathsf{GBadCh}(f) \\ \quad = \\ \mathsf{GBadQ}(f,0)\ldots\mathsf{GBadQ}(f,q-1)\ \mathsf{GBadQ}(f,q) = \mathsf{GBadQ2}(f,q)\ \mathsf{GBadQ2}(f,q-1)\ldots\mathsf{GBadQ2}(f,0) \\ \hfill = \\ \hfill \mathsf{GBadCh}(f+1) \end{array}$$

of $2q+1$ games. If the difference in advantage between $\mathsf{GBadCh}(f)$ and $\mathsf{GBadCh}(f+1)$ is non-negligible, then there must exist k such that the difference in advantage between either $\mathsf{GBadQ}(f,k)$ and $\mathsf{GBadQ}(f,k-1)$ or between $\mathsf{GBadQ2}(f,k)$ and $\mathsf{GBadQ2}(f,k-1)$ is non-negligible.

Now it seems that we are stuck as, for all f and k, games $\mathsf{GBadQ}(f,k-1)$ and $\mathsf{GBadQ}(f,k)$ can be distinguished by an adversary $\mathcal{A}$ using the following simple strategy. $\mathcal{A}$ prepares two challenges

$\vec{x}_0$ and $\vec{x}_1$ that coincide in the f-th component and asks as k-th key query the key Y for a vector $\vec{y}^{(k)}$ such that $\vec{y}_f^{(k)} = x_{0,f} = x_{1,f}$ and $\mathsf{Match}(\vec{x}_0, \vec{y}^{(k)}) = \mathsf{Match}(\vec{x}_1, \vec{y}^{(k)}) = 1$. Let X be the challenge ciphertext. Now, in $\mathsf{GBadQ}(f, k-1)$, the answer Y received by $\mathcal{A}$ to its k-th query is well-formed and thus $\mathsf{Test}(X, Y) = 1$. Instead in $\mathsf{GBadQ}(f, k)$, Y is random and thus $\mathsf{Test}(X, Y) = 0$ except with negligible probability. The above strategy requires the two challenges to coincide in the f-th component. Indeed, if $x_{0,f} \neq x_{1,f}$ then for $\vec{y}^{(k)}$ to be matching it must the case that $y_f = \star$ but then in this case Y is well-formed in both games. This perfectly fits our strategy as we have to prove that $\mathsf{GBadQ}(f, k)$ is indistinguishable from $\mathsf{GBadQ}(f, k-1)$ only for f for which $\mathcal{A}$ has a non-negligible probability of outputting two challenges that differ in the f-th component. Exactly, the same reasoning holds for $\mathsf{GBadQ2}$.

In the next section we describe a simulator $\mathcal{S}$ that takes as input the pair of integers f and k and an instance of Assumption 2 and, provided that the adversary does not output two challenges that coincide in the f-th component, simulates with some non-negligible probability $\mathsf{GBadQ}(f, k)$ or $\mathsf{GBadQ}(f, k-1)$ depending on the nature of the challenge. A similar simulator can be constructed for games $\mathsf{GBadQ2}(f, k)$ and $\mathsf{GBadQ2}(f, k-1)$. For the rest of the proof of security it is convenient to refer to the alternative and equivalent description of our HVE found in Section A.

5.2.2 Description of simulator $\mathcal{S}$

Input to $\mathcal{S}$. Integers $1 \leq f \leq \ell + 1$ and $0 \leq k \leq q$, and a randomly chosen instance (D, T) of Assumption 2; recall that $D = (\mathcal{I}, A_1, A_2, A_3, A_4, A_1^{\alpha} \cdot B_4, A_1^{\beta} \cdot C_4)$ and $T = T_1 = A_1^{\alpha\beta} \cdot D_4$ or $T = T_2$ random in $\mathbb{G}_{p_1 p_4}$.

Setup. To simulate the Setup phase $\mathcal{S}$ executes the following steps.

1. $\mathcal{S}$ sets $g_1 = A_1$, $g_2 = A_2$, $g_3 = A_3$, $g_4 = A_4$ and $g_{12} = A_1 \cdot A_2$.
2. For each $i \in [\ell]$ and $b \in \{0, 1\}$,
   $\mathcal{S}$ chooses random $v_{i,b} \in \mathbb{Z}_N$ and $R_{i,b} \in \mathbb{G}_{p_3}$, and sets $T_{i,b} = g_2^{v_{i,b}} \cdot R_{i,b}$.
3. $\mathcal{S}$ sets $\mathsf{Pk} = [N, g_3, (T_{i,b})_{i \in [\ell], b \in \{0,1\}}]$.
4. $\mathcal{S}$ picks random $\hat{\jmath} \in [\ell]$ and $\hat{b} \in \{0, 1\}$ and sets $\hat{c} = 1 - \hat{b}$.
5. For each $i \in [\ell] \setminus \{\hat{\jmath}\}$ and $b \in \{0, 1\}$,
   $\mathcal{S}$ chooses random $r_{i,b} \in \mathbb{Z}_N$ and $R'_{i,b} \in \mathbb{G}_{p_3}$.
   $\mathcal{S}$ sets $T'_{i,b} = g_1^{r_{i,b}} \cdot R'_{i,b}$.
6. $\mathcal{S}$ chooses random $r_{\hat{\jmath},\hat{c}} \in \mathbb{Z}_N$ and $R'_{\hat{\jmath},\hat{c}} \in \mathbb{G}_{p_3}$.
   $\mathcal{S}$ sets $T'_{\hat{\jmath},\hat{c}} = g_1^{r_{\hat{\jmath},\hat{c}}} \cdot R'_{\hat{\jmath},\hat{c}}$, $T'_{\hat{\jmath},\hat{b}} = \perp$ and $r_{\hat{\jmath},\hat{b}} = \perp$.
7. $\mathcal{S}$ sets $\mathsf{Pk}' = [N, g_3, (T'_{i,b})_{i \in [\ell], b \in \{0,1\}}]$ and $\mathsf{Msk} = [g_{12}, g_4, (r_{i,b}, v_{i,b})_{i \in [\ell], b \in \{0,1\}}]$.

Notice that the values $r_{\hat{\jmath},\hat{b}}$ and $T'_{\hat{\jmath},\hat{b}}$ are unspecified and thus Pk' and Msk are incomplete. As we shall see below, in answering key queries, $\mathcal{S}$ will implicitly set $r_{\hat{\jmath},\hat{b}} = 1/\beta$. Here β is the exponent of A_1 in $A_1^{\beta} \cdot C_4$ from instance D of Assumption 2 and we stress that $\mathcal{S}$ does not have access to the actual value of β.

$\mathcal{S}$ starts the interaction with $\mathcal{A}$ on input Pk.

Answering Key Query for $\vec{y} = \langle y_1, \ldots, y_\ell \rangle$.

- **First $k-1$ key queries**.

  We distinguish the following two mutually exclusive cases.

Case A.1: $y_f \neq \star$**.** In this case, $\mathcal{S}$ outputs a key whose $\mathbb{G}_{p_1}$ part is random. More precisely, $\mathcal{S}$ executes the following steps. For each $i \in S_{\vec{y}}$, $\mathcal{S}$ chooses random a''_i such that $\sum_{i \in S_{\vec{y}}} a''_i = 0$, random $C_i \in \mathbb{G}_{p_1}$, and random $W_i \in \mathbb{G}_{p_4}$. Then, for each $i \in S_{\vec{y}}$, $\mathcal{S}$ sets

$$Y_i = C_i \cdot g_2^{a''_i / v_{i,y_i}} \cdot W_i.$$

Case A.2: $y_f = \star$. In this case, $\mathcal{S}$ outputs a key that has the same distribution induced by algorithm KeyGen on input $\vec{y}$ and Msk. We observe that if $y_{\hat{\jmath}} = \hat{c}$ then Msk includes all the r_{i,y_i}'s and v_{i,y_i}'s that are needed. If instead $y_{\hat{\jmath}} = \hat{b}$, then Msk is missing $r_{\hat{\jmath},\hat{b}}$. In this case $\mathcal{S}$ computes $Y_{\hat{\jmath}}$ by using $A_1^{\beta} \cdot C_4$ from the challenge D of Assumption 2 received in input.

More precisely, for each $i \in S_{\vec{y}}$, $\mathcal{S}$ picks random $W_i \in \mathbb{G}_{p_4}$ and random $a'_i, a''_i \in \mathbb{Z}_N$ under the constraint that $\sum_{i \in S_{\vec{y}}} a'_i = \sum_{i \in S_{\vec{y}}} a''_i = 0$. Then for each $i \neq \hat{\jmath}$, $\mathcal{S}$ sets

$$Y_i = g_1^{a'_i / r_{i,y_i}} \cdot g_2^{a''_i / v_{i,y_i}} \cdot W_i.$$

Moreover, if $y_{\hat{\jmath}} = \hat{c}$, $\mathcal{S}$ sets

$$Y_{\hat{\jmath}} = g_1^{a'_{\hat{\jmath}} / r_{\hat{\jmath},\hat{c}}} \cdot g_2^{a''_{\hat{\jmath}} / v_{\hat{\jmath},\hat{c}}} \cdot W_{\hat{\jmath}}$$

otherwise, if $y_{\hat{\jmath}} = \hat{b}$, $\mathcal{S}$ sets

$$Y_{\hat{\jmath}} = (A_1^{\beta} \cdot C_4)^{a'_{\hat{\jmath}}} \cdot g_2^{a''_{\hat{\jmath}} / v_{\hat{\jmath},\hat{b}}} \cdot W_{\hat{\jmath}} = g_1^{a'_{\hat{\jmath}} \beta} \cdot g_2^{a''_{\hat{\jmath}} / v_{\hat{\jmath},\hat{b}}} \cdot (C_4^{a'_{\hat{\jmath}}} \cdot W_{\hat{\jmath}}).$$

Notice that this setting implicitly defines $r_{\hat{\jmath},\hat{b}} = 1/\beta$ which remains unknown to $\mathcal{S}$.

- k**-th query.**

  Let $\vec{y}^{(k)} = (y_1^{(k)}, \ldots, y_\ell^{(k)})$ be the k-th key query. We have three cases.

  Case B.1: $y_f^{(k)} = \star$. $\mathcal{S}$ performs the same steps of Case A.2.

  Case B.2: $y_f^{(k)} \neq \star$ **and** $y_{\hat{\jmath}}^{(k)} \neq \hat{b}$. In this case, $\mathcal{S}$ aborts.

  Case B.3: $y_f^{(k)} \neq \star$ **and** $y_{\hat{\jmath}}^{(k)} = \hat{b}$. Let $S = S_{\vec{y}} \setminus \{\hat{\jmath}, h\}$, where h is an index such that $y_h^{(k)} \neq \star$. Such an index h always exists since we assumed that each query contains at least two non-$\star$ entries. Then, for each $i \in S$, $\mathcal{S}$ chooses random $W_i \in \mathbb{G}_{p_4}$ and random $a'_i, a''_i \in \mathbb{Z}_N$ and sets

$$Y_i = g_1^{a'_i / r_{i,y_i^{(k)}}} \cdot g_2^{a''_i / v_{i,y_i^{(k)}}} \cdot W_i.$$

  $\mathcal{S}$ then chooses random $a''_{\hat{\jmath}} \in \mathbb{Z}_N$ and $W_{\hat{\jmath}}, W_h \in \mathbb{G}_{p_4}$ and sets

$$Y_{\hat{\jmath}} = T \cdot g_2^{a''_{\hat{\jmath}} / v_{\hat{\jmath},\hat{b}}} \cdot W_{\hat{\jmath}} \quad \text{and} \quad Y_h = (A_1^{\alpha} B_4)^{-1/r_{h,y_h^{(k)}}} \cdot g_1^{-s'/r_{h,y_h^{(k)}}} \cdot g_2^{-(s''+a''_{\hat{\jmath}})/v_{h,y_h^{(k)}}} \cdot W_h,$$

  where $s' = \sum_{i \in S} a'_i$ and $s'' = \sum_{i \in S} a''_i$.

  This terminates the description of how $\mathcal{S}$ handles the k-th key query.

- **Remaining $q-k$ queries.**

  $\mathcal{S}$ handles the remaining $q-k$ queries as in Case A.2, independently from whether $y_f = \star$ or $y_f \neq \star$.

  More precisely, if $y_{\hat{\jmath}} = \hat{c}$ then $\mathcal{S}$ runs KeyGen on input $\vec{y}$ and Msk and all the needed r_{i,y_i}'s and v_{i,y_i}'s are found in Msk. On the other hand, if $y_{\hat{\jmath}} = \hat{b}$, $\mathcal{S}$ can use $A_1^{\beta} \cdot C_4$ from D.

This terminates the description of how $\mathcal{S}$ answers key-queries.

Answering Challenge Query for $(\vec{x}_0, \vec{x}_1)$. $\mathcal{S}$ picks random $\eta \in \{0,1\}$ and sets $\vec{x} = \vec{x}_\eta$.

Then $\mathcal{S}$ tries to construct the challenge ciphertext by running algorithm $\mathsf{Encrypt}$ on input the challenge vector $\vec{x}$, public parameters Pk' and by randomizing the $\mathbb{G}_{p_1}$ part of all components X_i for $i < f$ such that $x_{0,i} \neq x_{1,i}$. However, Pk' is incomplete since it is missing $T'_{\hat{\jmath},\hat{b}}$ and thus $\mathcal{S}$ might have to abort.

More precisely, If $x_{\hat{\jmath}} = \hat{b}$, $\mathcal{S}$ aborts. Else (that is, if $x_{\hat{\jmath}} = \hat{c}$) $\mathcal{S}$ chooses random $s \in \mathbb{Z}_N$. For each $i \in [f-1]$ such that $x_{0,i} \neq x_{1,i}$, $\mathcal{S}$ sets r_i equal to a random element in $\mathbb{Z}_N$ and $r_i = 1$ for all remaining i's. Then, for each $i \in [\ell]$, $\mathcal{S}$ picks random $Z_i \in \mathbb{G}_{p_3}$ and sets $X_i = T'^{sr_i}_{i,x_i} \cdot Z_i$, and returns the tuple $(X_i)_{i\in[\ell]}$.

This ends the description of $\mathcal{S}$.

The simulator $\mathcal{S}$ described will be used to prove properties of games GBadQ. We can modify the simulator $\mathcal{S}$ so that, on input f and k, the challenge ciphertext is constructed by randomizing the $\mathbb{G}_{p_1}$ part also of the f-th component. The so modified simulator, that we call $\mathcal{S}_2$, closely simulates the work of games $\mathsf{GBadQ2}$ and will be used to prove properties of these games.

5.2.3 Properties of simulator $\mathcal{S}$

A sanity check. We verify that $\mathcal{S}$ cannot test the nature of T and thus break Assumption 2. Indeed to do so, $\mathcal{S}$ should use T to generate a key for $\vec{y}$ and ciphertext for $\vec{x}$ such that $\mathsf{Match}(\vec{x}, \vec{y}) = 1$. Then, if $T = T_1$ the Test procedure will have success; otherwise, it will fail. In constructing the key, $\mathcal{S}$ would use T to construct the $\hat{\jmath}$-th component (which forces $y_{\hat{\jmath}} = \hat{b}$) and then it would need $r_{\hat{\jmath},\hat{b}} = 1/\beta$ to construct the matching ciphertext. However, $\mathcal{S}$ does not have access to this value as part of the challenge. If we modify Assumption 2 to include such a value as part of the challenge, the resulting assumption is easily seen not to hold.

Why we need aborts. The aim of $\mathcal{S}$ is to use the value T from the challenge to simulate either $\mathsf{GBadQ}(f, k-1)$ or $\mathsf{GBadQ}(f,k)$ and it does so by embedding T in the $\hat{\jmath}$-th component of the reply to k-th key query. Suppose $y_f \neq \star$ and thus the two games are supposed to differ in the reply to the k-th key query. If $y_{\hat{\jmath}}^{(k)} = \star$ then the $\hat{\jmath}$-th component is empty. Moreover, if $y_{\hat{\jmath}}^{(k)} = \hat{c}$ then the $\hat{\jmath}$-th component can be computed using $1/r_{\hat{\jmath},\hat{c}}$ which is known to $\mathcal{S}$ but independent from T. Thus in both case $\mathcal{S}$'s plan fails and consequently $\mathcal{S}$ aborts (see Case B.2). Moreover, if $x_{\hat{\jmath}} = \hat{b}$ then $\mathcal{S}$ should use $T'_{\hat{\jmath},\hat{b}}$ which is missing from Pk'. Thus in this case $\mathcal{S}$ aborts too.

Notation. We use $\mathsf{NotAbort}^{\mathcal{A}}_{1,\mathcal{S}}(f,k)$ to denote the event that $\mathcal{S}$ does not abort while computing the answer to the k-th query in an interaction with $\mathcal{A}$ on input f and k. This is equivalent to the event that the k-th key query $y^{(k)}$ of adversary $\mathcal{A}$ is such that $y_f^{(k)} = \star$ or $y_{\hat{\jmath}}^{(k)} = \hat{b}$. In addition, we use $\mathsf{NotAbort}^{\mathcal{A}}_{2,\mathcal{S}}(f,k)$ to denote the event that $\mathcal{S}$ does not abort while computing the challenge ciphertext in an interaction with $\mathcal{A}$ on input f and k. This is equivalent to the event that adversary $\mathcal{A}$ outputs challenge vectors $\vec{x}_0$ and $\vec{x}_1$ such that $x_{\eta,\hat{\jmath}} = \hat{c}$.

For a game G between the challenger $\mathcal{C}$ and the adversary, we modify $\mathcal{C}$ so that $\mathcal{C}$ picks $\hat{\jmath}$ and $\hat{b}$ just like $\mathcal{S}$ does. This modification makes the definitions of events $\mathsf{NotAbort}^{\mathcal{A}}_{1,G}$ and $\mathsf{NotAbort}^{\mathcal{A}}_{2,G}$ meaningful. Notice however that, unlike the simulator $\mathcal{S}$, $\mathcal{C}$ never aborts its interaction with $\mathcal{A}$ and that this modification does not affect $\mathcal{A}$'s view. We write $\mathsf{NotAbort}^{\mathcal{A}}_{2}$ as a shorthand for $\mathsf{NotAbort}^{\mathcal{A}}_{2,\mathsf{GPK}}$.

The proof of the following lemma is found in Appendix F.2.

Lemma 8. *Suppose event* $\mathsf{NotAbort}^{\mathcal{A}}_{1,\mathcal{S}}(f,k)$ *occurs. If* $T = T_1$ *then* $\mathcal{A}$*'s view up to the Challenge Query in the interaction with* $\mathcal{S}$ *running on input* (f,k) *is the same as in* $\mathsf{GBadQ}(f,k-1)$*. If instead* $T = T_2$ *then* $\mathcal{A}$*'s view up to the Challenge Query in the interaction with* $\mathcal{S}$ *running on input* (f,k) *is the same as in* $\mathsf{GBadQ}(f,k)$*.*

Moreover, suppose events $\mathsf{NotAbort}^{\mathcal{A}}_{1,\mathcal{S}}(f,k)$ *and* $\mathsf{NotAbort}^{\mathcal{A}}_{2,\mathcal{S}}(f,k)$ *occur. If* $T = T_1$ *then* $\mathcal{A}$*'s total view in the interaction with* $\mathcal{S}$ *running on input* (f,k) *is the same as in* $\mathsf{GBadQ}(f,k-1)$*. If instead* $T = T_2$ *then* $\mathcal{A}$*'s total view in the interaction with* $\mathcal{S}$ *running on input* (f,k) *is the same as in* $\mathsf{GBadQ}(f,k)$*.*

Next we define event $E_f^{\mathcal{A}}$ as the event that in game GPK, the adversary $\mathcal{A}$ declares two challenge vectors that differ in the f-th component. When the adversary $\mathcal{A}$ is clear from the context we will simply write E_f.

Lemma 9. *Suppose there exists an adversary* $\mathcal{A}$ *and integers* $1 \le f \le \ell+1$ *and* $1 \le k \le q$ *such that* $\left|\mathsf{Adv}^{\mathcal{A}}[G] - \mathsf{Adv}^{\mathcal{A}}[H]\right| \ge \epsilon$*, where* $G = \mathsf{GBadQ}(f,k-1)$*,* $H = \mathsf{GBadQ}(f,k)$ *and* $\epsilon > 0$*. Then, there exists a PPT algorithm* $\mathcal{B}$ *with* $\mathsf{Adv}^{\mathcal{B}}_2 \ge \mathrm{Prob}[E_f]\cdot\epsilon/(2\cdot\ell^2) - \nu(\lambda)$*, for a negligible function* ν*.*

The proof is found in Appendix F.3. The following Lemma can be proved by referring to simulator $\mathcal{S}_2$. We omit further details since the proof is essentially the same as the one of Lemma 9.

Lemma 10. *Suppose there exists an adversary* $\mathcal{A}$ *and integers* $1 \le f \le \ell+1$ *and* $1 \le k \le q$ *such that* $\left|\mathsf{Adv}^{\mathcal{A}}[G] - \mathsf{Adv}^{\mathcal{A}}[H]\right| \ge \epsilon$*, where* $G = \mathsf{GBadQ2}(f,k-1)$*,* $H = \mathsf{GBadQ2}(f,k)$ *and* $\epsilon > 0$*. Then, there exists a PPT algorithm* $\mathcal{B}$ *with* $\mathsf{Adv}^{\mathcal{B}}_2 \ge \mathrm{Prob}[E_f]\cdot\epsilon/(2\cdot\ell^2) - \nu(\lambda)$*, for a negligible function* ν*.*

5.2.4 The advantage of $\mathcal{A}$ in GPK

In this section we prove that, under Assumption 2, every PPT adversary $\mathcal{A}$ has a negligible advantage in $\mathsf{GPK} = \mathsf{GBadCh}(1)$ by proving that it is computationally indistinguishable from $\mathsf{GBadCh}(\ell+1)$ that, by Observation 7, gives no advantage to any adversary. We state the lemma and postpone its proof to Appendix F.4.

Lemma 11. *If Assumption 2 holds, then, for any PPT adversary* $\mathcal{A}$*,* $\mathsf{Adv}^{\mathcal{A}}[\mathsf{GPK}]$ *is negligible. Specifically, if there is an adversary* $\mathcal{A}$ *with* $\mathsf{Adv}^{\mathcal{A}}[\mathsf{GPK}] = \epsilon$ *then there exists an adversary* $\mathcal{B}$ *against Assumption 2 such that* $\mathsf{Adv}^{\mathcal{B}}_2 \ge \frac{\epsilon^2}{2q\ell^4} - \nu(\lambda)$*, for some negligible function* ν*.*

5.3 Wrapping up

By combining Lemma 2 and Lemma 11 we obtain our main result.

Theorem 12. *If Assumption 1 and 2 hold, then the HVE scheme described in Section 4 is secure (in the sense of Definition 1).*

6 Conclusions and future directions

In this paper we have presented the *first* fully secure construction for a predicate encryption that is secure against unrestricted adversarial key queries by giving a construction for HVE. Our work shows that there is no need to restrict the querying power of the adversary to obtain a fully secure construction for a predicate encryption. We base our proof of security on natural hardness assumptions on groups of composite orders. Our proof technique stems from the dual system encryption methodology of Waters [26] which is augmented with a careful design of the intermediate security games based on observations on the relationship between the challenge ciphertexts and matching and non-matching queries.

We see two complementary directions for future work. The first one concentrates on the design of more efficient schemes and this can be achieved by reducing the size of keys and ciphertexts along the line of [22, 21, 15]. Along the second direction, one would like to design predicate encryption schemes for richer classes of predicates. The main and most challenging open problem in this area though remains to have a general result about predicates with poly-size circuits.

References

[1] Dan Boneh. Bilinear groups of composite order. In Tsuyoshi Takagi, Tatsuaki Okamoto, Eiji Okamoto, and Takeshi Okamoto, editors, *Pairing-Based Cryptography - Pairing 2007, First International Conference. Prooceedings*, volume 4575 of *LNCS*, pages 39–56, Tokyo, Japan, July 2–4, 2007. Springer-Verlag, Berlin, Germany.

[2] Dan Boneh and Xavier Boyen. Secure identity based encryption without random oracles. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 443–459, Santa Barbara, CA, USA, August 15–19, 2004. Springer-Verlag, Berlin, Germany.

[3] 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.

[4] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In *TCC 2005*, volume 3378 of *LNCS*, pages 325–341, Cambridge, MA, USA, February 10–12, 2005. Springer-Verlag, Berlin, Germany.

[5] Dan Boneh, Amit Sahai, and Brent Waters. Functional Encryption: Definitions and challenges. Technical Report 2010-543, Cryptology ePrint Archives, 2010. `http://eprint.iacr.org/2010/543/`.

[6] 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.

[7] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 207–222, Interlaken, Switzerland, May 2–6, 2004. Springer-Verlag, Berlin, Germany.

[8] David Mandell Freeman. Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 44–61, French Riviera, France, May 10 –June 3, 2010. Springer-Verlag, Berlin, Germany.

[9] Craig Gentry. Pratical identity-based encryption without random oracles. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 445–464, St. Petersburg, Russia, May 28 –June 1, 2006. Springer-Verlag, Berlin, Germany.

[10] Craig Gentry and Shai Halevi. Hierarchical identity based encryption with polynomially many levels. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 437–456, San Francisco, CA, USA, 2009. Springer-Verlag, Berlin, Germany.

[11] 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.

[12] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432, Istanbul, Turkey, April 13–17, 2008. Springer-Verlag, Berlin, Germany.

[13] 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.

[14] 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.

[15] Kwangsu Lee and Dong Hoon Lee. Improved hidden vector encryption with short ciphertexts and tokens. *Des. Codes Cryptography*, 58(3):297–319, 2011.

[16] Allison Lewko. Tools for simulating features of composite order bilinear groups in the prime order setting. Technical Report 2011-490, Cryptology ePrint Archives, 2011. `http://eprint.iacr.org/2011/490/`.

[17] Allison Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In Henri Gilbert, editor, *EUROCRYPT 2010*, pages 62–91, French Riviera, France, May 10 –June 3, 2010. Springer-Verlag, Berlin, Germany.

[18] Allison B. Lewko and Brent Waters. New techniques for dual system encryption and fully secure hibe with short ciphertexts. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 455–479, Zurich, Switzerland, February 9–11, 2010. Springer-Verlag, Berlin, Germany.

[19] Tatsuaki Okamoto and Katsuyuki Takashima. Fully Secure Functional Encryption with General Relations from the Decisional Linear Assumption. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 191–208, Santa Barbara, CA, USA, August 15–19, 2010. Springer-Verlag, Berlin, Germany.

[20] Adam O'Neill. Definitional issues in functional encryption. Technical Report 2010-556, Cryptology ePrint Archives, 2010. `http://eprint.iacr.org/2010/556/`.

[21] Jong Hwan Park and Dong Hoon Lee. A hidden vector encryption scheme with constant-size tokens and pairing computations. *IEICE Transactions*, 93-A(9):1620–1631, 2010.

[22] Saeed Sedghi, Peter van Liesdonk, Svetla Nikova, Pieter H. Hartel, and Willem Jonker. Searching keywords with wildcards on encrypted data. In Juan A. Garay and Roberto De Prisco, editors, *SCN 10*, volume 6280 of *LNCS*, pages 138–153, Amalfi, Italy, September 13–15, 2010. Springer-Verlag, Berlin, Germany.

[23] 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, May 20–23, 2007. IEEE Computer Society Press.

[24] Elaine Shi and Brent Waters. Delegating capabilities in predicate encryption systems. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *ICALP 2008*, volume 5126 of *LNCS*, pages 560–578, Reykjavik, Iceland, July 7–11, 2008. Springer-Verlag, Berlin, Germany.

[25] Brent Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 114–127, Aarhus, Denmark, May 22–26, 2005. Springer-Verlag, Berlin, Germany.

[26] Brent Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 619–636, Santa Barbara, CA, USA, August 16–20, 2009. Springer-Verlag, Berlin, Germany.

A An alternative description of our HVE

In this section we give an alternative albeit equivalent description of the HVE algorithms that will make our proof of security simpler.

We start from the simple observation that the exponent arithmetic is performed modulo the order of the base. For sake of concreteness, let us look at the KeyGen algorithm that sets Y_i equal to

$$Y_i = (g_1 \cdot g_2)^{a_i/t_{i,y_i}} \cdot W_i,$$

for $a_i, t_{i,y_i} \in \mathbb{Z}_N$. This is equivalent to computing Y_i as

$$Y_i = g_1^{a'_i/r_{i,y_i}} \cdot g_2^{a''_i/v_{i,y_i}} \cdot W_i$$

for $a'_i, a''_i, r_{i,y_i}, v_{i,y_i} \in \mathbb{Z}_N$ satisfying the following system of modular equations

$$\begin{aligned} a'_i &\equiv a_i \pmod{p_1} & \qquad r_{i,y_i} &\equiv t_{i,y_i} \pmod{p_1} \\ a''_i &\equiv a_i \pmod{p_2} & \qquad v_{i,y_i} &\equiv t_{i,y_i} \pmod{p_2} \end{aligned}$$

Conversely, computing $Y_i = g_1^{a'_i/r_{i,b}} \cdot g_2^{a''_i/v_{i,b}}$ for $a'_i, a''_i, r_{i,b}, v_{i,b} \in \mathbb{Z}_N$ is equivalent to computing $g_1^{a_i/t_{i,b}} \cdot g_2^{a_i/t_{i,b}}$ for $a_i, t_{i,b} \in \mathbb{Z}_N$ satisfying the above system of modular equations (in the unknowns a_i and $t_{i,b}$). By the Chinese remainder theorem the above systems have solutions in $\mathbb{Z}_N$ provided that N is a multiple of $p_1 \cdot p_2$. Moreover, for all values of $r_{i,b}$ and $v_{i,b}$, and of a'_i and a''_i the systems above have the same number of solutions. Therefore, the distributions of $Y_i = g_{12}^{a_i/t_{i,b}}$ for random $a_i, t_{i,b} \in \mathbb{Z}_N$ and the distribution of $Y_i = g_1^{a'_i/r_{i,b}} \cdot g_2^{a''_i/v_{i,b}}$ for random $a'_i, a''_i, r_{i,b}, v_{i,b} \in \mathbb{Z}_N$ coincide.

In view of the above observation, we can describe the Setup and KeyGen algorithms in the following way.

$\mathsf{Setup}(1^\lambda, 1^\ell)$**:** The setup algorithm chooses a description of a bilinear group $\mathcal{I} = (N = p_1p_2p_3p_4, \mathbb{G}, \mathbb{G}_T, \mathbf{e})$ with known factorization by running a generator algorithm $\mathcal{G}$ on input 1^λ. The setup algorithm chooses random $g_1 \in \mathbb{G}_{p_1}$, $g_2 \in \mathbb{G}_{p_2}$, $g_3 \in \mathbb{G}_{p_3}$, $g_4 \in \mathbb{G}_{p_4}$, and, for $i \in [\ell]$ and $b \in \{0,1\}$, random $r_{i,b}, v_{i,b} \in Z_N$ and random $R_{i,b} \in \mathbb{G}_{p_3}$ and sets $T_{i,b} = g_1^{r_{i,b}} \cdot R_{i,b}$.

The public parameters are $\mathsf{Pk} = [N, g_3, (T_{i,b})_{i\in[\ell],b\in\{0,1\}}]$ and the master secret key is $\mathsf{Msk} = [g_{12}, g_4, (r_{i,b}, v_{i,b})_{i\in[\ell],b\in\{0,1\}}]$, where $g_{12} = g_1 \cdot g_2$.

$\mathsf{KeyGen}(\mathsf{Msk}, \vec{y})$**:** Let $S_{\vec{y}}$ be the set of indices i such that $y_i \neq \star$. The key generation algorithm chooses random $a'_i \in \mathbb{Z}_N$ for $i \in S_{\vec{y}}$ and random $a''_i \in \mathbb{Z}_N$ for $i \in S_{\vec{y}}$ under the constraint that $\sum_{i\in S_{\vec{y}}} a'_i = \sum_{i\in S_{\vec{y}}} a''_i = 0$. For $i \in S_{\vec{y}}$, the algorithm chooses random $W_i \in \mathbb{G}_{p_4}$ and sets

$$Y_i = g_1^{a'_i/r_{i,y_i}} \cdot g_2^{a''_i/v_{i,y_i}} \cdot W_i.$$

The algorithm returns the tuple $(Y_i)_{i\in S_{\vec{y}}}$.

B Correctness of our HVE scheme

Lemma 13. *The HVE scheme presented in Section 4 is correct.*

PROOF. Suppose that $\mathsf{Ct} = (X_i)_{i\in[\ell]}$ is a ciphertext for attribute vector $\vec{x}$ and that $\mathsf{Sk}_{\vec{y}} = (Y_i)_{i\in S_{\vec{y}}}$ is a key for vector $\vec{y}$ such that $\mathsf{Match}(\vec{x},\vec{y}) = 1$. We show that $\mathsf{Test}(\mathsf{Ct},\mathsf{Sk}_{\vec{y}})$ returns TRUE. By definition we have that $\mathsf{Test}(\mathsf{Ct},\mathsf{Sk}_{\vec{y}})$ computes the following product

$$\begin{aligned}
\prod_{i\in S_{\vec{y}}} \mathbf{e}(X_i, Y_i) &= \prod_{i\in S_{\vec{y}}} \mathbf{e}(T_{i,x_i}^{s} \cdot Z_i, g_{12}^{a_i/t_{i,y_i}} \cdot W_i) && (1)\\
&= \prod_{i\in S_{\vec{y}}} \mathbf{e}(g_1^{t_{i,x_i}s}, g_1^{a_i/t_{i,y_i}}) && (2)\\
&= \prod_{i\in S_{\vec{y}}} \mathbf{e}(g_1^{t_{i,x_i}s}, g_1^{a_i/t_{i,x_i}}) && (3)\\
&= \prod_{i\in S_{\vec{y}}} \mathbf{e}(g_1, g_1)^{a_i s} && (4)\\
&= \mathbf{e}(g_1, g_1)^{s\sum_{i\in S_{\vec{y}}} a_i} && (5)\\
&= 1. && (6)
\end{aligned}$$

Equation 1 follows by definition of the X_i's and Y_i's. Equation 2 follows from Equation 1 by definition of the T_{i,x_i} and by the orthogonality property of bilinear groups of composite order discussed in Section 3. Equation 3 follows from Equation 2 since, if $\mathsf{Match}(\vec{x},\vec{y}) = 1$, then, for each $i \in S_{\vec{y}}$, $x_i = y_i$. Equation 4 follows from Equation 3 by the bilinear property of $\mathbf{e}$. Equation 5 follows from Equation 4 by simple algebraic manipulations. Equation 6 follows from Equation 5 since by construction we have $\sum_{i\in S_{\vec{y}}} a_i = 0$. Finally, notice that if the value obtained by this computation is 1 (the identity of the target group) then, by definition, the Test procedure returns TRUE as expected. Suppose now that $\mathsf{Match}(\vec{x},\vec{y}) = 0$. Let A be the set of positions i such that $x_i = y_i$ and $B = S_{\vec{y}} \setminus A$. By definition we have that $\mathsf{Test}(\mathsf{Ct},\mathsf{Sk}_{\vec{y}})$ computes the following product

$$\prod_{i\in S_{\vec{y}}} \mathbf{e}(X_i, Y_i) = \mathbf{e}(g_1, g_1)^{s\sum_{i\in A} a_i} \cdot \mathbf{e}(g_1, g_1)^{s\sum_{i\in B} a_i w_i} \quad (7)$$

Equation 7 follows by some of the facts already observed in the discussion of Equations 1-5, by definition of A and B, and by setting $w_i \stackrel{\text{def}}{=} t_{i,x_i}/t_{i,y_i}$. Since that with very high probability $s \neq 0 \mod \mathbb{Z}_N$ then, by the fact that $\mathbf{e}$ is non-degenerate, such a product equals 1 (the unity of the target group) if and only if

$$\sum_{i\in A} a_i + \sum_{i\in B} a_i w_i = 0. \quad (8)$$

By recalling that $\sum_{i\in S_{\vec{y}}} a_i = 0$ and by the fact that A and B are a partition of $S_{\vec{y}}$, from Equation 8 follows that

$$\sum_{i\in S_{\vec{y}}} a_i + \sum_{i\in B} a_i(w_i - 1) = \sum_{i\in B} a_i(w_i - 1). \quad (9)$$

If $i \in B$, then $x_i \neq y_i, y_i \neq \star$, and thus, without loss of generality, $w_i = t_{i,0}/t_{i,1}$ is distributed randomly in $\mathbb{Z}_N$ with very high probability since that $t_{i,0}$ and $t_{i,1}$ are chosen randomly and independently from $\mathbb{Z}_N$. Hence, since that the set B is of cardinality exponentially smaller than N,

with very high probability for each $i \in B$, $w_i - 1$ is distributed independently and randomly in $\mathbb{Z}_N$. From the fact that $\mathsf{Match}(\vec{x}, \vec{y}) = 0$ it follows that the set B is non-empty and so, from the fact that the $w_i - 1$'s are distributed randomly and independently in $\mathbb{Z}_N$, it follows that Equation 9 is different from $0 \mod N$ with very high probability. Thus, with very high probability, the product computed by the Test procedure is different from the identity of the target group and so with the same probability it returns FALSE as expected. □

C Is the binary alphabet a limitation?

We have presented our construction for the binary alphabet. This was for the sake of presentation and does not represent a shortcoming of our construction. Indeed one can easily observe that an HVE scheme for a general alphabet Σ can be obtained with an expansion of $\log_2 |\Sigma|$: the encryption simply encrypts bit by bit by using the binary HVE, and the key generation procedure proceeds analogously. We stress that this reduction is *black-box* and does not depend on our specific scheme. We recall that, though the known fully secure (against restricted adversaries) predicate encryption constructions of [17, 19] directly support a large alphabet, this is at cost of a *quadratic time* complexity of the encryption and key generation procedures. Finally, we stress that in some applications the choice of the binary alphabet can be a *feature* and not a limitation. For example, image that our attributes are integers, and we wish the capability to test if an encrypted integer is even or odd. In this case we *need* bit-grained control of the encrypted data *by design*.

D Prime-order bilinear groups

Bilinear groups of composite-order introduced by [4] are a useful tool for constructing advanced cryptosystems and secure protocols but they give constructions that are less efficient than the ones based on prime order groups.

Even though composite-order group enjoy two properties, *orthogonality* (also called *cancellation*) and *projection*, several constructions only use the orthogonality property by which pairing cancels out elements from distinct subgroups. Based on this, Freeman [8] presented an elegant abstract framework to instantiate composite-order cryptosystems using prime-order groups. Though Freeman's transformation is not a compiler, it hints at general ideas for obtaining instantiations in prime order groups for protocols that use only the orthogonality property. On the same line, also previous work of Groth and Sahai [12] shows that composite-order and prime-order groups instantiations of their proof systems can be interchanged. Unfortunately, Freeman's strategy does not seem to apply to our construction as at some point in our proof we use the fact that $\mathbb{G}$ has two subgroups with relatively prime order (see for example, our alternative description in Appendix A). A similar problem occurs with the proof of Lewko and Waters [18]. Recently Lewko [16] explored a general methodology for converting composite order pairing-based cryptography into the prime order setting by employing the dual pairing vector space (DPVS) approach initiated by Okamoto and Takashima [19] which can be used to translate composite order schemes for which the prior techniques of Freeman were insufficient. Thus we could restate our results in the prime order groups using these tools, but we preferred to present our results in composite order groups because it results in shorter proofs of security and concise scheme.

E Generic security of our Complexity Assumptions

We now prove that, if factoring is hard, our two complexity assumptions hold in the generic group model. We adopt the framework of [14] to reason about assumptions in bilinear groups $\mathbb{G}, \mathbb{G}_T$ of composite order $N = p_1p_2p_3p_4$. We fix generators $g_{p_1}, g_{p_2}, g_{p_3}, g_{p_4}$ of the subgroups $\mathbb{G}_{p_1}, \mathbb{G}_{p_2}, \mathbb{G}_{p_3}, \mathbb{G}_{p_4}$ and thus each element of $x \in \mathbb{G}$ can be expressed as $x = g_{p_1}^{a_1} g_{p_2}^{a_2} g_{p_3}^{a_3} g_{p_4}^{a_4}$, for $a_i \in \mathbb{Z}_{p_i}$. For sake of ease of notation, we denote element $x \in \mathbb{G}$ by the tuple (a_1, a_2, a_3, a_4). We do the same with elements in $\mathbb{G}_T$ (with the respect to generator $\mathbf{e}(g_{p_i}, g_{p_i})$) and will denote elements in that group as bracketed tuples $[a_1, a_2, a_3, a_4]$. We use capital letters to denote random variables and reuse random variables to denote relationships between elements. For example, $X = (A_1, B_1, C_1, D_1)$ is a random element of $\mathbb{G}$, and $Y = (A_2, B_1, C_2, D_2)$ is another random element that shares the same $\mathbb{G}_{p_2}$ part.

We say that a random variable X is *dependent* from the random variables $\{A_i\}$ if there exists $\lambda_i \in \mathbb{Z}_N$ such that $X = \sum_i \lambda_i A_i$ as formal random variables. Otherwise, we say that X is *independent* of $\{A_i\}$. We state the following theorems from [14].

Theorem 14 (Theorem A.1 of [14])**.** *Let $N = \prod_{i=1}^m p_i$ be a product of distinct primes, each greater than 2^λ. Let $\{X_i\}$ be random variables over $\mathbb{G}$ and $\{Y_i\}, T_1$ and T_2 be random variables over $\mathbb{G}_T$. Denote by t the maximum degree of a random variable and consider the following experiment in the generic group model:*

Algorithm $\mathcal{A}$ is given $N, \{X_i\}, \{Y_i\}$ and T_b for random $b \in \{0,1\}$ and outputs $b' \in \{0,1\}$. $\mathcal{A}$'s advantage is the absolute value of the difference between the probability that $b = b'$ and $1/2$.

Suppose that T_1 and T_2 are independent of $\{Y_i\} \cup \{\mathbf{e}(X_i, X_j)\}$. Then if $\mathcal{A}$ performs at most q group operations and has advantage δ, then there exists an algorithm that outputs a nontrivial factor of N in time polynomial in λ and the running time of $\mathcal{A}$ with probability at least $\delta - \mathcal{O}(q^2t/2^\lambda)$.

Theorem 15 (Theorem A.2 of [14])**.** *Let $N = \prod_{i=1}^m p_i$ be a product of distinct primes, each greater than 2^λ. Let $\{X_i\}, T_1, T_2$ be random variables over $\mathbb{G}$ and let $\{Y_i\}$ be random variables over G_T, where all random variables have degree at most t.*

Let $N = \prod_{i=1}^m p_i$ be a product of distinct primes, each greater than 2^λ. Let $\{X_i\}, T_1$ and T_2 be random variables over $\mathbb{G}$ and let $\{Y_i\}$ be random variables over $\mathbb{G}_T$. Denote by t the maximum degree of a random variable and consider the same experiment as the previous theorem in the generic group model.

Let $S := \{i \mid \mathbf{e}(T_1, X_i) \neq \mathbf{e}(T_2, X_i)\}$ (where inequality refers to inequality as formal polynomials). Suppose each of T_1 and T_2 is independent of $\{X_i\}$ and furthermore that for all $k \in S$ it holds that $\mathbf{e}(T_1, X_k)$ is independent of $\{B_i\} \cup \{\mathbf{e}(X_i, X_j)\} \cup \{\mathbf{e}(T_1, X_i)\}_{i \neq k}$ and $\mathbf{e}(T_2, X_k)$ is independent of $\{B_i\} \cup \{\mathbf{e}(X_i, X_j)\} \cup \{\mathbf{e}(T_2, X_i)\}_{i \neq k}$. Then if there exists an algorithm $\mathcal{A}$ issuing at most q instructions and having advantage δ, then there exists an algorithm that outputs a nontrivial factor of N in time polynomial in λ and the running time of $\mathcal{A}$ with probability at least $\delta - \mathcal{O}(q^2t/2^\lambda)$.

We apply Theorem 15 to prove the security of our assumptions in the generic group model.

Assumption 1. We can express this assumption as:

$$X_1 = (0,0,1,0),\ X_2 = (A_1,0,A_3,0),\ X_3 = (B_1,0,B_3,0),\ X_4 = (0,0,0,1)$$

and

$$T_1 = (Z_1,0,Z_3,0), T_2 = (0,Z_2,Z_3,0).$$

It is easy to see that T_1 and T_2 are both independent of $\{X_i\}$ because, for example, Z_3 does not appear in the X_i's. Next, we note that for this assumption we have $S = \{2, 3\}$, and thus, considering T_1 first, we obtain the following tuples:

$$C_{1,2} = \mathbf{e}(T_1, X_2) = [Z_1 A_1, 0, Z_3 A_3, 0], \quad C_{1,3} = \mathbf{e}(T_1, X_3) = [Z_1 B_1, 0, Z_3 B_3, 0].$$

It is easy to see that $C_{1,k}$ with $k \in \{2, 3\}$ is independent of $\{\mathbf{e}(X_i, X_j)\} \cup \{\mathbf{e}(T_1, X_i)\}_{i \neq k}$. An analogous arguments apply for the case of T_2. Thus the independence requirements of Theorem 15 are satisfied and Assumption 1 is generically secure, assuming it is hard to find a nontrivial factor of N.

Assumption 2. We can express this assumption as:

$$\begin{array}{lll} X_1 = (1, 0, 0, 0), & X_2 = (0, 1, 0, 0), & X_3 = (0, 0, 1, 0), \\ X_4 = (0, 0, 0, 1), & X_5 = (A, 0, 0, B_4), & X_6 = (B, 0, 0, C_4) \end{array}$$

and

$$T_1 = [AB, 0, 0, D_4], \quad T_2 = [Z_1, 0, 0, Z_4].$$

We note that D_4 and Z_1 do not appear in $\{X_i\}$ and thus T_1 and T_2 are both independent from them. Next, we note that for this assumption we have $S = \{1, 4, 5, 6\}$, and thus, considering T_1 first, we obtain the following tuples:

$$\begin{array}{ll} C_{1,1} = \mathbf{e}(T_1, X_1) = [AB, 0, 0, 0], & C_{1,4} = \mathbf{e}(T_1, X_4) = [0, 0, 0, D_4] \\ C_{1,5} = \mathbf{e}(T_1, X_5) = [A^2 B, 0, 0, D_4 B_4], & C_{1,6} = \mathbf{e}(T_1, X_6) = [AB^2, 0, 0, D_4 C_4] \end{array}$$

It is easy to see that $C_{1,k}$ with $k \in \{4, 5, 6\}$ is independent of $\{\mathbf{e}(X_i, X_j)\} \cup \{\mathbf{e}(T_1, X_i)\}_{i \neq k}$. For $C_{1,1}$, we observe that the only way to obtain an element whose first component contains AB is by computing $\mathbf{e}(X_5, X_6) = [AB, 0, 0, B_4 C_4]$ but then there is no way to generate an element whose fourth component is $B_4 C_4$ and hence no way to cancel that term.

Analogous arguments apply for the case of T_2.

Thus the independence requirement of Theorem 15 is satisfied and Assumption 2 is generically secure, assuming it is hard to find a nontrivial factor of N.

F Proofs

F.1 Proof of indistinguishability of GReal and GPK

Remark 16. *Let* $\mathsf{Pk} = [N, g_3, (T_{i,b})_{i \in [\ell], b \in \{0,1\}}]$ *and* $\mathsf{Msk} = [g_1 \cdot g_2, g_4, (t_{i,b})_{i \in [\ell], b \in \{0,1\}}]$ *be a pair of public parameter and master secret key output by the* Setup *algorithm and consider* $\mathsf{Pk}' = [N, g_3, (T'_{i,b})_{i \in [\ell], b \in \{0,1\}}]$ *and* $\mathsf{Msk}' = [\hat{g}_1 \cdot g_2, g_4, (t_{i,b})_{i \in [\ell], b \in \{0,1\}}]$ *with* $T'_{i,b} = \hat{g}_1^{t_{i,b}} \cdot R'_{i,b}$ *for some* $\hat{g}_1 \in \mathbb{G}_{p_1}$ *and* $R'_{i,b} \in \mathbb{G}_{p_3}$. *We make the following easy observations.*

1. *For every* $\vec{y} \in \{0, 1, \star\}^\ell$, *the distributions* $\mathsf{KeyGen}(\mathsf{Msk}, \vec{y})$ *and* $\mathsf{KeyGen}(\mathsf{Msk}', \vec{y})$ *are identical.*
2. *Similarly, for every* $\vec{x} \in \{0, 1\}^\ell$, *the distributions* $\mathsf{Encrypt}(\mathsf{Pk}, \vec{x})$ *and* $\mathsf{Encrypt}(\mathsf{Pk}', \vec{x})$ *are identical.*

Lemma 2. *If Assumption* 1 *holds, then for any PPT adversary* $\mathcal{A}$, $\left|\mathsf{Adv}^{\mathcal{A}}[\mathsf{GReal}(\lambda, \ell)] - \mathsf{Adv}^{\mathcal{A}}[\mathsf{GPK}(\lambda, \ell)]\right|$ *is negligible.*

PROOF. We show a PPT algorithm $\mathcal{B}$ which receives $(\mathcal{I}, A_3, A_4, A_{13}, A_{12})$ and T and, depending on the nature of T, simulates $\mathsf{GReal}(\lambda, \ell)$ or $\mathsf{GPK}(\lambda, \ell)$ with $\mathcal{A}$. This suffices to prove the Lemma.

Setup. $\mathcal{B}$ starts by constructing public parameters Pk and Pk' in the following way. $\mathcal{B}$ sets $g_{12} = A_{12}, g_3 = A_3, g_4 = A_4$ and, for each $i \in [\ell]$ and $b \in \{0,1\}$, $\mathcal{B}$ chooses random $t_{i,b} \in \mathbb{Z}_N$ and sets $T_{i,b} = T^{t_{i,b}}$ and $T'_{i,b} = A_{13}^{t_{i,b}}$. Then $\mathcal{B}$ sets $\mathsf{Pk} = [N, g_3, (T_{i,b})_{i\in[\ell],b\in\{0,1\}}]$, $\mathsf{Msk} = [g_{12}, g_4, (t_{i,b})_{i\in[\ell],b\in\{0,1\}}]$, and $\mathsf{Pk}' = [N, g_3, (T'_{i,b})_{i\in[\ell],b\in\{0,1\}}]$ and starts the interaction with $\mathcal{A}$ on input Pk.

Answering Key Query for $(\vec{y})$. $\mathcal{B}$ runs algorithm KeyGen on input Msk and $\vec{y}$.

Answering Challenge Query for $(\vec{x}_0, \vec{x}_1)$. The challenge is created by $\mathcal{B}$ by picking random $\eta \in \{0,1\}$ and running the $\mathsf{Encrypt}$ algorithm on input $\vec{x}_\eta$ and Pk'.

This concludes the description of algorithm $\mathcal{B}$.

Now suppose $T \in \mathbb{G}_{p_1p_3}$, and thus it can be written as $T = h_1 \cdot h_3$ for $h_1 \in \mathbb{G}_{p_1}$ and $h_3 \in \mathbb{G}_{p_3}$. This implies that Pk received in input by $\mathcal{A}$ in the interaction with $\mathcal{B}$ has the same distribution as in GReal. Moreover, by writing A_{13} as $A_{13} = \hat{h}_1 \cdot \hat{h}_3$ for $\hat{h}_1 \in \mathbb{G}_{p_1}$ and $\hat{h}_3 \in \mathbb{G}_{p_3}$ which is possible since by assumption $A_{13} \in \mathbb{G}_{p_1p_3}$, we notice that that Pk and Pk' are as in the hypothesis of Remark 16 (with $g_1 = h_1$ and $\hat{g}_1 = \hat{h}_1$). Therefore the answers to key queries and the challenge ciphertext given by $\mathcal{B}$ to $\mathcal{A}$ have the same distribution as the answers and the challenge ciphertext received by $\mathcal{A}$ in $\mathsf{GReal}(\lambda, \ell)$. We can thus conclude that, when $T \in \mathbb{G}_{p_1p_3}$, $\mathcal{C}$ has simulated $\mathsf{GReal}(\lambda, \ell)$ with $\mathcal{A}$.

Let us discuss now the case $T \in \mathbb{G}_{p_2p_3}$. In this case, Pk provided by $\mathcal{B}$ has the same distribution as the public parameters produced by $\mathcal{C}$ in $\mathsf{GPK}(\lambda, \ell)$. Therefore, $\mathcal{C}$ is simulating $\mathsf{GPK}(\lambda, \ell)$ for $\mathcal{A}$.

This concludes the proof of the lemma. □

F.2 Proof of Lemma 8

Lemma 8. *Suppose event $\mathsf{NotAbort}^{\mathcal{A}}_{1,\mathcal{S}}(f,k)$ occurs. If $T = T_1$ then $\mathcal{A}$'s view up to the Challenge Query in the interaction with $\mathcal{S}$ running on input (f,k) is the same as in $\mathsf{GBadQ}(f, k-1)$. If instead $T = T_2$ then $\mathcal{A}$'s view up to the Challenge Query in the interaction with $\mathcal{S}$ running on input (f,k) is the same as in $\mathsf{GBadQ}(f,k)$.*

Moreover, suppose events $\mathsf{NotAbort}^{\mathcal{A}}_{1,\mathcal{S}}(f,k)$ and $\mathsf{NotAbort}^{\mathcal{A}}_{2,\mathcal{S}}(f,k)$ occur. If $T = T_1$ then $\mathcal{A}$'s total view in the interaction with $\mathcal{S}$ running on input (f,k) is the same as in $\mathsf{GBadQ}(f, k-1)$. If instead $T = T_2$ then $\mathcal{A}$'s total view in the interaction with $\mathcal{S}$ running on input (f,k) is the same as in $\mathsf{GBadQ}(f,k)$.

PROOF. First observe that Pk has the same distribution as the public parameters seen by $\mathcal{A}$ in both games. The same holds for the answers to the first $(k-1)$ Key Queries and to the last $(q-k)$ Key Queries. Let us now focus on the answer to the k-th Key Query. We have two cases:

Case 1: $y_f^{(k)} = \star$. Then the view of $\mathcal{A}$ in the interaction with $\mathcal{S}$ is independent from T (see Case B.1) and, on the other hand, by definition, the two games coincide. Therefore the lemma holds in this case.

Case 2: $y_f^{(k)} \neq \star$. Suppose $T = T_1 = A_1^{\alpha\beta} \cdot D_4$ and that $\mathsf{NotAbort}^{\mathcal{A}}_{1,\mathcal{S}}(f,k)$ occurs. Therefore, $y_{\hat{\jmath}}^{(k)} = \hat{b}$ and $\mathcal{S}$'s answer to the k-th key query has the same distributions as in $\mathsf{GBadQ}(f, k-1)$. Indeed, we have that

$$Y_{\hat{\jmath}} = g_1^{a'_{\hat{\jmath}}/r_{\hat{\jmath},\hat{b}}} \cdot g_2^{a''_{\hat{\jmath}}/v_{\hat{\jmath},\hat{b}}} \cdot D_4 \cdot W_{\hat{\jmath}}$$

with $a'_{\hat{\jmath}} = \alpha$ and $r_{\hat{\jmath},\hat{b}} = 1/\beta$ and

$$Y_h = g_1^{-(a'_{\hat{\jmath}}+s')/r_{h,y_h^{(k)}}} \cdot g_2^{-(a''_{\hat{\jmath}}+s'')/v_{h,y_h^{(k)}}} \cdot \left(B_4^{-1/r_{h,y_h^{(k)}}} \cdot W_h \right)$$

and thus the a'_is and a''_is are random and sum up to 0.

On the other hand if T is random in $\mathbb{G}_{p_1p_4}$ and $\mathsf{NotAbort}^{\mathcal{A}}_{1,\mathcal{S}}(f,k)$ occurs, the $\mathbb{G}_{p_1}$ parts of the Y_i's are random and thus the answer to the k-th query of $\mathcal{A}$ is distributed as in $\mathsf{GBadQ}(f,k)$.

For the second part of the lemma, we observe that the challenge ciphertext has the same distribution in both games and that, if $\mathsf{NotAbort}^{\mathcal{A}}_{2,\mathcal{S}}(f,k)$ occurs, $\mathcal{S}$ properly constructs the challenge ciphertext. □

F.3 Lower bounding the advantage in breaking Assumption 2

Before proving Lemma 9, we prove some useful lemmata.

Lemma 17. *For all f, k and $\mathcal{A}$,* $\mathrm{Prob}[\mathsf{NotAbort}^{\mathcal{A}}_{1,\mathcal{S}}(f,k)] \geq \frac{1}{\ell}$.

Proof. The probability of $\mathsf{NotAbort}^{\mathcal{A}}_{1,\mathcal{S}}(f,k)$ is at least the probability that $y^{(k)}_{\hat{\jmath}} = \hat{b}$. Moreover, the view of $\mathcal{A}$ up to the k-th key query is independent from $\hat{b}$ and $\hat{\jmath}$. Now observe that $\vec{y}^{(k)}$ has at least two non-star entry and, provided that $\hat{\jmath}$ is one of these (which happens with probability at least $2/\ell$), the probability that $y^{(k)}_{\hat{\jmath}} = \hat{b}$ is $1/2$. □

Lemma 18. *For all f, k and $\mathcal{A}$,* $\mathrm{Prob}[\mathsf{NotAbort}^{\mathcal{A}}_{2,G}(f,k)] \geq \frac{1}{2\ell}$ *where* $G = \mathsf{GBadQ}(f,k)$.

Proof. $\mathsf{NotAbort}^{\mathcal{A}}_{2,G}(f,k)$ is the event that $y^{(k)}_{\hat{\jmath}} \neq x_{\eta,\hat{\jmath}}$ in the game G played by the challenger $\mathcal{C}$ with $\mathcal{A}$. It is easy to see that the probability that $\mathcal{C}$ correctly guesses $\hat{\jmath}$ and $\hat{b}$ such that $x_{\eta,\hat{\jmath}} = \hat{c} = 1-\hat{b}$ is at least $1/(2\ell)$, independently from the view of $\mathcal{A}$. □

Next we define event $E^{\mathcal{A}}_{f,G}$ as the event that in game G the adversary $\mathcal{A}$ declares two challenge vectors that differ in the f-th component. When the adversary $\mathcal{A}$ is clear from the context we will simply write $E_{f,G}$. We extend the definition of $E_{f,G}$ to include the game played by $\mathcal{A}$ against the simulator $\mathcal{S}$. Thus we denote by $E^{\mathcal{A}}_{f,\mathcal{S}}(f',k)$ the event that in the interaction between $\mathcal{A}$ and $\mathcal{S}$ on input f' and k, $\mathcal{S}$ does not abort and $\mathcal{A}$ declares two challenge vectors that differ in the f-th component. If $\mathcal{A}, f'$ and k are clear from the context, we will simply write $E_{f,\mathcal{S}}$.

Lemma 19. *If Assumption 2 holds, then for $k = 1, \ldots, q$ and $f = 1, \ldots, \ell+1$, and for all PPT adversaries $\mathcal{A}$,* $\left|\mathrm{Prob}[E^{\mathcal{A}}_{f,G}] - \mathrm{Prob}[E^{\mathcal{A}}_{f,H}]\right|$ *and* $\left|\mathrm{Prob}[\mathsf{NotAbort}^{\mathcal{A}}_{2,G}] - \mathrm{Prob}[\mathsf{NotAbort}^{\mathcal{A}}_{2,H}]\right|$ *are negligible functions of λ, for games $G = \mathsf{GBadQ}(f,k-1)$ and $H = \mathsf{GBadQ}(f,k)$.*

Proof. We prove the lemma for $E_{f,G}$ and $E_{f,H}$. A similar reasoning holds for $\mathsf{NotAbort}^{\mathcal{A}}_{2,G}$ and $\mathsf{NotAbort}^{\mathcal{A}}_{2,H}$. For the sake of contradiction, suppose that $\mathrm{Prob}[E^{\mathcal{A}}_{f,G}] \geq \mathrm{Prob}[E^{\mathcal{A}}_{f,H}] + \epsilon$ for some non-negligible ϵ. Then we can modify simulator $\mathcal{S}$ into algorithm $\mathcal{B}$ with a non-negligible advantage in breaking Assumption 2. Algorithm $\mathcal{B}$ simply execute $\mathcal{S}$'s code. By Lemma 17 event $\mathsf{NotAbort}_{1,\mathcal{S}}$ occurs with probability at least $1/\ell$ and in this case $\mathcal{B}$ can continue the execution of $\mathcal{S}$'s code and receive the challenge vectors from $\mathcal{A}$. At this point, $\mathcal{B}$ checks whether they differ in the f-th component. If they do, $\mathcal{B}$ outputs 1; else $\mathcal{B}$ outputs 0. It is easy to see that, by Lemma 8, the above algorithm has a non-negligible advantage in breaking Assumption 2. □

The proof of the following corollary is straightforward from Lemma 19 and Observations 3-5.

Corollary 20. *For all $f = 1, \ldots, \ell+1$ and $k = 0, \ldots, q$, and all PPT adversaries $\mathcal{A}$, we have that, for $H = \mathsf{GBadQ}(f,k)$ $\left|\text{Prob}[E^{\mathcal{A}}_{f,H}] - \text{Prob}[E^{\mathcal{A}}_{f}]\right|$ and $\left|\text{Prob}[\mathsf{NotAbort}^{\mathcal{A}}_{2,H}] - \text{Prob}[\mathsf{NotAbort}^{\mathcal{A}}_{2}]\right|$ are negligible.*

We define event $\mathsf{Succ}^{\mathcal{A}}(f,k)$ as

$$\mathsf{Succ}^{\mathcal{A}}(f,k) := \mathsf{NotAbort}^{\mathcal{A}}_{1,\mathcal{S}}(f,k) \wedge \ \mathsf{NotAbort}^{\mathcal{A}}_{2,\mathcal{S}}(f,k) \wedge E^{\mathcal{A}}_{f,\mathcal{S}}(f,k). \tag{10}$$

We are now ready to prove Lemma 9.

Lemma 9. *Suppose there exists an adversary $\mathcal{A}$ and integers $1 \leq f \leq \ell+1$ and $1 \leq k \leq q$ such that $\left|\mathsf{Adv}^{\mathcal{A}}[G] - \mathsf{Adv}^{\mathcal{A}}[H]\right| \geq \epsilon$, where $G = \mathsf{GBadQ}(f,k-1)$, $H = \mathsf{GBadQ}(f,k)$ and $\epsilon > 0$. Then, there exists a PPT algorithm $\mathcal{B}$ with $\mathsf{Adv}^{\mathcal{B}}_2 \geq \text{Prob}[E_f] \cdot \epsilon/(2 \cdot \ell^2) - \nu(\lambda)$, for a negligible function ν.*

Proof. Assume without loss of generality that $\mathsf{Adv}^{\mathcal{A}}[G] \geq \mathsf{Adv}^{\mathcal{A}}[H] + \epsilon$ and consider the following algorithm $\mathcal{B}$. $\mathcal{B}$ uses simulator $\mathcal{S}$ as a subroutine and interacts with $\mathcal{A}$ on input integers f and k for which the above inequality holds, and an instance (D,T) of Assumption 2.

If event $\mathsf{Succ}^{\mathcal{A}}(f,k)$ does not occur, $\mathcal{B}$ outputs $\perp$. Otherwise, $\mathcal{B}$ receives $\mathcal{A}$'s output η' and checks if $\eta = \eta'$ (recall that η is the random bit chosen by $\mathcal{S}$ in preparing the challenge ciphertext). If $\eta = \eta'$ then $\mathcal{B}$ outputs 1; otherwise $\mathcal{B}$ outputs 0. Therefore we have

$$\begin{aligned} \text{Prob}[\mathcal{B} \text{ outputs } 1 | T = T_1] \ &= \ \text{Prob}[\mathcal{B} \text{ outputs } 1 | T = T_1 \wedge \mathsf{Succ}^{\mathcal{A}}(f,k)] \cdot \\ &\qquad \text{Prob}[\mathsf{Succ}_{\mathcal{A}}(f,k) | T = T_1] \end{aligned} \tag{11}$$

By definition of event $\mathsf{Succ}^{\mathcal{A}}(f,k)$ we have

$$\begin{aligned} \text{Prob}[\mathsf{Succ}_{\mathcal{A}}(f,k) | T = T_1] \ &= \ \text{Prob}[E_{f,\mathcal{S}} \wedge \mathsf{NotAbort}_{1,\mathcal{S}} \wedge \mathsf{NotAbort}_{2,\mathcal{S}} | T = T_1] \\ &= \ \text{Prob}[\mathsf{NotAbort}_{1,\mathcal{S}} | T = T_1] \cdot \text{Prob}[E_{f,\mathcal{S}} \wedge \mathsf{NotAbort}_{2,\mathcal{S}} | \mathsf{NotAbort}_{1,\mathcal{S}} \wedge T = T_1]. \end{aligned}$$

Now observe that event $\mathsf{NotAbort}_{1,\mathcal{S}}$ is determined before $\mathcal{S}$ uses T and thus

$$\text{Prob}[\mathsf{NotAbort}_{1,\mathcal{S}} | T = T_1] = \text{Prob}[\mathsf{NotAbort}_{1,\mathcal{S}}].$$

Moreover, by Lemma 8, if event $\mathsf{NotAbort}_{1,\mathcal{S}}$ occurs and $T = T_1$, the view of $\mathcal{A}$ up to Challenge Query is equal to the view of $\mathcal{A}$ in game G and thus

$$\text{Prob}[E_{f,\mathcal{S}} \wedge \mathsf{NotAbort}_{2,\mathcal{S}} | \mathsf{NotAbort}_{1,\mathcal{S}} \wedge T = T_1] = \text{Prob}[E_{f,G} \wedge \mathsf{NotAbort}_{2,G}]$$

whence

$$\begin{aligned} \text{Prob}[\mathsf{Succ}^{\mathcal{A}}(f,k) | T = T_1] \ &= \ \text{Prob}[\mathsf{NotAbort}_{1,\mathcal{S}}] \cdot \text{Prob}[\mathsf{NotAbort}_{2,G} \wedge E_{f,G}] \\ &= \ \text{Prob}[\mathsf{NotAbort}_{1,\mathcal{S}}] \cdot \text{Prob}[\mathsf{NotAbort}_{2,G}] \cdot \text{Prob}[E_{f,G}] \\ &\quad (\mathsf{NotAbort}_{2,G} \text{ and } E_{f,G} \text{ are independent}) \end{aligned}$$

Finally, if $T = T_1$ and $\mathsf{Succ}^{\mathcal{A}}(f,k)$ occures, then, by Lemma 8, $\mathcal{A}$'s view is exactly as in game G, and thus the probability that $\mathcal{B}$ outputs 1 is equal to the probability that $\mathcal{A}$ wins in game G. We can thus rewrite Eq. 11 as

$$\text{Prob}[\mathcal{B} \text{ outputs } 1 | T = T_1] = \text{Prob}[\mathcal{A} \text{ wins in } G] \cdot \text{Prob}[\mathsf{NotAbort}_{1,\mathcal{S}}] \cdot \text{Prob}[\mathsf{NotAbort}_{2,G}] \cdot \text{Prob}[E_{f,G}]$$

A similar reasoning yields

$$\text{Prob}[\mathcal{B} \text{ outputs } 1 | T = T_2] = \text{Prob}[\mathcal{A} \text{ wins in } H] \cdot \text{Prob}[\mathsf{NotAbort}_{1,\mathcal{S}}] \cdot \text{Prob}[\mathsf{NotAbort}_{2,H}] \cdot \text{Prob}[E_{f,H}]$$

By using Corollary 20, Lemma 17 and Lemma 18, we can conclude that there exists a negligible function ν such that we have

$$\begin{aligned} \mathsf{Adv}_2^{\mathcal{B}} &= \text{Prob}[\mathsf{NotAbort}_{1,\mathcal{S}}] \cdot \text{Prob}[\mathsf{NotAbort}_2] \cdot \text{Prob}[E_f] \cdot \Big(\text{Prob}[\mathcal{A} \text{ wins in } G] - \text{Prob}[\mathcal{A} \text{ wins in } H]\Big) - \nu(\lambda) \\ &\geq \frac{\epsilon}{2\ell^2} \cdot \text{Prob}[E_f] - \nu(\lambda). \end{aligned}$$

□

F.4 Upper bounding the advantage in GPK

Before proving Lemma 11, we state and prove some useful observations.

Let $E^{\mathcal{A}}_{f,f'}$ denote the event that during the execution of $\mathsf{GBadCh}(f')$ adversary $\mathcal{A}$ outputs two challenge vectors that differ in the f-th component. For an event E, we define the *advantage* $\mathsf{Adv}^{\mathcal{A}}[G|E]$ *of* $\mathcal{A}$ *in* G *conditioned on event* E as $\mathsf{Adv}^{\mathcal{A}}[G|E] = \text{Prob}[\mathcal{A} \text{ wins in game } G|E] - \frac{1}{2}$.

Observation 21. *For all PPT adversaries* $\mathcal{A}$ *and all* $1 \leq f \leq \ell$*, we have that*

$$\mathsf{Adv}^{\mathcal{A}}[\mathsf{GBadCh}(f)|\neg E_{f,f}] = \mathsf{Adv}^{\mathcal{A}}[\mathsf{GBadCh}(f+1)|\neg E_{f,f+1}].$$

Proof. By definition of GBadCh, if the two challenge vectors coincide in the f-th component, then the views of $\mathcal{A}$ in $\mathsf{GBadCh}(f)$ and $\mathsf{GBadCh}(f+1)$ are the same. □

Observation 22. *For all PPT adversaries* $\mathcal{A}$ *and all* $1 \leq f \leq \ell$*, we have that*

$$\text{Prob}[E^{\mathcal{A}}_{f,f}] = \text{Prob}[E^{\mathcal{A}}_{f,f+1}].$$

Proof. The view of $\mathcal{A}$ in $\mathsf{GBadCh}(f)$ up to the Challenge Query is independent from f. □

Therefore we can set $\text{Prob}[E^{\mathcal{A}}_{f,f}] = \text{Prob}[E^{\mathcal{A}}_{f,1}] = \text{Prob}[E^{\mathcal{A}}_f]$.

Lemma 11. *If Assumption 2 holds, then, for any PPT adversary* $\mathcal{A}$*,* $\mathsf{Adv}^{\mathcal{A}}[\mathsf{GPK}]$ *is negligible. Specifically, if there is an adversary* $\mathcal{A}$ *with* $\mathsf{Adv}^{\mathcal{A}}[\mathsf{GPK}] = \epsilon$ *then there exists an adversary* $\mathcal{B}$ *against Assumption 2 such that* $\mathsf{Adv}^{\mathcal{B}}_2 \geq \frac{\epsilon^2}{2q\ell^4} - \nu(\lambda)$*, for some negligible function* ν*.*

Proof. Let $\mathcal{A}$ be a PPT adversary such that $\mathsf{Adv}^{\mathcal{A}}[\mathsf{GPK}] \geq \epsilon$. Since $\mathsf{GPK} = \mathsf{GBadCh}(1)$ and $\mathsf{Adv}^{\mathcal{A}}[\mathsf{GBadCh}(\ell+1)] = 0$ there must exist $f \in [\ell]$ such that

$$\Big|\mathsf{Adv}^{\mathcal{A}}[\mathsf{GBadCh}(f)] - \mathsf{Adv}^{\mathcal{A}}[\mathsf{GBadCh}(f+1)]\Big| \geq \epsilon' = \epsilon/\ell. \tag{12}$$

Now recall that $\mathsf{GBadCh}(f) = \mathsf{GBadQ}(f, 0)$ and $\mathsf{GBadCh}(f+1) = \mathsf{GBadQ2}(f, 0)$. Thus, there exists k, $1 \leq k \leq q$ such that

$$\Big|\mathsf{Adv}^{\mathcal{A}}[G] - \mathsf{Adv}^{\mathcal{A}}[H]\Big| \geq \epsilon'/(2q)$$

where $G = \mathsf{GBadQ}(f, k)$ and $H = \mathsf{GBadQ}(f, k-1)$ or where $G = \mathsf{GBadQ2}(f, k)$ and $H = \mathsf{GBadQ2}(f, k-1)$. Then by Lemma 9, in the former case, and by Lemma 10 in the latter, we can construct an adversary $\mathcal{B}$ against Assumption 2, such that

$$\mathsf{Adv}^{\mathcal{B}}_2 \geq \frac{\epsilon}{4q\ell^3} \cdot \text{Prob}[E_f] - \nu(\lambda)$$

Now it remains to estimate $\text{Prob}[E_f]$. Notice that we can write

$$\begin{aligned}\mathsf{Adv}^{\mathcal{A}}[\mathsf{GBadCh}(f)] = \text{Prob}[E_{f,f}] \cdot \mathsf{Adv}^{\mathcal{A}}[\mathsf{GBadCh}(f)|E_{f,f}] + \\ \text{Prob}[\neg E_{f,f}] \cdot \mathsf{Adv}^{\mathcal{A}}[\mathsf{GBadCh}(f)|\neg E_{f,f}].\end{aligned}$$

and

$$\begin{aligned}\mathsf{Adv}^{\mathcal{A}}[\mathsf{GBadCh}(f+1)] = \text{Prob}[E_{f,f+1}] \cdot \mathsf{Adv}^{\mathcal{A}}[\mathsf{GBadCh}(f+1)|E_{f,f+1}] + \\ \text{Prob}[\neg E_{f,f+1}] \cdot \mathsf{Adv}^{\mathcal{A}}[\mathsf{GBadCh}(f+1)]|\neg E_{f,f+1}].\end{aligned}$$

and, by combining Equation 12 and Observations 21 and 22, we obtain

$$\text{Prob}[E_f] \cdot \left|\mathsf{Adv}^{\mathcal{A}}[\mathsf{GBadCh}(f)|E_{f,f}] - \mathsf{Adv}^{\mathcal{A}}[\mathsf{GBadCh}(f+1)|E_{f,f+1}]\right| \geq \epsilon'.$$

Since no advantage is greater than $1/2$, we can conclude that $\text{Prob}[E_f] \geq 2 \cdot \epsilon'$ and thus $\mathcal{B}$ as advantage

$$\mathsf{Adv}_2^{\mathcal{B}} \geq \frac{\epsilon^2}{2q\ell^4} - \nu(\lambda)$$

□

G Full-Fledged HVE

A *Full-Fledged* HVE scheme is a tuple of four efficient probabilistic algorithms (Setup, Encrypt, KeyGen, Decrypt) with the following semantics.

- $\mathsf{Setup}(1^\lambda, 1^\ell)$: takes as input a security parameter λ and a length parameter ℓ (given in unary), and outputs the public parameters Pk and the master secret key Msk.
- $\mathsf{KeyGen}(\mathsf{Msk}, \vec{y})$: takes as input the master secret key Msk and a vector $\vec{y} \in \{0,1,\star\}^\ell$, and outputs a secret key $\mathsf{Sk}_{\vec{y}}$.
- $\mathsf{Encrypt}(\mathsf{Pk}, \vec{x}, M)$: takes as input the public parameters Pk and a vector $\vec{x} \in \{0,1\}^\ell$, and a message M in some associated message space. It outputs a ciphertext Ct.
- $\mathsf{Decrypt}(\mathsf{Pk}, \mathsf{Ct}, \mathsf{Sk}_{\vec{y}})$: takes as input the public parameters Pk, a ciphertext Ct encrypting attribute vector $\vec{x}$ and message M, and a secret key $\mathsf{Sk}_{\vec{y}}$. It outputs plaintext M'.

Correctness. For correctness we require that for pairs $(\mathsf{Pk}, \mathsf{Msk})$ output by $\mathsf{Setup}(1^\lambda, 1^\ell)$, it holds that for any vectors $\vec{x} \in \{0,1\}^\ell$ and for any plaintext M and for any $\vec{y} \in \{0,1,\star\}^\ell$, we have that

$$\mathsf{Decrypt}(\mathsf{Pk}, \mathsf{Encrypt}(\mathsf{Pk}, \vec{x}, M), \mathsf{KeyGen}(\mathsf{Msk}, \vec{y})) = M$$

is negligibly in λ close to 1 if $\mathsf{Match}(\vec{x}, \vec{y}) = 1$ and negligibly in λ close to 0 if $\mathsf{Match}(\vec{x}, \vec{y}) = 0$.

G.1 Security definition

We define security by means of the following game GReal played between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

Setup. $\mathcal{C}$ runs the Setup algorithm on input the security parameter λ and the length parameter ℓ (given in unary) to generate public parameters Pk and master secret key Msk. $\mathcal{C}$ starts the interaction with $\mathcal{A}$ on input Pk.

Key Query Answering. Upon receiving a query for vector $\vec{y}$, $\mathcal{C}$ returns $\mathsf{KeyGen}(\mathsf{Msk}, \vec{y})$.

Challenge Construction. Upon receiving the pair $((\vec{x}_0, M_0), (\vec{x}_1, M_1))$, $\mathcal{C}$ picks random $\eta \in \{0, 1\}$ and returns $\mathsf{Encrypt}(\mathsf{Pk}, \vec{x}_\eta, M_\eta)$.

Winning Condition. Let η' be $\mathcal{A}$'s output. We say that $\mathcal{A}$ *wins* the game if $\eta = \eta'$ and for all $\vec{y}$ for which $\mathcal{A}$ has issued a Key Query, it holds $\mathsf{Match}(\vec{x}_0, \vec{y}) = \mathsf{Match}(\vec{x}_1, \vec{y})$. Moreover, if $\mathcal{A}$ has issued a Key Query for a vector $\vec{y}$ such that $\mathsf{Match}(\vec{x}_0, \vec{y}) = \mathsf{Match}(\vec{x}_1, \vec{y}) = 1$, then M_0 must be equal to M_1.

We define the advantage $\mathsf{Adv}^{\mathcal{A}}(\lambda)$ of $\mathcal{A}$ in GReal as the probability of winning minus $1/2$ and for security we require the advantage to be negligible.

G.2 Full-fledged scheme for HVE

It is easy to extend our scheme for HVE to the full-fledged case in the following way. In the schemes for Hidden Vector Encryption we add the value $\Omega = \mathbf{e}(g_1, g_1)^z$ for a random z to the public key and add z to the master secret key. In constructing the secret keys, we choose that the a_i's so that they sum up to z (instead of summing up to 0). In the encryption for a message $M \in \mathbb{G}_T$, we add the element $\Omega = M \cdot \Omega^s$, where s is the same random values used to compute the other components of the ciphertext. Then it is easy to see that the blinding factor Ω^s can be obtained from the keys and the ciphertext. For completeness, we present the formal description of the full-fledged scheme. The associated message space of the following full-fledged scheme is the target group $\mathbb{G}_T$. As for the predicate-only scheme of Section 4, we assume that all vectors associated with the key have at least two non-star positions.

$\mathsf{Setup}(1^\lambda, 1^\ell)$: The setup algorithm chooses a description of a bilinear group $\mathcal{I} = (N = p_1p_2p_3p_4, \mathbb{G}, \mathbb{G}_T, \mathbf{e})$ with known factorization by running a generator algorithm $\mathcal{G}$ on input 1^λ. The setup algorithm chooses random $z \in \mathbb{Z}_N$, $g_1 \in \mathbb{G}_{p_1}$, $g_2 \in \mathbb{G}_{p_2}$, $g_3 \in \mathbb{G}_{p_3}$, $g_4 \in \mathbb{G}_{p_4}$, and, for $i \in [\ell]$ and $b \in \{0, 1\}$, random $t_{i,b} \in Z_N$ and random $R_{i,b} \in \mathbb{G}_{p_3}$ and sets $T_{i,b} = g_1^{t_{i,b}} \cdot R_{i,b}$. Furthermore, it sets $\Omega = \mathbf{e}(g_1, g_1)^z$. The public parameters are $\mathsf{Pk} = [N, g_3, \Omega, (T_{i,b})_{i\in[\ell],b\in\{0,1\}}]$ and the master secret key is $\mathsf{Msk} = [g_{12}, g_4, (t_{i,b})_{i\in[\ell],b\in\{0,1\}}]$, where $g_{12} = g_1 \cdot g_2$.

$\mathsf{KeyGen}(\mathsf{Msk}, \vec{y})$: Let $S_{\vec{y}}$ be the set of indices i such that $y_i \neq \star$. The key generation algorithm chooses random $a_i \in \mathbb{Z}_N$ for $i \in S_{\vec{y}}$ under the constraint that $\sum_{i\in S_{\vec{y}}} a_i = z$. For $i \in S_{\vec{y}}$, the algorithm chooses random $W_i \in \mathbb{G}_{p_4}$ and sets

$$Y_i = g_{12}^{a_i/t_{i,y_i}} \cdot W_i.$$

The algorithm returns the tuple $(Y_i)_{i\in S_{\vec{y}}}$. Here we use the fact that $S_{\vec{y}}$ has size at least 2.

$\mathsf{Encrypt}(\mathsf{Pk}, \vec{x}, M)$: The encryption algorithm chooses random $s \in \mathbb{Z}_N$. For $i \in [\ell]$, the algorithm chooses random $Z_i \in \mathbb{G}_{p_3}$ and sets

$$X_i = T_{i,x_i}^s \cdot Z_i.$$

Furthermore, it computes $W = M \cdot \Omega^{-s}$. and returns the tuple $(W, (X_i)_{i\in[\ell]})$.

$\mathsf{Decrypt}(\mathsf{Ct}, \mathsf{Sk}_{\vec{y}})$: The decryption algorithm computes $T = \prod_{i\in S_{\vec{y}}} \mathbf{e}(X_i, Y_i)$. It returns $M' = W \cdot T$.

By adding a MAC (or some parity check code) to the plaintext M, it is possible to detect invalid decryptions. We omit the proof of correctness and security for this scheme since they are analogous to those for the predicate-only scheme of Section 4.

H Reductions

In this section we show how to construct an encryption scheme for the class of Boolean predicates that can be expressed as a k-CNF or k-DNF formula and disjunctions from an HVE scheme.

We first start by giving formal definitions for the Boolean Satisfaction Problem and its security properties.

H.1 Boolean Satisfaction Encryption

Let $\mathbb{B} = \{\mathbb{B}_n\}_{n>0}$ be a class of Boolean predicates indexed by the number n of variables. We define the $\mathsf{Satisfy}$ predicate as $\mathsf{Satisfy}(\Phi, \vec{z}) = \Phi(\vec{z})$ for $\vec{z} \in \{0,1\}^n$.

An *Encryption scheme for class* $\mathbb{B}$ is a tuple of four efficient probabilistic algorithms (Setup, $\mathsf{Encrypt}$, KeyGen, Test) with the following semantics.

$\mathsf{Setup}(1^\lambda, 1^n)$: takes as input a security parameter λ and the number n of variables, and outputs the public parameters Pk and the master secret key Msk.

$\mathsf{KeyGen}(\mathsf{Msk}, \Phi)$: takes as input the master secret key Msk and a formula $\Phi \in \mathbb{B}_n$ and outputs a secret key Sk_Φ.

$\mathsf{Encrypt}(\mathsf{Pk}, \vec{z})$: takes as input the public parameters Pk and a truth assignment $\vec{z}$ for n variables and outputs a ciphertext Ct.

$\mathsf{Test}(\mathsf{Pk}, \mathsf{Ct}, \mathsf{Sk}_\Phi)$: takes as input the public parameters Pk, a ciphertext Ct and a secret key Sk_Φ and outputs TRUE iff and only if the ciphertext is an encryption of a truth assignment $\vec{z}$ that satisfies Φ.

Correctness of Boolean Satisfaction Encryption. We require that for all pairs $(\mathsf{Pk}, \mathsf{Msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$, it holds that for any truth assignment $\vec{z}$ for n variables, for any formula $\Phi \in \mathcal{B}_n$ over n variables we have that the probability that $\mathsf{Test}(\mathsf{Pk}, \mathsf{Encrypt}(\mathsf{Pk}, \vec{z}), \mathsf{KeyGen}(\mathsf{Msk}, \Phi)) \neq \mathsf{Satisfy}(\Phi, \vec{z})$ is negligible in λ.

H.2 Security definitions for Boolean Satisfaction Encryption

For Boolean Satisfaction encryption, we have a game similar to that of HVE. GReal can be described in the following way.

Setup. $\mathcal{C}$ runs the Setup algorithm, $(\mathsf{Pk}, \mathsf{Msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$. Then $\mathcal{C}$ starts the interaction with $\mathcal{A}$ on input Pk.

Key Query Answering. For $\Phi \in \mathbb{B}_n$, $\mathcal{C}$ returns $\mathsf{KeyGen}(\mathsf{Msk}, \Phi)$.

Challenge Construction. Upon receiving the pair $(\vec{z}_0, \vec{z}_1)$ of truth assignments over n variables, $\mathcal{C}$ picks random $\eta \in \{0, 1\}$ and returns $\mathsf{Encrypt}(\mathsf{Pk}, \vec{z}_\eta)$.

Winning Condition. Let η' be $\mathcal{A}$'s output. We say that $\mathcal{A}$ *wins* the game if $\eta = \eta'$ and, for all Φ for which $\mathcal{A}$ has issued a Key Query, it holds that $\mathsf{Satisfy}(\Phi, z_0) = \mathsf{Satisfy}(\Phi, z_1)$.

We define the advantage $\mathsf{Adv}^{\mathcal{A}}_{\mathbb{B}}(\lambda)$ of $\mathcal{A}$ in GReal to be the probability of winning minus $1/2$.

Definition 23. *An Encryption scheme for class $\mathbb{B}$ is* secure *if for all PPT adversaries $\mathcal{A}$, we have that $\mathsf{Adv}^{\mathcal{A}}_{\mathbb{B}}(\lambda)$ is a negligible function of λ.*

H.3 Reducing k-CNF to HVE

We consider formulae Φ in k-CNF, for constant k, over n variables in which each clause $C \in \Phi$ contains exactly k distinct variables. We call such a clause *admissible* and denote by $\mathbb{C}_n$ the set of all admissible clauses over the n variables $x_1, \ldots, x_n$ and set $M_n = |\mathbb{C}|$. Notice that $M_n = \Theta(n^k)$. We also fix a canonical ordering $C_1, \ldots, C_{M_n}$ of the clauses in $\mathbb{C}_n$.

Let $\mathcal{H} = (\mathsf{Setup}_{\mathcal{H}}, \mathsf{KeyGen}_{\mathcal{H}}, \mathsf{Encrypt}_{\mathcal{H}}, \mathsf{Test}_{\mathcal{H}})$ be an HVE scheme and construct a k-CNF scheme $\mathsf{kCNF} = (\mathsf{Setup}_{\mathsf{kCNF}}, \mathsf{KeyGen}_{\mathsf{kCNF}}, \mathsf{Encrypt}_{\mathsf{kCNF}}, \mathsf{Test}_{\mathsf{kCNF}})$ as follows:

$\mathsf{Setup}_{\mathsf{kCNF}}(1^\lambda, 1^n)$: The algorithm returns the output of $\mathsf{Setup}_{\mathcal{H}}(1^\lambda, 1^{M_n})$.

$\mathsf{KeyGen}_{\mathsf{kCNF}}(\mathsf{Msk}, \Phi)$: For a k-CNF formula Φ, the key generation algorithm constructs vector $\vec{y} \in \{0, 1, \star\}^{M_n}$ by setting, for each $i \in \{1, \ldots, M_n\}$, $y_i = 1$ if $C_i \in \Phi$; $y_i = \star$ otherwise. We denote this transformation by $y = \mathsf{FEncode}(\Phi)$. Then the key generation algorithm returns $\mathsf{Sk}_\Phi = \mathsf{KeyGen}_{\mathcal{H}}(\mathsf{Msk}, \vec{y})$.

$\mathsf{Encrypt}_{\mathsf{kCNF}}(\mathsf{Pk}, \vec{z})$: The algorithm constructs vector $\vec{x} \in \{0, 1\}^{M_n}$ in the following way: For each $i \in \{1, \ldots, M_n\}$ the algorithms sets $x_i = 1$ if C_i is satisfied by $\vec{z}$; $x_i = 0$ if C_i is not satisfied by $\vec{z}$. We denote this transformation by $\vec{x} = \mathsf{AEncode}(\vec{z})$. Then the encryption algorithm returns $\mathsf{Ct} = \mathsf{Encrypt}_{\mathcal{H}}(\mathsf{Pk}, \vec{x})$.

$\mathsf{Test}_{\mathsf{kCNF}}(\mathsf{Sk}_\Phi, \mathsf{Ct})$: The algorithm returns the output of $\mathsf{Test}_{\mathcal{H}}(\mathsf{Sk}_\Phi, \mathsf{Ct})$.

Correctness. Correctness follows from the observation that for formula Φ and assignment $\vec{z}$, we have that $\mathsf{Match}(\mathsf{AEncode}(\vec{z}), \mathsf{FEncode}(\Phi)) = 1$ if and only if $\mathsf{Satisfy}(\Phi, \vec{z}) = 1$.

Security. Let $\mathcal{A}$ be an adversary for kCNF that tries to break the scheme for n variables and consider the following adversary $\mathcal{B}$ for $\mathcal{H}$ that uses $\mathcal{A}$ as a subroutine and tries to break a $\mathcal{H}$ with $\ell = M_n$ by interacting with challenger $\mathcal{C}$. $\mathcal{B}$ receives a Pk for $\mathcal{H}$ and passes it to $\mathcal{A}$. Whenever $\mathcal{A}$ asks for the key for formula Φ, $\mathcal{B}$ constructs $\vec{y} = \mathsf{FEncode}(\Phi)$ and asks $\mathcal{C}$ for a key $\mathsf{Sk}_{\vec{y}}$ for $\vec{y}$ and returns it to $\mathcal{A}$. When $\mathcal{A}$ asks for a challenge by providing truth assignments $\vec{z}_0$ and $\vec{z}_1$, $\mathcal{B}$ simply

computes $\vec{x}_0 = \mathsf{AEncode}(\vec{z}_0)$ and $\vec{x}_1 = \mathsf{AEncode}(\vec{z}_1)$ and gives the pair $(\vec{x}_0, \vec{x}_1)$ to $\mathcal{C}$. $\mathcal{B}$ then returns the challenge ciphertext obtained from $\mathcal{C}$ to $\mathcal{A}$. Finally, $\mathcal{B}$ outputs $\mathcal{A}$'s guess.

First, $\mathcal{B}$'s simulation is perfect. Indeed, we have that if for all $\mathcal{A}$'s queries Φ we have that $\mathsf{Satisfy}(\Phi, \vec{z}_0) = \mathsf{Satisfy}(\Phi, \vec{z}_1)$, then all $\mathcal{B}$'s queries $\vec{y}$ to $\mathcal{C}$ also have the property $\mathsf{Match}(\vec{y}, \vec{x}_0) = \mathsf{Match}(\vec{y}, \vec{x}_1)$. Thus $\mathcal{B}$'s advantage is the same as $\mathcal{A}$'s. By combining the above reduction with our constructions for HVE, we have the following theorems.

Theorem 24. *For any constant $k > 0$, if Assumption 1 and 2 hold for generator $\mathcal{G}$ then there exists a secure encryption scheme for the class of predicates that can be represented by k-CNF formulae.*

H.4 Reducing Disjunctions to HVE

In this section we consider the class of Boolean predicates that can be expressed as a single disjunction. We assume without loss of generality that a disjunction does not contain a variable and its negated.

Let $\mathcal{H} = (\mathsf{Setup}_{\mathcal{H}}, \mathsf{KeyGen}_{\mathcal{H}}, \mathsf{Encrypt}_{\mathcal{H}}, \mathsf{Test}_{\mathcal{H}})$ be an HVE scheme and construct the predicate-only scheme $\vee = (\mathsf{Setup}_{\vee}, \mathsf{KeyGen}_{\vee}, \mathsf{Encrypt}_{\vee}, \mathsf{Test}_{\vee})$ for disjunctions in the following way:

$\mathsf{Setup}_{\vee}(1^{\lambda}, 1^{n})$: the algorithm returns the output of $\mathsf{Setup}_{\mathcal{H}}(1^{\lambda}, 1^{n})$.

$\mathsf{KeyGen}_{\vee}(\mathsf{Msk}, C)$: For a clause C, the key generation algorithm constructs vector $\vec{y} \in \{0, 1, \star\}^n$ in the following way. Let $\vec{w}$ be a truth assignment to the n variables that does not satisfy clause C. For each $i \in \{1, \ldots, n\}$, the algorithms sets $y_i = w_i$ if the i-th variable appears in C; $y_i = \star$ otherwise. We denote this transformation by $\vec{y} = \mathsf{CEncode}(C)$. The output is $\mathsf{Sk}_C = \mathsf{KeyGen}_{\mathcal{H}}(\mathsf{Msk}, \vec{y})$.

$\mathsf{Encrypt}_{\vee}(\mathsf{Pk}, \vec{z})$: The encryption algorithm returns $\mathsf{Ct} = \mathsf{Encrypt}_{\mathcal{H}}(\mathsf{Pk}, \vec{z})$.

$\mathsf{Test}_{\vee}(\mathsf{Sk}_C, \mathsf{Ct})$: The algorithm returns $1 - \mathsf{Test}_{\mathcal{H}}(\mathsf{Sk}_C, \mathsf{Ct})$.

Correctness. It follows from the observation that for a clause C and assignment $\vec{z}$, $\mathsf{Satisfy}(C, \vec{z}) = 1$ if and only if $\mathsf{Match}(\mathsf{CEncode}(C), \vec{z}) = 0$.

Security. It is easy to see that if $\mathcal{H}$ is secure then $\vee$ is secure. In particular, notice that if for $\mathcal{A}$'s query C we have that $\mathsf{Satisfy}(C, \vec{z}_0) = \mathsf{Satisfy}(C, \vec{z}_1) = \xi \in \{0, 1\}$, then for $\mathcal{B}$'s query $\vec{y} = \mathsf{CEncode}(C)$ to $\mathcal{C}$ we have that $\mathsf{Match}(\vec{y}, \vec{z}_0) = \mathsf{Match}(\vec{y}, \vec{z}_1) = 1 - \xi$.

Theorem 25. *If Assumption 1 and 2 hold for generator $\mathcal{G}$ then there exists a secure encryption scheme for the class of predicates that can be represented by a disjunction.*

H.5 Reducing k-DNF to k-CNF

We observe that if Φ is a predicate represented by a k-DNF formula then its negation $\bar{\Phi}$ can be represented by a k-CNF formula. Therefore let $\mathsf{kCNF} = (\mathsf{Setup}_{\mathsf{kCNF}}, \mathsf{KeyGen}_{\mathsf{kCNF}}, \mathsf{Encrypt}_{\mathsf{kCNF}}, \mathsf{Test}_{\mathsf{kCNF}})$ and consider the following scheme $\mathsf{kDNF} = (\mathsf{Setup}_{\mathsf{kDNF}}, \mathsf{KeyGen}_{\mathsf{kDNF}}, \mathsf{Encrypt}_{\mathsf{kDNF}}, \mathsf{Test}_{\mathsf{kDNF}})$. The setup algorithm $\mathsf{Setup}_{\mathsf{kDNF}}$ is the same as $\mathsf{Setup}_{\mathsf{kCNF}}$. The key generation algorithm $\mathsf{Setup}_{\mathsf{kDNF}}$ for predicate Φ represented by a k-DNF simply invokes the key generation algorithm $\mathsf{Setup}_{\mathsf{kCNF}}$ for $\bar{\Phi}$ that can be represented by a k-CNF formula. The encryption algorithm $\mathsf{Encrypt}_{\mathsf{kDNF}}$ is the same

as $\mathsf{Encrypt}_{\mathsf{kCNF}}$. The test algorithm $\mathsf{Test}_{\mathsf{kDNF}}$ on input ciphertext Ct and key for k-DNF formula Φ (that is actually a for k-CNF formula $\bar{\Phi}$) thus $\mathsf{Test}_{\mathsf{kCNF}}$ on Ct and the key and complements the result. Correctness and security can be easily argued as for Disjunctions.

By combining the above reduction with the construction given by Theorem 24.

Theorem 26. *If Assumption* 1 *and* 2 *hold for generator* $\mathcal{G}$ *then there exists a secure encryption scheme for the class of predicates represented by* k*-DNF formulae.*