Efficient Fully Secure Predicate Encryption for Conjunctions, Disjunctions and k-CNF/DNF formulae

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.

March 7, 2012

Abstract

Predicate encryption is an important cryptographic primitive that has found wide applications as it allows for fine-grained key management. In a predicate encryption scheme for a class $\mathbb{P}$ of predicates, the owner of the master secret key can derive a secret key Sk_P for any predicate $P \in \mathbb{P}$. Similarly, when encrypting plaintext M, the sender can specify an attribute vector $\vec{x}$ for the ciphertext Ct. Then, key Sk_P can decrypt all ciphertexts Ct with attribute vector $\vec{x}$ such that $P(\vec{x}) = 1$.

In this paper, we give *fully secure* implementations Conjunctions, Disjunctions and k-CNF/DNF predicates that guarantee the security of the plaintext *and* of the attribute. Our constructions for Disjunctions and Conjunctions are linear in the number of variables. Previous fully secure constructions for Disjunction required time exponential in the number of variables while for Conjunctions the best previous construction was quadratic in the number of variables.

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

[*]Work done while visiting the Department of Computer Science of The Johns Hopkins University.

Contents

1 Introduction and related work

Predicate encryption is an important cryptographic primitive that has been recently studied [2, 4, 5, 7] and that has found wide applications as it allows for fine-grained key management. Roughly speaking, in a predicate encryption scheme for predicate Q the owner of the master secret key Msk can derive secret key Sk_Q, for any predicate Q for a specified class of predicates. In encrypting plaintext M, the sender can specify an *attribute* vector $\vec{x}$ and the resulting ciphertext Ct can be decrypted only by using keys Sk_Q such that $Q(\vec{x}) = 1$. A predicate encryption scheme thus gives the owner of the master secret key control on which ciphertexts can be decrypted and this allows her to delegate the decryption of different types of messages (as specified by the attribute vector) to different entities. Several constructions for specific predicates have been given, starting from the *equality* predicate of [2], to the *hidden vector* predicate of [4] and to the *inner product* predicate of [7]. A construction for the larger class of monotone formulae have been by [5] and then extended to formulae by [12] (see also [8, 11]). These constructions did not guarantee the security of the attribute vector $\vec{x}$. This extra security property is very important for the applications and was guaranteed, in the *selective* model, by the constructions for specific predicates of [4, 7]. Following the recent breakthrough of [15, 9] that gave fully secure implementation of Identity Based Encryption (and of its hierarchical version), Lewko et al. [8] gave fully secure implementation for the inner product predicate.

Our results. We concentrate on *fully secure* implementations of encryption schemes for binary conjunctions and disjunctions and their applications. For conjunctions we adhere to the standard terminology of *hidden vector encryption* (or HVE in short) as introduced by [4]. In a HVE scheme, the ciphertext attributes are vectors $\vec{x} = \langle x_1, \ldots, x_\ell \rangle$ over alphabet $\{0, 1\}$, the keys are vectors $\vec{y} = \langle y_1, \ldots, y_\ell \rangle$ over alphabet $\{0, 1, \star\}$ and we consider the $\mathsf{Match}(\vec{x}, \vec{y})$ predicate defined to be true if and only if, for all i, $y_i \neq \star$ implies $x_i = y_i$. We distinguish two security notions, that we call 0-security and 1-security (see Section 2) differing in the type of keys that the adversary can ask for. Roughly speaking, ξ-security considers adversaries that can ask keys for vectors $\vec{y}$ such that $\mathsf{Match}(\vec{x}_0, \vec{y}) = \mathsf{Match}(\vec{x}_1, \vec{y}) = \xi$ where $\vec{x}_0$ and $\vec{x}_1$ are the challenge vectors chosen by the adversary. The notion of 0-security is known in the literature as *match revealing* security and all previous fully secure constructions are 0-secure. We give a full secure implementation of HVE for *both* notions of security. Our secure implementations of HVE are proved fully secure under non-interactive constant sized (that is, independent of ℓ and of the running time of the adversary) assumptions on bilinear groups of composite order. We stress that our 0-secure construction is more efficient than previous ones and that our 1-secure construction is the first to be proved 1-secure. We prove 1-security by means of a *tight* security reduction; that is, the security proof does not depend on the running time of the adversary. We then show polynomial deterministic reductions to HVE for any predicate represented by a formula in k-CNF, or by a formula in k-DNF, or by a disjunction. We prove that our reduction for k-CNF preserves full ξ-security and thus it results in a scheme in which the key for a formula of m clauses contains exactly m group elements. In addition we prove that our reductions for k-DNF and disjunctions complements security in the sense that if we apply our reductions to a ξ-secure scheme for HVE we obtain a $(1 - \xi)$-secure scheme for k-DNF (or for disjunctions). Furthermore, our reduction for disjunction is linear. Finally, we give a construction of Hierarchical HVE in which the holder of $\mathsf{Sk}_{\vec{y}}$ (the key for vector $\vec{y}$) can create (and give to a third party) the key for any vector $\vec{w}$ that is obtained by instantiating some of the $\star$ entries of $\vec{y}$ to 0 or 1.

Proof Strategy. We achieve full security by means of a proof strategy that elaborates on the dual encryption methodology [15] pointing out, once again, its great applicability to prove full security. Let us now briefly review the strategy. The first step, to achieve ξ-security, consists in projecting the public key to a different subspace in such a way to make it independent from the challenge ciphertext. Then the proof proceeds to show that the secret keys do not help the adversary and, for this, our strategy forks depending on the value of ξ. For $\xi = 0$, we do so by proving that, in the view of the adversary, the valid secret keys are indistinguishable from keys that are random in the subgroup in which the plaintext is embedded. More precisely, keys continue to be valid in the subgroup where the public key was projected, and are random in the other subgroups. On the other hand, for $\xi = 1$ we show that the secret keys do not help the adversary by proving that, in the view of the adversary, the valid secret keys are indistinguishable from keys which do not have a component in the subgroup in which the plaintext is embedded.

Related Work. An implementation of fully secure HVE can be derived from the fully secure construction of inner product of Lewko et al. [8] using the reduction of Katz et al. [7]. We point out though that the Inner Product construction of [8] has a master key of quadratic size and the key generation and the encryption algorithm suffer of an extra quadratic slowdown in the time complexity when compared to ours. Also, we notice that [8] only considered 0-security and [7] is in the selective model. Similar considerations can be made for the recent construction of Inner Product of [11]. We mention that Katz et al. [7] have presented a reduction of CNF to inner product that is polynomial (actually, cubic) when applied to 3-CNF formulae. By composing this reduction with the one from inner product to HVE gives a reduction that can be applied to our HVE implementation. The resulting scheme for 3-CNF still has a quadratic slowdown when compared to ours. Finally, for disjunctions, we mention that [7] have a construction that is exponential in the number of variables. In contrast, our 0-secure and 1-secure constructions are linear in the number of variables.

2 Hidden Vector Encryption and Boolean Satisfaction Encryption

In this section we give formal definitions for Hidden Vector Encryption (HVE) and for Boolean Satisfaction Problem and their security properties. Following [13], for sake of simplicity we present predicate-only definitions instead of full-fledged ones (see [7]). Also, for (H)HVE we present our construction for the binary alphabet, but as outlined in [6], it is possible to modify the construction for larger alphabets without a penalty in the length of the key or the ciphertext.

Hidden Vector Encryption. Let $\vec{x} \in \{0,1\}^\ell$ and $\vec{y} \in \{0,1,\star\}^\ell$ for a given length ℓ. Define the predicate $\mathsf{Match}(\vec{x},\vec{y}) = \text{TRUE}$ if and only if for any $i \in [\ell]$, it holds that $x_i = y_i$ or $y_i = \star$. This predicate is called Hidden Vector Encryption (HVE) and was introduced in [4]. This predicate has like very special case Anonymous IBE but it has many other applications. For a full account of the applications, see [4].

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, 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 $\mathsf{Match}(\vec{x}, \vec{y})$.

Correctness of HVE. We require that for all pairs $(\mathsf{Pk}, \mathsf{Msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$, it holds that for $\vec{x} \in \{0,1\}^\ell$ 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 probability negligible in λ.

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.

Security definitions for HVE. We give two security notions depending on the type of queries the adversary $\mathcal{A}$ is allowed to ask. The notions are formalized through security games $\mathsf{Game}_{\mathsf{Real}}(\xi)$, with $\xi \in \{0,1\}$, between $\mathcal{A}$ and a challenger $\mathcal{C}$. $\mathsf{Game}_{\mathsf{Real}}(\xi)$ consists of a *Setup phase* and of a *Query Answering phase*. In the Query Answering phase, $\mathcal{A}$ can issue any 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 $\mathcal{C}$. More precisely, for $\xi \in \{0,1\}$, we define game $\mathsf{Game}_{\mathsf{Real}}(\xi)$ in the following way.

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

Key Query Answering. For vector $\vec{y}$, $\mathcal{C}$ returns $\mathsf{KeyGen}(\mathsf{Msk}, \vec{y})$.

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

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 $\mathsf{Match}(\vec{x}_0, \vec{y}) = \mathsf{Match}(\vec{x}_1, \vec{y}) = \xi$.

We call such an $\mathcal{A}$ a ξ-adversary and define its advantage $\mathsf{Adv}_{\mathsf{HVE}}^{\mathcal{A},\xi}(\lambda)$ in $\mathsf{Game}_{\mathsf{Real}}(\xi)$ to be the probability of winning minus 1/2.

Definition 2.1 *An Hidden Vector Encryption scheme is* ξ-secure *if for all PPT* ξ*-adversaries* $\mathcal{A}$*, we have that* $\mathsf{Adv}^{\mathcal{A},\xi}_{\mathsf{HVE}}(\lambda)$ *is a negligible function of* λ*.*

In our security definitions we have the extra constraint that each adversary either requests keys which match *both* challenges (this is the case $\xi = 1$) or keys which match *neither* challenges (this is the case $\xi = 0$). We share this limitation on the security model with [8] (which considered only the case of non-matching queries; that is, $\xi = 0$). To the best of our knowledge, it is an open problem to design a scheme that is secure without this extra constraint.

Security Definitions for Boolean Satisfaction Encryption. For Boolean Satisfaction encryption, we have similar games $\mathsf{Game}_{\mathsf{Real}}(\xi)$ that 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 $\mathsf{Satisfy}(\Phi, z_0) = \mathsf{Satisfy}(\Phi, z_1) = \xi$.

We call such an adversary $\mathcal{A}$ a ξ-adversary and define its advantage $\mathsf{Adv}^{\mathcal{A},\xi}_{\mathbb{B}}(\lambda)$ to be the probability of winning minus 1/2.

Definition 2.2 *An Encryption scheme for class* $\mathbb{B}$ *is* ξ-secure *if for all PPT* ξ*-adversaries* $\mathcal{A}$*, we have that* $\mathsf{Adv}^{\mathcal{A},\xi}_{\mathbb{B}}(\lambda)$ *is a negligible function of* λ*.*

3 Complexity Assumptions

Composite order bilinear groups were first used in Cryptography by [3] (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 $\mathcal{I}$ of a bilinear setting. The description $\mathcal{I}$ of the bilinear setting consists of $\mathcal{I} = (N, \mathbb{G}, \mathbb{G}_T, \mathbf{e})$ 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 groups. 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 $N = p_1p_2p_3p_4$ 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, we have 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.

The first assumption that we state is a subgroup-decision type assumption for bilinear settings with groups of order product of four primes. For a generator $\mathcal{G}$ returning bilinear settings of order product of four primes, we define the following distribution. First pick a random bilinear setting $\mathcal{I} = (N, \mathbb{G}, \mathbb{G}_T, \mathbf{e}) \leftarrow \mathcal{G}(1^\lambda)$ and then pick $A_3 \leftarrow \mathbb{G}_{p_3},\ A_{13} \leftarrow \mathbb{G}_{p_1 p_3},\ A_{12} \leftarrow \mathbb{G}_{p_1 p_2},\ A_4 \leftarrow\in \mathbb{G}_{p_4},\ T_1 \leftarrow \mathbb{G}_{p_1 p_3},\ T_2 \leftarrow \mathbb{G}_{p_2 p_3}$. 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

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

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*.*

The second assumption can be seen as the Decision Diffie-Hellman Assumption for composite order groups and is defined as follows. First pick a random bilinear setting $\mathcal{I} = (N, \mathbb{G}, \mathbb{G}_T, \mathbf{e}) \leftarrow \mathcal{G}(1^\lambda)$ and then pick $A_1 \leftarrow \mathbb{G}_{p_1}, A_2 \leftarrow \mathbb{G}_{p_2}, A_3 \leftarrow \mathbb{G}_{p_3}, A_4, B_4, C_4, D_4 \leftarrow \mathbb{G}_{p_4}, \alpha, \beta \leftarrow \mathbb{Z}_{p_1}, T_2 \leftarrow \mathbb{G}_{p_1 p_4}$ 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

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

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*.*

Assumption 3 is a generalization of Assumption 2 in the sense it posits the difficult of deciding if two triplets sharing an element are *both* Diffie-Hellman and it is defined as follows. First pick a random bilinear setting $\mathcal{I} = (N, \mathbb{G}, \mathbb{G}_T, \mathbf{e}) \leftarrow \mathcal{G}(1^\lambda)$ and then pick $A_1 \leftarrow \mathbb{G}_{p_1}, A_2 \leftarrow \mathbb{G}_{p_2}, A_3 \leftarrow \mathbb{G}_{p_3}, A_4, B_4, C_4, D_4, E_4, F_4,\ G_4 \leftarrow \mathbb{G}_{p_4},\ \alpha, \beta, \gamma \leftarrow \mathbb{Z}_{p_1}, T_2 \leftarrow \mathbb{G}_{p_1 p_4}$ and set $T_1 = A_1^{\alpha\beta} \cdot G_4$ and $D = (\mathcal{I}, A_1, A_2, A_3, A_4,\ A_1^{\alpha} \cdot B_4, A_1^{\beta} \cdot C_4, A_1^{\gamma} \cdot D_4, A_1^{\alpha\gamma} \cdot E_4, A_1^{\alpha\beta\gamma} \cdot F_4)$. We define the advantage of an algorithm $\mathcal{A}$ in breaking Assumption 3 to be

$$\mathsf{Adv}_3^{\mathcal{A}}(\lambda) = |\text{Prob}[\mathcal{A}(D, T_1) = 1] - \text{Prob}[\mathcal{A}(D, T_2) = 1]|$$

Assumption 3 *We say that Assumption* 3 *holds for generator* $\mathcal{G}$ *if for all probabilistic polynomial-time algorithms* $\mathcal{A}$*,* $\mathsf{Adv}_3^{\mathcal{A}}(\lambda)$ *is a negligible function of* λ*.*

It is easy to see that Assumption 3 implies Assumption 2.

Our final assumption is again a subgroup-decision type of assumption and it is defined as follows. First pick a random bilinear setting $\mathcal{I} = (N, \mathbb{G}, \mathbb{G}_T, \mathbf{e}) \leftarrow \mathcal{G}(1^\lambda)$ and then pick $A_2 \leftarrow \mathbb{G}_{p_2},\ A_3 \leftarrow \mathbb{G}_{p_3},\ A_4, B_4, \leftarrow \mathbb{G}_{p_4},\ A_{14}, B_{14} \leftarrow \mathbb{G}_{p_1 p_4}$ and set $T_1 = B_{14}$, $T_2 = B_4$ and $D = (\mathcal{I}, A_2, A_3, A_4, A_{14})$. We define the advantage of an algorithm $\mathcal{A}$ in breaking Assumption 4 to be

$$\mathsf{Adv}_4^{\mathcal{A}}(\lambda) = |\text{Prob}[\mathcal{A}(D, T_1) = 1] - \text{Prob}[\mathcal{A}(D, T_2) = 1]|$$

Assumption 4 *We say that Assumption* 4 *holds for generator* $\mathcal{G}$ *if for all probabilistic polynomial-time algorithms* $\mathcal{A}$*,* $\mathsf{Adv}_4^{\mathcal{A}}(\lambda)$ *is a negligible function of* λ*.*

4 Constructing 0-secure HVE

In this section we describe our construction for a 0-secure (also called match revealing) HVE scheme. We assume without loss of generality that the vectors $\vec{y}$ of the keys have at least two indices i, j such that $y_i, y_j \neq \star$.

$\mathsf{Setup}(1^\lambda, 1^\ell)$: The setup algorithm chooses a description of a bilinear group $\mathcal{I} = (N = p_1p_2p_3p_4, \mathbb{G}, \mathbb{G}_T, \mathbf{e}) \leftarrow \mathcal{G}(1^\lambda)$ with known factorization. and random $g_1 \in \mathbb{G}_{p_1}$, $g_2 \in \mathbb{G}_{p_2}$, $g_3 \in \mathbb{G}_{p_3}$, $g_4 \in \mathbb{G}_{p_4}$. For $i \in [\ell]$ and $b \in \{0,1\}$, the algorithm chooses 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 $Y_i = g_{12}^{a_i/t_{i,y_i}} W_i$. The algorithm returns the tuple $(Y_i)_{i\in S_{\vec{y}}}$. Notice that here we used 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 $X_i = T_{i,x_i}^s Z_i$, and returns the tuple $(X_i)_{i\in[\ell]}$.

$\mathsf{Test}(\mathsf{Ct}, \mathsf{Sk}_{\vec{y}})$**:** The test algorithm computes $T = \prod_{i\in S_{\vec{y}}} \mathbf{e}(X_i, Y_i)$ and returns TRUE iff $T = 1$.
Correctness. It is easy to verify the correctness of the scheme.

4.1 Security of our HVE scheme

In this section we prove that our HVE scheme is 0-secure. To prove security we rely on Assumptions 1 and 2. For a probabilistic polynomial-time 0-adversary $\mathcal{A}$ which makes q queries for KeyGen, our proof of security will be structured as a sequence of $q + 2$ games between $\mathcal{A}$ and a challenger $\mathcal{C}$. The first game, $\mathsf{Game}_{\mathsf{Real}}$, is the real HVE security game described in the previous section. The remaining games, called $\mathsf{Game}_0, \ldots, \mathsf{Game}_q$, are described (and shown indistinguishable) in the following sections. In the rest of this section, when we refer to adversaries we mean 0-adversaries and when we refer to $\mathsf{Game}_{\mathsf{Real}}$ we mean $\mathsf{Game}_{\mathsf{Real}}(0)$.

4.1.1 Proof of indistinguishability of $\mathsf{Game}_{\mathsf{Real}}$ and Game_0

Description of Game_0. Game_0 is like $\mathsf{Game}_{\mathsf{Real}}$, except that $\mathcal{C}$ uses g_2 instead of g_1 to construct the public parameters Pk given to $\mathcal{A}$. Specifically,
Setup. $\mathcal{C}$ chooses a description of a bilinear group $\mathcal{I} = (N = p_1p_2p_3p_4, \mathbb{G}, \mathbb{G}_T, \mathbf{e}) \leftarrow \mathcal{G}(1^\lambda)$ with known factorization and 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 $T_{i,b} = g_2^{t_{i,b}} \cdot R_{i,b}$ and $T'_{i,b} = g_1^{t_{i,b}} \cdot R_{i,b}$. 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, $\mathcal{C}$ interacts with $\mathcal{A}$ on input Pk.
Key Query Answering. On vector $\vec{y}$, $\mathcal{C}$ returns the output of $\mathsf{KeyGen}(\mathsf{Msk}, \vec{y})$.
Challenge Construction. $\mathcal{C}$ picks one of the two challenge vectors provided by $\mathcal{A}$ and encrypts it with respect to public parameters Pk'.

Lemma 4.1 *Suppose there exists a PPT algorithm $\mathcal{A}$ such that $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{Game}_{\mathsf{Real}}} - \mathsf{Adv}^{\mathcal{A}}_{\mathsf{Game}_0} = \epsilon$. Then, there exists a PPT algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption* 1.

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 $\mathsf{Game}_{\mathsf{Real}}$ or Game_0 with $\mathcal{A}$. This suffices to prove the Lemma.

Setup. $\mathcal{B}$ starts by constructing Pk and Pk' as follow. $\mathcal{B}$ sets $g_3 = A_3, g_{12} = A_{12}, g_4 = A_4$ 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.

Key Query Answering. Whenever $\mathcal{A}$ asks to see the secret key $\mathsf{Sk}_{\vec{y}}$ associated with vector $\vec{y}$, $\mathcal{B}$ runs algorithm KeyGen on input Msk and $\vec{y}$.

Challenge Construction. The challenge is created by $\mathcal{B}$ by picking one of the two vectors provided by $\mathcal{A}$, let us call it $\vec{x}$, and by encrypting it by running the $\mathsf{Encrypt}$ algorithm on input $\vec{x}$ 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 $\mathsf{Game}_{\mathsf{Real}}$. Furthermore, it's easy to see that 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 $\mathsf{Game}_{\mathsf{Real}}$. We can thus conclude that $\mathcal{C}$ has simulated $\mathsf{Game}_{\mathsf{Real}}$ with $\mathcal{A}$.

Instead, when $T \in \mathbb{G}_{p_2p_3}$, Pk provided by $\mathcal{B}$ has the same distribution as that produced by $\mathcal{C}$ in Game_0. Therefore, $\mathcal{C}$ is simulating Game_0 for $\mathcal{A}$. □

4.1.2 Proof of indistinguishability of Game_{k-1} and Game_k

Description of Game_k, for $1 \le k \le q$. Each of these games is like Game_0, except that the first k key queries issued by $\mathcal{A}$ are answered with keys whose $\mathbb{G}_{p_1}$ parts are random. The remaining key queries (that is, from the $(k+1)$-st to the q-th) are answered like in the previous game. The $\mathbb{G}_{p_2}$ parts of all the answers to key queries are like in Game_0. More precisely, in Game_k, the Setup phase and the Challenge Construction are like in Game_0 and the Key Query phase is the following.

Key Query Answering. $\mathcal{C}$ answers the first k key queries in the following way. On input vector $\vec{y}$, for $i \in S_{\vec{y}}$, $\mathcal{C}$ chooses random $a_i, c_i \in \mathbb{Z}_N$ under the constraint that $\sum_{i\in S_{\vec{y}}} a_i = 0$ and random $W_i \in \mathbb{G}_{p_4}$. $\mathcal{C}$ sets, for $i \in S_{\vec{y}}$, $Y_i = g_1^{c_i} \cdot g_2^{a_i/t_{i,y_i}} \cdot W_i$. The remaining $q-k$ queries are answered like in Game_0.

Lemma 4.2 *Suppose there exists a PPT algorithm $\mathcal{A}$ such that $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{Game}_{k-1}} - \mathsf{Adv}^{\mathcal{A}}_{\mathsf{Game}_k} = \epsilon$. Then, there exists a PPT algorithm $\mathcal{B}$ with advantage at least $\epsilon/(2\ell)$ in breaking Assumption* 2.

Proof. $\mathcal{B}$ receives $(\mathcal{I}, A_1, A_2, A_3, A_4, A_1^{\alpha} \cdot B_4, A_1^{\beta} \cdot C_4)$ and T and, depending on the nature of T, simulates Game_{k-1} or Game_k with $\mathcal{A}$.

$\mathcal{B}$ starts by guessing the index j such that the j-th bit $y_j^{(k)}$ of the k-th query $\vec{y}^{(k)}$ is different from $\star$ and different from the j-th bit x_j of the challenge vectors provided by $\mathcal{A}$ that $\mathcal{C}$ uses to construct the challenge ciphertext. Notice that the probability that $\mathcal{B}$ correctly guesses j and $y_j^{(k)}$ is at least $1/(2\ell)$, independently from the view of $\mathcal{A}$. Notice that, if during the simulation this is not the case, then $\mathcal{B}$ aborts the simulation and fails. We next describe and prove the correctness of the simulation under the assumption that $\mathcal{B}$'s initial guess is correct. Notice that if the initial guess is correct x_j and $y_j^{(k)}$ are uniquely determined and it holds that $x_j = 1 - y_j^{(k)}$.

Setup. $\mathcal{B}$ sets $g_1 = A_1$, $g_2 = A_2$, $g_3 = A_3$, $g_4 = A_4$ and $g_{12} = A_1 \cdot A_2$. For each $i \in [\ell] \setminus \{j\}$ and $b \in \{0,1\}$, $\mathcal{B}$ chooses random $t_{i,b} \in \mathbb{Z}_N$ and $R_{i,b} \in \mathbb{G}_{p_3}$, and sets $T_{i,b} = g_2^{t_{i,b}} \cdot R_{i,b}$. Moreover, $\mathcal{B}$ chooses random $t_{j,x_j} \in Z_N, R_{j,x_j} \in \mathbb{G}_{p_3}$, $r_{j,y_j^{(k)}} \in Z_N$ and $R_{j,y_j^{(k)}} \in \mathbb{G}_{p_3}$ and sets

$$T_{j,x_j} = g_2^{t_{j,x_j}} \cdot R_{j,x_j} \qquad T_{j,y_j^{(k)}} = g_2^{r_{j,y_j^{(k)}}} \cdot R_{j,y_j^{(k)}}.$$

Notice that by assumption $x_j \neq y_j^{(k)}$. $\mathcal{B}$ then sets $\mathsf{Pk} = [N, g_3, (T_{i,b})_{i\in[\ell],b\in\{0,1\}}]$. In addition, for each $i \in [\ell] \setminus \{j\}$ and $b \in \{0,1\}$ and $\mathcal{B}$ chooses random $R'_{i,b} \in \mathbb{G}_{p_3}$ and sets $T'_{i,b} = g_1^{t_{i,b}} \cdot R'_{i,b}$. Moreover $\mathcal{B}$ chooses random R_{j,x_j} and sets $T'_{j,x_j} = g_1^{t_{x,x_j}} \cdot R'_{j,x_j}$. The value of $T'_{j,y_j^{(k)}}$ remains unspecified. As we shall see below, in answering key queries, $\mathcal{B}$ will implicitly set $T'_{j,y_j^{(k)}} = g_1^{1/\beta} \cdot R'_{j,y_j^{(k)}}$ for a random $R'_{j,y_j^{(k)}} \in \mathbb{G}_{p_3}$. $\mathcal{B}$ starts the interaction with $\mathcal{A}$ on input Pk. Notice that Pk has the same distribution as that seen by $\mathcal{A}$ in Game_{k-1} and Game_k.

Key Query Answering. For the first $k-1$ queries $\mathcal{B}$ proceeds as follows. Let $\vec{y}$ be the input vector. For $i \in S_{\vec{y}}$, $\mathcal{B}$ chooses random a_i such that $\sum_{i\in S_{\vec{y}}} a_i = 0$, random $z_i \in \mathbb{Z}_N$, and random $W_i \in \mathbb{G}_{p_4}$. Then, for $i \in S_{\vec{y}}$, $\mathcal{B}$ computes

$$Y_i = \begin{cases} g_1^{z_i} \cdot g_2^{a_i/t_{i,y_i}} \cdot W_i, & \text{if } i \neq j; \\ g_1^{z_j} \cdot g_2^{a_j/t_{j,y_j}} \cdot W_j, & \text{if } i = j \text{ and } y_j = x_j; \\ g_1^{z_j} \cdot g_2^{a_j/r_{j,y_j}} \cdot W_j, & \text{if } i = j \text{ and } y_j \neq \star. \end{cases}$$

Also notice that the first $k-1$ answers produced by $\mathcal{B}$ have the same distribution as the first $k-1$ answers seen by $\mathcal{A}$ in Game_{k-1} and Game_k.

Let us now describe how $\mathcal{B}$ answers the k-th query the vector $\vec{y}^{(k)}$. Let h be an index such that $h \neq j$ and $y_h^{(k)} \neq \star$; such an index always exists by our assumption that all keys are for vectors with at least two entries different from $\star$. Also we remind the reader that $y_j^{(k)} = 1 - x_j$.

Let $S = S_{\vec{y}} \setminus \{j, h\}$. For each $i \in S$, $\mathcal{B}$ chooses random $a_i \in \mathbb{Z}_N$ and $W_i \in \mathbb{G}_{p_4}$. Moreover $\mathcal{B}$ chooses random $a'_j \in \mathbb{Z}_N$ and $W_j, W_h \in \mathbb{G}_{p_4}$ and sets

$$Y_i = g_{12}^{a_i/t_{i,y_i^{(k)}}} \cdot W_i, \quad Y_j = T \cdot g_2^{a'_j/r_{j,y_j^{(k)}}} \cdot W_j,$$

$$Y_h = (A_1^{\alpha} B_4)^{-1/t_{h,y_h^{(k)}}} \cdot g_1^{-s/t_{h,y_h^{(k)}}} \cdot g_2^{-(s+a_j)/t_{h,y_h^{(k)}}} \cdot W_h,$$

where $s = \sum_{i\in S} a_i$. This terminates the description of how $\mathcal{B}$ handles the k-th key query. Let us now verify that when $T = A_1^{\alpha\beta} \cdot D_4$ then $\mathcal{B}$'s answer to the k-th key query is like in Game_{k-1}. By our settings, we have that $Y_j = g_1^{\alpha/t'_{j,y_j^{(k)}}} \cdot g_2^{a'_j/r_{j,y_j^{(k)}}} \cdot D_4 \cdot W_j$ with $t'_{j,y_j^{(k)}} = 1/\beta$. By the Chinese Remainder Theorem, we can conclude that the answer to the k-th query of $\mathcal{A}$ is distributed as in Game_{k-1}. Instead, if T is random in $\mathbb{G}_{p_1p_4}$ then 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 Game_k.

For the l-th key queries for $l = k+1, \ldots, q$, notice that if the j-th bit of the l-th query vector is equal to x_j then $\mathcal{B}$ has all the t_{i,y_i}'s needed for running algorithm KeyGen. If this is not the case then, by the previous settings, $t_{j,y_j} \equiv 1/\beta \bmod p_1$ and $\mathcal{B}$ can use $A_1^{\beta} \cdot C_4 = g_1^{1/t_{j,y_j}} \cdot C_4$ (see Assumption 2). So, the answers to the last $q-k$ queries have the same distribution as in Game_k and Game_{k-1}.

Challenge Construction. The challenge is created by running algorithm $\mathsf{Encrypt}$ on input the randomly chosen challenge vector $\vec{x}$ and Pk'. Under the assumption that $\mathcal{B}$ has correctly guessed x_j and thus $x_j = 1-y_j^{(k)}$, Pk' contains all the values to compute an encryption of $\vec{x}$. Then the challenge ciphertext is distributed exactly like in Game_{k-1} and Game_k. $\square$

4.1.3 Game_q gives no advantage.

We observe that in Game_q the $\mathbb{G}_{p_1}$ part of the challenge ciphertext is the only one depending on η and the Pk and the answer to the key queries give no help to $\mathcal{A}$. Therefore we can conclude that for all adversaries $\mathcal{A}$, $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{Game}_q} = 0$. We have thus proved.

Theorem 4.3 *If Assumptions* 1 *and* 2 *hold for generator $\mathcal{G}$, then the HVE scheme presented is* 0*-secure (also called match revealing secure).*

5 Constructing 1-secure HVE

In this section we describe our construction for a 1-secure HVE scheme.

$\mathsf{Setup}(1^\lambda, 1^\ell)$*:* The setup algorithm chooses a description of a bilinear group $\mathcal{I} = (N = p_1p_2p_3p_4, \mathbb{G}, \mathbb{G}_T, \mathbf{e}) \leftarrow \mathcal{G}(1^\lambda)$ with known factorization and chooses random $g_1 \in \mathbb{G}_{p_1}$, $g_2 \in \mathbb{G}_{p_2}$, $g_3 \in \mathbb{G}_{p_3}$, $g_4 \in \mathbb{G}_{p_4}$. For $i \in [\ell]$ and $b \in \{0,1\}$, the algorithm chooses random $t_{i,b} \in Z_N$, random $v_i \in Z_N$ and random $R_{i,b} \in \mathbb{G}_{p_3}$ and sets $T_{i,b} = g_1^{t_{i,b}} \cdot g_4^{v_i} \cdot R_{i,b}$. 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 $\mathsf{Msk} = [g_{12} = g_1 \cdot g_2, g_4, (t_{i,b})_{i\in[\ell], b\in\{0,1\}}, (v_i)_{i\in[\ell]}]$.

$\mathsf{KeyGen}(\mathsf{Msk}, \vec{y})$*:* Let $S_{\vec{y}}$ be the set of indices i such that $y_i \neq \star$. For $i \in S_{\vec{y}}$, the key generation algorithm chooses random $a_i \in \mathbb{Z}_N$ such that $\sum_{i\in S_{\vec{y}}} a_i = 0$. For $i \in S_{\vec{y}}$, the algorithm sets $Y_i = g_{12}^{a_i/t_{i,y_i}} g_4^{a_i/v_i}$. The algorithm returns the tuple $(Y_i)_{i\in S_{\vec{y}}}$. Notice that here we used 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 $Z_i \in \mathbb{G}_{p_3}$, sets $X_i = T_{i,x_i}^s Z_i$, and returns $(X_i)_{i\in[\ell]}$.

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

Correctness. It is easy to verify the correctness of the scheme.

5.1 Security of our HVE scheme

To prove that our HVE scheme is 1-secure, we rely on static Assumptions 1 and 4. For a probabilistic polynomial-time 1-adversary $\mathcal{A}$ our proof of security will be structured as a sequence of 2 games between $\mathcal{A}$ and a challenger $\mathcal{C}$. The first game, $\mathsf{Game}_{\mathsf{Real}}(1)$, is the real HVE security game described in the previous section. The remaining games, called $\mathsf{Game}_0, \mathsf{Game}_1$, are described (and shown indistinguishable) in the following sections.

In the rest of this section, when we refer to adversaries we mean 1-adversaries and when we refer to $\mathsf{Game}_{\mathsf{Real}}$ we mean $\mathsf{Game}_{\mathsf{Real}}(1)$.

5.1.1 Proof of indistinguishability of $\mathsf{Game}_{\mathsf{Real}}$ and Game_0

Description of Game_0. Game_0 is like $\mathsf{Game}_{\mathsf{Real}}$, except that $\mathcal{C}$ uses g_2 instead of g_1 to construct the public parameters Pk given to $\mathcal{A}$. Specifically,

Setup. $\mathcal{C}$ chooses random $g_1 \in \mathbb{G}_{p_1}, g_2 \in \mathbb{G}_{p_2}, g_3 \in \mathbb{G}_{p_3}, g_4 \in \mathbb{G}_{p_4}$. For $i \in [\ell]$ and $b \in \{0,1\}$, $\mathcal{C}$ chooses random $t_{i,b} \in Z_N$, random $v_i \in Z_N$ and random $R_{i,b} \in \mathbb{G}_{p_3}$ and sets $T_{i,b} = g_2^{t_{i,b}} \cdot g_4^{v_i} \cdot R_{i,b}$ and $T'_{i,b} = g_1^{t_{i,b}} \cdot g_4^{v_i} \cdot R_{i,b}$. Then $\mathcal{C}$ sets $\mathsf{Pk} = [N, g_3, (T_{i,b})_{i\in[\ell],b\in\{0,1\}}]$ and $\mathsf{Pk}' = [N, g_3, (T'_{i,b})_{i\in[\ell],b\in\{0,1\}}]$. $\mathcal{C}$ starts the interaction with $\mathcal{A}$ on input Pk.

Key Query Answering. On a query for vector $\vec{y}$, $\mathcal{C}$ returns the output of KeyGen on input $\vec{y}$ and $\mathsf{Msk} = [g_{12}, g_4, (t_{i,b})_{i\in[\ell],b\in\{0,1\}}, (v_i)_{i\in[\ell]}]$, where $g_{12} = g_1 \cdot g_2$.

Challenge Construction. $\mathcal{C}$ picks one of the two challenge vectors provided by $\mathcal{A}$ and encrypts it with respect to public parameters Pk'.

Lemma 5.1 *Suppose there exists a PPT algorithm $\mathcal{A}$ such that $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{Game}_{\mathsf{Real}}} - \mathsf{Adv}^{\mathcal{A}}_{\mathsf{Game}_0} = \epsilon$. Then, there exists a PPT algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption* 1.

The proof follows the line of that of Lemma 4.1 and therefore we omit the details.

5.1.2 Proof of indistinguishability of Game_0 and Game_1.

Description of Game_1. This game is like Game_0, except that in the answers provided by $\mathcal{C}$ the key queries. Specifically the queries are answered without the $\mathbb{G}_{p_1}$ part. The $\mathbb{G}_{p_2}$ part of all answers is like in Game_0. Specifically, we have.

Query answering. $\mathcal{C}$ answers the queries in the following way. On input vector $\vec{y}$, for $i \in S_{\vec{y}}$, $\mathcal{C}$ chooses random $a_i, b_i \in \mathbb{Z}_N$ under the constraint that $\sum_{i\in S_{\vec{y}}} a_i = \sum_{i\in S_{\vec{y}}} b_i = 0$. $\mathcal{C}$ sets, for $i \in S_{\vec{y}}$, $Y_i = g_2^{a_i/t_{i,y_i}} \cdot g_4^{b_i/v_i} \cdot W_i$.

Lemma 5.2 *Suppose there exists a PPT algorithm $\mathcal{A}$ such that $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{Game}_0} - \mathsf{Adv}^{\mathcal{A}}_{\mathsf{Game}_1} = \epsilon$. Then, there exists a PPT algorithm $\mathcal{B}$ with advantage at least ϵ in breaking Assumption 4.*

PROOF. $\mathcal{B}$ receives $(\mathcal{I}, A_2, A_3, A_4, A_{14})$ and T and, depending on the nature of T, simulates Game_0 or Game_1 with $\mathcal{A}$.

Setup. $\mathcal{B}$ sets $g_2 = A_2$, $g_3 = A_3$, $g_4 = A_4$. For $i \in [\ell]$ and $b \in \{0,1\}$, $\mathcal{B}$ chooses random $t_{i,b}, v_i \in \mathbb{Z}_N$ and $R_{i,b} \in \mathbb{G}_{p_3}$, and sets $T_{i,b} = g_2^{t_{i,b}} \cdot g_4^{v_i} \cdot R_{i,b}$. These settings determine $\mathsf{Pk} = [N, g_3, (T_{i,b})_{i\in[\ell],b\in\{0,1\}}]$. used by $\mathcal{B}$ to interact with $\mathcal{A}$. Notice that Pk has the same distribution as that seen by $\mathcal{A}$ in Game_0 and Game_1.

Key Query Answering. $\mathcal{B}$ computes the answer to query for vector $\vec{y}$ as follows. For $i \in S_{\vec{y}}$, $\mathcal{B}$ chooses random $a_i \in \mathbb{Z}_N$ subject to $\sum_{i\in S_{\vec{y}}} a_i = 0$ and sets $Y_i = g_2^{a_i/t_{i,y_i}} \cdot T^{a_i/v_i}$. Now suppose $T = B_{14}$ and write $T = g_1 g_4^c$ for some $g_1 \in \mathbb{G}_{p_1}$ and $c \in \mathbb{Z}_{p_4}$. By our setting we have $Y_i = g_1^{a_i/v_i} \cdot g_2^{a_i/t_{i,y_i}} \cdot g_4^{ca_i/v_i}$ which implicitly sets $t_{i,y_i} \equiv v_i \mod p_1$. It is easy to see that the answer to the query is distributed as in Game_0. Instead, if $T = B_4$ then the key does not contain the $\mathbb{G}_{p_1}$ part and thus the answer to the query is distributed as in Game_1.

Notice that, since $\mathcal{A}$ is a 1-adversary, then for every query vector $\vec{y}$ that $\mathcal{A}$ can submit, it holds that for each $i \in [\ell]$, $y_i = \star$ or $y_i = x_{0,i} = x_{1,i}$. Therefore, or each $i \in [\ell]$, $\mathcal{B}$ needs only to determine $t_{i,x_{0,i}} = t_{i,x_{1,i}}$ and it does so by setting it congruent to $v_i \mod p_1$.

Challenge Construction. For $i \in [\ell]$, $\mathcal{B}$ chooses random $Z_i \in \mathbb{G}_{p_3}$ and sets, for $i \in [\ell]$, $X_i = A_{14}^{v_i} \cdot Z_i$. Finally notice that by writing $A_{14} = (g_1 \cdot g_4^c)^s$, the challenge ciphertext is distributed exactly like in Game_0 and Game_1. □

5.1.3 Game_1 gives no advantage.

We observe that the $\mathbb{G}_{p_1}$ part of the challenge ciphertext is the only one depending on η and the Pk and the answer to the key queries give no help to $\mathcal{A}$. Therefore we can conclude that for all adversaries $\mathcal{A}$, $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{Game}_1} = 0$. We have thus proved.

Theorem 5.3 *If Assumptions* 1 *and* 4 *hold for generator $\mathcal{G}$, then the HVE scheme presented is* 1*-secure.*

5.2 Merging the schemes

It is easy to see that our 1-secure HVE scheme can be extended to a scheme that is also 0-secure. This can be done by using a bilinear instance of order product of five primes and by using the new subgroup to randomize the secret keys. The proof of the 0-security of the resulting scheme is very similar to that provided in section 4. We point out that this does not mean that the resulting scheme is secure against adversaries that can request *both* matching and non-matching queries.

6 Reductions

6.1 Reducing k-CNF to HVE.

In this section we show how to construct a ξ-secure Encryption scheme for the class of Boolean predicates that can be expressed as a k-CNF formula from a ξ-secure HVE scheme. 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 a ξ-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 $\mathsf{Satisfy}(\Phi, \vec{z}_0) = \mathsf{Satisfy}(\Phi, \vec{z}_1) = \xi$, then all $\mathcal{B}$'s queries $\vec{y}$ to $\mathcal{C}$ also have the property $\mathsf{Match}(\vec{y}, \vec{x}_0) = \mathsf{Match}(\vec{y}, \vec{x}_1) = \xi$. 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 6.1 *For any constant $k > 0$, if Assumption 1 and 2 hold for generator $\mathcal{G}$ then there exists a 0-secure encryption scheme for the class of predicates that can be represented by k-CNF formulae. Moreover, If Assumption 1 and 4 hold for generator $\mathcal{G}$ then there exists a 1-secure encryption scheme for the class of predicates that can be represented by k-CNF formulae.*

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 $\vec{z}$, $\mathsf{Satisfy}(\vec{z}, C) = 1$ if and only if $\mathsf{Match}(\vec{z}, \mathsf{CEncode}(C)) = 0$.

Security. It is easy to see that if $\mathcal{H}$ is $(1-\xi)$-secure then $\vee$ is ξ-secure. Indeed notice that we can transform any ξ-adversary $\mathcal{A}$ for $\vee$ into a $(1 - \xi)$-adversary $\mathcal{B}$ for $\mathcal{H}$ in the obvious way and that any ξ-query of $\mathcal{A}$ for a key for $\vee$ can be answered by making a $(1-\xi)$-query for $\mathcal{H}$. By applying the above reduction to the 0-secure and 1-secure HVE construction of the previous sections we obtain the following theorem.

Theorem 6.2 *If Assumption 1 and 4 hold for generator $\mathcal{G}$ then there exists a 0-secure encryption scheme for the class of predicates that can be represented by a disjunction. Moreover, if Assumption 1 and 2 hold for generator $\mathcal{G}$ then there exists a 1-secure encryption scheme for the class of predicates that can be represented by a disjunction.*

6.2 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 can be easily argued. We notice that this reduction however does not preserve ξ-security but rather complements it. More precisely, for proving ξ-security we can easily see that any ξ-adversary for kDNF can be used to construct a $(1-\xi)$-adversary for kCNF.

By combining the above reduction with the construction given by Theorem 6.1.

Theorem 6.3 *If Assumption* 1 *and* 2 *hold for generator* $\mathcal{G}$ *then there exists a* 1*-secure encryption scheme for the class of predicates represented by k-DNF formulae.*

If Assumption 1 *and* 4 *hold for generator* $\mathcal{G}$ *then there exists a* 0*-secure encryption scheme for the class of predicates represented by k-DNF formulae.*

7 Open problems and future work

We leave as a future work the implementation of a symmetric-key version of our encryption schemes (see for example [13]).

We proved the full security in two models: in the case in which the adversary can request keys which do not satisfy both the challenges (0-security, which is the only notion considered in [8] and subsequent works) and in the case in which the adversary can request keys which satisfy both the challenges (1-security). It would be interesting to have a construction that is secure against adversaries allowed to request keys which either satisfy both challenges or satisfy neither (match concealing model).

We gave a tight reduction that does not depend on the running time (and number of queries q) for the case of our 1-secure scheme. It is an open problem the design of a scheme with a tight security reduction for 0-secure schemes.

References

[1] Dan Boneh. Bilinear groups of composite order. *Pairing 2007.* Volume 4575 of *LNCS*, pages 39–56, Tokyo, Japan, July 2–4. Springer-Verlag, Berlin, Germany.

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

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

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

[5] 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, Oct. 30 - Nov. 3, 2006. ACM Press.

[6] Vincenzo Iovino and Giuseppe Persiano. Hidden-vector encryption with groups of prime order. *Pairing 2008*. Volume 5209 of *LNCS*, pages 75–88, Egham, UK, September 1–3, 2008. Springer-Verlag, Berlin, Germany.

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

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

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

[10] Tatsuaki Okamoto and Katsuyuki Takashima. Hierarchical predicate encryption for inner-products. *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 214–231, Tokyo, Japan, December 6–10, 2009. Springer-Verlag, Berlin, Germany.

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

[12] Rafail Ostrovsky, Amit Sahai, and Brent Waters. Attribute-based encryption with non-monotonic access structures. *ACM CCS 07*, pages 195–203, Alexandria, VA, USA, October 28 - 31, 2007. ACM Press.

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

[14] Elaine Shi and Brent Waters. Delegating capabilities in predicate encryption systems. *ICALP 2008*, volume 5126 of *LNCS*, pages 560–578, Reykjavik, Iceland, July 7–11, 2008. Springer-Verlag, Berlin, Germany.

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