Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products

Jonathan Katz
jkatz@cs.umd.edu

Amit Sahai
sahai@cs.ucla.edu

Brent Waters
bwaters@csl.sri.com

Abstract

Predicate encryption is a new paradigm for public-key encryption generalizing, among other things, identity-based encryption. In a predicate encryption scheme, secret keys correspond to *predicates* and ciphertexts are associated with *attributes*; the secret key SK_f corresponding to a predicate f can be used to decrypt a ciphertext associated with attribute I if and only if $f(I) = 1$. Constructions of such schemes are currently known for certain classes of predicates.

We construct such a scheme for predicates corresponding to the evaluation of *inner products* over $\mathbb{Z}_N$ (for some large integer N). This, in turn, enables constructions in which predicates correspond to the evaluation of disjunctions, polynomials, CNF/DNF formulae, or threshold predicates (among others). Besides serving as a significant step forward in the theory of predicate encryption, our results lead to a number of applications that are interesting in their own right.

1 Introduction

Traditional public-key encryption is coarse-grained: a sender encrypts a message M with respect to a public key PK, and only the owner of the (unique) secret key associated with PK can decrypt the resulting ciphertext and recover the message. These straightforward semantics suffice for point-to-point communication, where encrypted data is intended for one particular recipient who is known in advance to the sender. In other settings, however, the sender may instead want to define a *policy* determining who is allowed to recover the encrypted data. For example, classified data might be associated with certain keywords; this data should be accessible both to users who are allowed to read *all* classified information, as well as to users allowed to read information associated with the particular keywords in question. Or, perhaps a patient's records should be accessible only to physicians who have treated that patient in the past. In other applications, it may be sufficient to detect only whether a certain predicate is satisfied; for example, an email firewall should potentially be able to evaluate whether an encrypted email satisfies certain attributes (so that it can be forwarded appropriately), without learning anything else about the encrypted message.

Applications such as those sketched above require new cryptographic mechanisms that provide more fine-grained control over access to encrypted data. *Predicate encryption* offers one such tool. At a high level (formal definitions are given in Section 2), secret keys in a predicate encryption scheme correspond to predicates (i.e., boolean functions) in some class $\mathcal{F}$, and a sender associates a ciphertext with an attribute in a set Σ; a ciphertext associated with the attribute $I \in \Sigma$ can be decrypted by a secret key SK_f corresponding to the predicate $f \in \mathcal{F}$ if and only if $f(I) = 1$.

The "basic" level of security achieved by such schemes guarantees, informally, that a ciphertext associated with attribute I hides all information about the underlying message unless one is in possession of a secret key giving the explicit ability to decrypt. I.e., if an adversary $\mathcal{A}$ holds keys $SK_{f_1}, \ldots, SK_{f_\ell}$, then $\mathcal{A}$ learns nothing about the message if $f_1(I) = \cdots = f_\ell(I) = 0$. We refer to this security notion as *payload hiding*. A stronger notion of security, that we call *attribute hiding*, requires that the ciphertext hide the message as above, but additionally requires that the ciphertext hide all information about the associated attribute I except that which is explicitly leaked by the keys in one's possession; i.e., an adversary holding secret keys as above learns only $f_1(I), \ldots, f_\ell(I)$ (and the message, in case one of these evaluates to 1), and learns nothing else about I. See Section 2 for formal definitions.

Much recent work aimed at constructing different types of fine-grained encryption schemes can be cast in the framework of predicate encryption. Identity-based encryption (IBE) [24, 10, 16, 4, 5, 27] can be viewed as predicate encryption for the class of equality tests; the standard notion of security for IBE [10, 15] corresponds to payload hiding, while *anonymous* IBE [9, 13, 17] corresponds to the stronger notion of attribute hiding. Attribute-based encryption schemes [23, 18, 3, 22], as well as a recent scheme handling range queries [25], can also be cast in the framework of predicate encryption, though in this case all the listed constructions achieve payload hiding only. Boneh and Waters [12] construct a predicate encryption scheme that handles conjunctions of, e.g., equality tests; their scheme satisfies the stronger notion of attribute hiding.

Other work introducing concepts related to the idea of predicate encryption includes [2, 1]. In contrast to the present work, however, the threat model in those works do not consider *collusion* among users holding different secret keys.

1.1 Our Results

An important research direction is to construct predicate encryption schemes for predicate classes $\mathcal{F}$ that are as expressive as possible, with the ultimate goal being to handle all polynomial-time predicates. In addition, it is of independent interest to explore constructions of attribute-hiding (in contrast to payload-hiding) schemes. In this work, we make progress in both these directions.

The aim of our work is to construct attribute-hiding schemes handling disjunctions. Most prior work (as surveyed above) yields only payload-hiding schemes, and the existing techniques for obtaining attribute hiding were limited to enforcing *conjunctions*. (Indeed, handling disjunctions was left as an open question in [12].) On a technical level, this is because the underlying cryptographic mechanism used in the schemes enforcing conjunction is to pair components of the secret key with corresponding components of the ciphertext and then multiply the intermediate results together; a "cancelation" in the exponent occurs if everything "matches", but a random group element results if there is any "mismatch". Thus, the holder of a non-matching secret key learns only that there was a mismatch in *at least one* position, but does not learn the number of mismatches or their locations (as required for attribute hiding). On the other hand, very different techniques seem needed to support disjunctions since now a mismatch in a single position should not give a random group element but must instead somehow result in a "cancelation" if there is a match in any other position. (We stress that what makes this difficult when attribute hiding is desired is that we must hide the position of a match and only reveal the existence of a match in at least one position.)

As we have mentioned, the aim of our work is to construct attribute-hiding schemes handling disjunctions. As a stepping stone toward this goal, we first focus on predicates corresponding to the computation of inner products over $\mathbb{Z}_N$ (for some large integer N). Formally, we take $\Sigma = \mathbb{Z}_N^n$ as our set of attributes, and take our class of predicates to be $\mathcal{F} = \{f_{\vec{x}} \mid \vec{x} \in \mathbb{Z}_N^n\}$ where $f_{\vec{x}}(\vec{y}) = 1$ iff $\langle \vec{x}, \vec{y} \rangle = 0$. (Here, $\langle \vec{x}, \vec{y} \rangle$ denotes the standard inner product $\sum_{i=1}^n x_i \cdot y_i \bmod N$ of two vectors $\vec{x}$

and $\vec{y}$.) We construct a predicate encryption scheme for this $\mathcal{F}$ without random oracles, based on two new assumptions in composite-order groups equipped with a bilinear map. Our assumptions are non-interactive and of fixed size (i.e., not "q-type"), and can be shown to hold in the generic group model. A pessimistic interpretation of our results would be that we prove security in the generic group model, but we believe it is of importance that we are able to distill our necessary assumptions to ones that are compact and falsifiable. Our construction uses new techniques, including the fact that we work in a bilinear group whose order is a product of *three* primes.

We view our main construction as a significant step toward increasing the expressiveness of predicate encryption in general. Moreover, we show that any predicate encryption scheme supporting "inner product" predicates as described above can be used as a building block to construct predicates of more general types:

- As an easy warm-up, we show that it implies (anonymous) identity-based encryption as well as hidden-vector encryption [12]. As a consequence, our work implies all the results of [12].
- We can also construct predicate encryption schemes supporting polynomial evaluation. Here, we take $\mathbb{Z}_N$ as our set of attributes, and predicates correspond to polynomials over $\mathbb{Z}_N$ of some bounded degree; a predicate evaluates to 1 iff the corresponding polynomial evaluates to 0 on the attribute in question. We can also extend this to include multi-variate polynomials (in some bounded number of variables). A "dual" of this construction allows the attributes to be polynomials, and the predicates to correspond to evaluation at a fixed point.
- Given the above, we can fairly easily support predicates that are *disjunctions* of other predicates (e.g., equality), thus achieving our main goal. In the context of identity-based encryption, this gives the ability to issue a secret key corresponding to a *set* S of identities that enables decryption whenever a ciphertext is encrypted to any one of the identities in S (without leaking which identity was actually used to encrypt).
- We also show how to handle predicates corresponding to DNF and CNF formulas of some bounded size.
- Working directly with our "inner product" construction, we can derive a scheme supporting threshold queries of the following form: Attributes are subsets of $A = \{1, \ldots, \ell\}$, and predicates take the form $\{f_{S,t} \mid S \subseteq A\}$ where $f_{S,t}(S') = 1$ iff $S \cap S' = t$. This is useful in the "fuzzy IBE" setting of Sahai and Waters [23], and improves on their work in that we achieve attribute hiding (rather than only payload hiding) and handle *exact* thresholds.

We defer further discussion regarding the above until Section 5.

2 Definitions

We define the syntax of predicate encryption and the security properties discussed informally in the Introduction. (Our definitions follow the general framework of those given in [12].) Throughout this section, we consider the general case where Σ denotes an arbitrary set of attributes and $\mathcal{F}$ denotes an arbitrary set of predicates over Σ. Formally, both Σ and $\mathcal{F}$ may depend on the security parameter and/or the master public parameters (and, indeed, this will be the case in our main constructions); for simplicity, we leave this dependence implicit.

Definition 2.1. *A* predicate encryption scheme *for the class of predicates $\mathcal{F}$ over the set of attributes Σ consists of four* PPT *algorithms* Setup, GenKey, Enc, Dec *such that:*

- Setup *takes as input the security parameter* 1^n *and outputs a (master) public key* PK *and a (master) secret key* SK.
- GenKey *takes as input the master secret key* SK *and a (description of a) predicate* $f \in \mathcal{F}$. *It outputs a key* SK_f.
- Enc *takes as input the public key* PK, *an attribute* $I \in \Sigma$, *and a message* M *in some associated message space. It returns a ciphertext* C. *We write this as* $C \leftarrow \mathsf{Enc}_{PK}(I, M)$.
- Dec *takes as input a secret key* SK_f *and a ciphertext* C. *It outputs either a message* M *or the distinguished symbol* $\perp$.

For correctness, we require that for all n, *all* (PK, SK) *generated by* $\mathsf{Setup}(1^n)$, *all* $f \in \mathcal{F}$, *any key* $SK_f \leftarrow \mathsf{GenKey}_{SK}(f)$, *and all* $I \in \Sigma$:

- *If* $f(I) = 1$ *then* $\mathsf{Dec}_{SK_f}(\mathsf{Enc}_{PK}(I, M)) = M$.
- *If* $f(I) = 0$ *then* $\mathsf{Dec}_{SK_f}(\mathsf{Enc}_{PK}(I, M)) = \perp$ *with all but negligible probability.*

A useful variant of the above is a *predicate-only scheme*. Here, Enc takes only an attribute I (and no message), and the correctness requirement is that $\mathsf{Dec}_{SK_f}(\mathsf{Enc}_{PK}(I)) = f(I)$ (except possibly with negligible probability). Actually, our construction achieves "computational correctness" only: namely, that it is hard to find f and I for which $\mathsf{Dec}_{SK_f}(\mathsf{Enc}_{PK}(I)) \neq f(I)$.

Our definition of attribute-hiding security corresponds to the notion described informally earlier. Here, an adversary may request keys corresponding to the predicates $f_1, \ldots, f_\ell$ and is also given either $\mathsf{Enc}_{PK}(I_0, M_0)$ or $\mathsf{Enc}_{PK}(I_1, M_1)$ for attributes I_0, I_1 such that $f_i(I_0) = f_i(I_1)$ for all i. Furthermore, if $M_0 \neq M_1$ then it is required that $f_i(I_0) = f_i(I_1) = 0$ for all i. The goal of the adversary is to determine which attribute/message pair was encrypted, and the stated conditions ensure that this is not trivial. Our definition uses the "selective" notion of security introduced in [15]. Observe that when specialized to the case when $\mathcal{F}$ consists of equality tests on strings, this notion corresponds to *anonymous* identity-based encryption (with selective-ID security).

Definition 2.2. *A predicate encryption scheme with respect to* $\mathcal{F}$ *and* Σ *is* attribute hiding *if for all* PPT *adversaries* $\mathcal{A}$, *the advantage of* $\mathcal{A}$ *in the following experiment is negligible in the security parameter* n:

1. $\mathcal{A}(1^n)$ *outputs* $I_0, I_1 \in \Sigma$.
2. $\mathsf{Setup}(1^n)$ *is run to generate* PK *and* SK, *and the adversary is given* PK.
3. $\mathcal{A}$ *may adaptively request keys for any predicates* $f_1, \ldots, f_\ell \in \mathcal{F}$ *subject to the restriction that* $f_i(I_0) = f_i(I_1)$ *for all* i. *In response,* $\mathcal{A}$ *is given the corresponding keys* $SK_{f_i} \leftarrow \mathsf{GenKey}_{SK}(f_i)$.
4. $\mathcal{A}$ *outputs two equal-length messages* M_0, M_1. *If there is an* i *for which* $f_i(I_0) = f_i(I_1) = 1$, *then it is required that* $M_0 = M_1$. *A random bit* b *is chosen, and* $\mathcal{A}$ *is given the ciphertext* $C \leftarrow \mathsf{Enc}_{PK}(I_b, M_b)$.
5. *The adversary may continue to request keys for additional predicates, subject to the same restrictions as before.*
6. $\mathcal{A}$ *outputs a bit* b', *and succeeds if* $b' = b$.

The advantage of $\mathcal{A}$ *is the absolute value of the difference between its success probability and* $1/2$.

For predicate-only encryption schemes, attribute hiding is define by simply omitting the messages in the above experiment. *Payload hiding*, a strictly weaker notion of security, is defined by forcing

$I_0 = I_1 = I$ in the above experiment (in which case $\mathcal{A}$ has no possible advantage if it ever holds that $f_i(I) = 1$).

3 Background on Pairings and Complexity Assumptions

We assume some familiarity with *bilinear maps* as used, e.g., in [19, 20, 10], though our treatment will be self-contained. We will specifically focus on bilinear groups of *composite* order, first used in cryptographic applications by [11]. In contrast to all prior work using composite-order bilinear groups, however, we use groups whose order N is a product of *three* (distinct) primes.

Let $\mathcal{G}$ be an algorithm that takes as input a security parameter 1^n and outputs a tuple $(p, q, r, \mathbb{G}, \mathbb{G}_T, \hat{e})$ where p, q, r are distinct primes, $\mathbb{G}$ and $\mathbb{G}_T$ are two cyclic groups of order $N = pqr$, and $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a non-degenerate *bilinear map*, i.e., $\forall u, v \in \mathbb{G}$ and $\forall a, b \in \mathbb{Z}$ we have $\hat{e}(u^a, v^b) = \hat{e}(u, v)^{ab}$, and if g generates $\mathbb{G}$ then $\hat{e}(g, g)$ generates $\mathbb{G}_T$. We assume multiplication in $\mathbb{G}$ and $\mathbb{G}_T$, as well as the bilinear map $\hat{e}$, are all computable in time polynomial in n. Furthermore, we assume that the descriptions of $\mathbb{G}$ and $\mathbb{G}_T$ include generators of $\mathbb{G}$ and $\mathbb{G}_T$, respectively.

We use the notation $\mathbb{G}_p, \mathbb{G}_q, \mathbb{G}_r$ to denote the subgroups of $\mathbb{G}$ having order p, q, and r, respectively. Observe that $\mathbb{G} = \mathbb{G}_p \times \mathbb{G}_q \times \mathbb{G}_r$. Note also that if g is a generator of $\mathbb{G}$, then the element g^{pq} is a generator of $\mathbb{G}_r$; the element g^{pr} is a generator of $\mathbb{G}_q$; and the element g^{qr} is a generator of $\mathbb{G}_p$. Furthermore, if $h_p \in \mathbb{G}_p$ and $h_q \in G_q$ then

$$\hat{e}(h_p, h_q) = \hat{e}\left((g^{qr})^{\alpha_1}, (g^{pr})^{\alpha_2}\right) = \hat{e}\left(g^{\alpha_1}, g^{r\alpha_2}\right)^{pqr} = 1,$$

where $\alpha_1 = \log_{g^{qr}} h_p$ and $\alpha_2 = \log_{g^{pr}} h_q$. A similar rule holds whenever $\hat{e}$ is applied to elements in distinct subgroups.

3.1 Our Assumptions

We now state the assumptions we use to prove security of our construction. As remarked earlier, these assumptions are new but we justify them in Appendix A by proving that they hold in the generic group model under the assumption that finding a non-trivial factor of N (the group order) is hard. At a minimum, then, our construction can be viewed as secure in the generic group model. Nevertheless, we state our assumptions explicitly and highlight that they are non-interactive (in contrast to, e.g., the LRSW assumption [14]) and of fixed size (in contrast to, e.g., the q-SDH assumption [6]).

Assumption 1. Let $\mathcal{G}$ be as above. We say that $\mathcal{G}$ satisfies Assumption 1 if the advantage of any PPT algorithm $\mathcal{A}$ in the following experiment is negligible in the security parameter n:

1. $\mathcal{G}(1^n)$ is run to obtain $(p, q, r, \mathbb{G}, \mathbb{G}_T, \hat{e})$. Set $N = pqr$, and let g_p, g_q, g_r be generators of $\mathbb{G}_p$, $\mathbb{G}_q$, and $\mathbb{G}_r$, respectively.
2. Choose random $Q_1, Q_2, Q_3 \in \mathbb{G}_q$, random $R_1, R_2, R_3 \in \mathbb{G}_r$, random $a, b, s \in \mathbb{Z}_p$, and a random bit b. Give to $\mathcal{A}$ the values $(N, \mathbb{G}, \mathbb{G}_T, \hat{e})$ as well as
$$g_p, \;\; g_r, \;\; g_q R_1, \;\; g_p^b, \;\; g_p^{b^2}, \;\; g_p^a g_q, \;\; g_p^{ab} Q_1, \;\; g_p^s, \;\; g_p^{bs} Q_2 R_2.$$
If $b = 0$ give $\mathcal{A}$ the value $T = g_p^{b^2 s} R_3$, while if $b = 1$ give $\mathcal{A}$ the value $T = g_p^{b^2 s} Q_3 R_3$.
3. $\mathcal{A}$ outputs a bit b', and succeeds if $b' = b$.

The advantage of $\mathcal{A}$ is the absolute value of the difference between its success probability and 1/2.

Assumption 2. Let $\mathcal{G}$ be as above. We say that $\mathcal{G}$ satisfies Assumption 2 if the advantage of any PPT algorithm $\mathcal{A}$ in the following experiment is negligible in the security parameter n:

1. $\mathcal{G}(1^n)$ is run to obtain $(p, q, r, \mathbb{G}, \mathbb{G}_T, \hat{e})$. Set $N = pqr$, and let g_p, g_q, g_r be generators of $\mathbb{G}_p$, $\mathbb{G}_q$, and $\mathbb{G}_r$, respectively.
2. Choose random $h \in \mathbb{G}_p$ and $Q_1, Q_2 \in \mathbb{G}_q$, random $s, \gamma \in \mathbb{Z}_q$, and a random bit b. Give to $\mathcal{A}$ the values $(N, \mathbb{G}, \mathbb{G}_T, \hat{e})$ as well as

$$g_p,\;\; g_q,\;\; g_r,\;\; h,\;\; g_p^s,\;\; h^s Q_1,\;\; g_p^\gamma Q_2,\;\; \hat{e}(g_p, h)^\gamma.$$

   If $b = 0$ then give $\mathcal{A}$ the value $\hat{e}(g_p, h)^{\gamma s}$, while if $b = 1$ then give $\mathcal{A}$ a random element of $\mathbb{G}_T$.
3. $\mathcal{A}$ outputs a bit b', and succeeds if $b' = b$.

The advantage of $\mathcal{A}$ is the absolute value of the difference between its success probability and 1/2.

Both the above assumptions imply the hardness of finding any non-trivial factor of N.

4 Our Main Construction

Our main construction is a predicate encryption scheme where the set of attributes is $\Sigma = \mathbb{Z}_N^n$, and the class of predicates is $\mathcal{F} = \{f_{\vec{v}} \mid \vec{v} \in \mathbb{Z}_N^n\}$ with $f_{\vec{v}}(\vec{x}) = 1$ iff $\langle \vec{v}, \vec{x} \rangle = 0 \bmod N$. We will begin by presenting a *predicate-only* version of the scheme, and discuss in Appendix C how to generalize it to a full-fledged predicate encryption scheme in a fairly straightforward manner. Before giving the details, we provide some intuition for the construction.

4.1 Intuition for the Construction

In our construction, each ciphertext has associated with it a (secret) vector $\vec{x}$, and each secret key corresponds to a vector $\vec{v}$. The decryption procedure must check whether $\vec{x} \cdot \vec{v} = 0 \bmod N$, and reveal nothing about $\vec{x}$ but whether this is true. To do this, we will make use of a bilinear group $\mathbb{G}$ whose order N is the product of three primes p, q, and r. Let $\mathbb{G}_p$, $\mathbb{G}_q$, and $\mathbb{G}_r$ denote the subgroups of $\mathbb{G}$ having order p, q, and r, respectively. We will (informally) assume, as in [11], that a random element in any of these subgroups is indistinguishable from a random element of $\mathbb{G}$.[1] Thus, we can use random elements from one subgroup to "mask" elements from another subgroup.

At a high level, we will use these subgroups as follows: $\mathbb{G}_q$ will be used to encode the vectors $\vec{x}$ and $\vec{v}$ in the ciphertext and secret keys, respectively. (This will be done, e.g., in the case of ciphertexts, by putting each element of the vector $\vec{x} = (x_1, \ldots, x_n)$ in the exponent of its own component of the ciphertext.) Computation of the inner product $\langle \vec{v}, \vec{x} \rangle$ will be done in $\mathbb{G}_q$, in the exponent, using the bilinear map. The subgroup $\mathbb{G}_p$ will be used to encode an equation (again in the exponent) that evaluates to zero when decryption is done properly. This subgroup is used to prevent an adversary from improperly "manipulating" the computation (by, e.g., changing the order of components of the ciphertext or secret key, raising these components to some power, etc.). On an intuitive level, if the adversary tries to manipulate the computation in any way, then the computation occurring in the $\mathbb{G}_p$ subgroup will no longer yield the identity (i.e., will no longer yield

[1] This is only for intuition. Our actual computational assumptions are given in Section 3.

0 in the exponent), but will instead have the effect of "masking" the correct answer with a random element of $\mathbb{G}_p$ (which will invalidate the entire computation). Elements in $\mathbb{G}_r$ are used for "general masking" of terms in other subgroups; i.e., random elements of $\mathbb{G}_r$ will be multiplied with various components of the ciphertext (and secret key) in order to "hide" information that might be present in the $\mathbb{G}_p$ and $\mathbb{G}_q$ subgroups.

4.2 A Predicate-Only Encryption Scheme

We now describe our scheme in detail.

$\mathsf{Setup}(1^n)$ The setup algorithm first runs $\mathcal{G}(1^n)$ to obtain $(p, q, r, \mathbb{G}, \mathbb{G}_T, \hat{e})$ with $\mathbb{G} = \mathbb{G}_p \times \mathbb{G}_q \times \mathbb{G}_r$. Next, it computes g_p, g_q, and g_r as generators of $\mathbb{G}_p, \mathbb{G}_q$, and $\mathbb{G}_r$, respectively. It then chooses $R_{1,i}, R_{2,i} \in \mathbb{G}_r$ and $h_{1,i}, h_{2,i} \in \mathbb{G}_p$ uniformly at random for $i = 1$ to n, and $R_0 \in \mathbb{G}_r$ uniformly at random. The public parameters include $(N = pqr, \mathbb{G}, \mathbb{G}_T, \hat{e})$ along with:

$$PK = \left(g_p, \quad g_r, \quad Q = g_q \cdot R_0, \quad \{H_{1,i} = h_{1,i} \cdot R_{1,i}, \quad H_{2,i} = h_{2,i} \cdot R_{2,i}\}_{i=1}^n\right).$$

The master secret key SK is $\left(p, q, r, g_q, \{h_{1,i},\ h_{2,i}\}_{i=1}^n\right)$.

$\mathsf{Enc}_{PK}(\vec{x})$ Let $\vec{x} = (x_1, \ldots, x_n)$ with $x_i \in \mathbb{Z}_N$. This algorithm chooses random $s, \alpha, \beta \in \mathbb{Z}_N$ and $R_{3,i}, R_{4,i} \in \mathbb{G}_r$ for $i = 1$ to n. (Note: a random element $R \in \mathbb{G}_r$ can be sampled, even without knowing the factorization of N, by choosing random $\delta \in \mathbb{Z}_N$ and setting $R = g_r^\delta$.) It outputs the ciphertext

$$C = \left(C_0 = g_p^s, \quad \left\{C_{1,i} = H_{1,i}^s \cdot Q^{\alpha \cdot x_i} \cdot R_{3,i}, \quad C_{2,i} = H_{2,i}^s \cdot Q^{\beta \cdot x_i} \cdot R_{4,i}\right\}_{i=1}^n\right).$$

$\mathsf{GenKey}_{SK}(\vec{v})$ Let $\vec{v} = (v_1, \ldots, v_n)$, and recall $SK = \left(p, q, r, g_q, \{h_{1,i},\ h_{2,i}\}_{i=1}^n\right)$. This algorithm chooses random $r_{1,i}, r_{2,i} \in \mathbb{Z}_p$ for $i = 1$ to n, random $R_5 \in \mathbb{G}_r$, random $f_1, f_2 \in \mathbb{Z}_q$, and random $Q_6 \in \mathbb{G}_q$. It then outputs

$$SK_{\vec{v}} = \left(K = R_5 \cdot Q_6 \cdot \prod_{i=1}^n h_{1,i}^{-r_{1,i}} \cdot h_{2,i}^{-r_{2,i}}, \quad \left\{K_{1,i} = g_p^{r_{1,i}} \cdot g_q^{f_1 \cdot v_i}, \quad K_{2,i} = g_p^{r_{2,i}} \cdot g_q^{f_2 \cdot v_i}\right\}_{i=1}^n\right).$$

$\mathsf{Dec}_{SK_{\vec{v}}}(C)$ Let $C = \left(C_0, \{C_{1,i},\ C_{2,i}\}_{i=1}^n\right)$ and $SK_{\vec{v}} = \left(K, \{K_{1,i},\ K_{2,i}\}_{i=1}^n\right)$ be as above. The decryption algorithm outputs 1 iff

$$\hat{e}(C_0, K) \cdot \prod_{i=1}^n \hat{e}(C_{1,i}, K_{1,i}) \cdot \hat{e}(C_{2,i}, K_{2,i}) \stackrel{?}{=} 1.$$

Correctness. To see that correctness holds, let C and $SK_{\vec{v}}$ be as above. Then

$$
\begin{aligned}
&\hat{e}(C_0, K) \cdot \prod_{i=1}^{n} \hat{e}(C_{1,i}, K_{1,i}) \cdot \hat{e}(C_{2,i}, K_{2,i}) \\
&= \hat{e}\left(g_p^s, \quad R_5 Q_6 \prod_{i=1}^{n} h_{1,i}^{-r_{1,i}} h_{2,i}^{-r_{2,i}}\right) \\
&\quad \cdot \prod_{i=1}^{n} \hat{e}\left(H_{1,i}^s Q^{\alpha \cdot x_i} R_{3,i}, \quad g_p^{r_{1,i}} g_q^{f_1 \cdot v_i}\right) \cdot \hat{e}\left(H_{2,i}^s Q^{\beta \cdot x_i} R_{4,i}, \quad g_p^{r_{2,i}} g_q^{f_2 \cdot v_i}\right) \\
&= \hat{e}\left(g_p^s, \quad \prod_{i=1}^{n} h_{1,i}^{-r_{1,i}} h_{2,i}^{-r_{2,i}}\right) \cdot \prod_{i=1}^{n} \hat{e}\left(h_{1,i}^s \cdot g_q^{\alpha \cdot x_i}, \quad g_p^{r_{1,i}} g_q^{f_1 \cdot v_i}\right) \cdot \hat{e}\left(h_{2,i}^s \cdot g_q^{\beta \cdot x_i}, \quad g_p^{r_{2,i}} g_q^{f_2 \cdot v_i}\right) \\
&= \prod_{i=1}^{n} \hat{e}(g_q, g_q)^{(\alpha f_1 + \beta f_2) x_i v_i} \quad = \quad \hat{e}(g_q, g_q)^{(\alpha f_1 + \beta f_2 \bmod q) \cdot \langle \vec{x}, \vec{v} \rangle},
\end{aligned}
$$

where α, β are random in $\mathbb{Z}_N$ and f_1, f_2 are random in $\mathbb{Z}_q$. If $\langle \vec{x}, \vec{v} \rangle = 0 \bmod N$, then the above evaluates to 1. If $\langle \vec{x}, \vec{v} \rangle \neq 0 \bmod N$ there are two cases: if $\langle \vec{x}, \vec{v} \rangle \neq 0 \bmod q$ then with all but negligible probability (over choice of α, β, f_1, f_2) the above evaluates to an element other than the identity. The other possibility is that $\langle \vec{x}, \vec{v} \rangle = 0 \bmod q$, in which case the above would always evaluate to 1; however, this would reveal a non-trivial factor of N and so this occurs with only negligible probability (recall, our assumptions imply hardness of finding a non-trivial factor of N).

There may appear to be some redundancy in our construction; for instance, the $C_{1,i}$ and $C_{2,i}$ components play identical roles. In fact we can view the encryption scheme as consisting of two parallel sub-systems linked via the C_0 component (and the K component of the secret key). A natural question is whether this redundancy can be eliminated to achieve better performance. While such a construction appears to be secure, our current proof relies in an essential way on having these two parallel sub-systems.

4.3 Proof Intuition

The most challenging aspect to providing a proof of our scheme arises from the disjunctive capabilities of our system. In the previous attribute-hiding conjunctive scheme [12], security was proved via a sequence of hybrid games in which the "challenge ciphertext" associated with a vector $\vec{x}$ was changed component-by-component to a challenge ciphertext associated with a vector $\vec{y}$. The adversary in that case was only allowed to request secret keys that did not match either of $\vec{x}$ or $\vec{y}$, and so in every hybrid game it was the case that the adversary's secret keys would not "match" the challenge ciphertext. Thus, the hybrids could be handled in a relatively straightforward manner.

In our proof the adversary will again try to determine whether the challenge ciphertext is associated with either of two vectors $\vec{x}$ or $\vec{y}$. However, in our case the adversary can legally request a secret key $SK_{\vec{v}}$ that "matches" *both* $\vec{x}$ and $\vec{y}$, i.e., the adversary may request a secret key $SK_{\vec{v}}$ for which both $\langle \vec{x}, \vec{v} \rangle = 0$ and $\langle \vec{y}, \vec{v} \rangle = 0$. This means that we cannot use a naive sequence of hybrid games as outlined above. To see why, note that if we change one component at a time in the challenge ciphertext, then the hybrid vector used in an intermediate step will likely not "match" $SK_{\vec{v}}$ (i.e., will not be orthogonal to $\vec{v}$), and the adversary can detect this just by running the legal decryption procedure.

Therefore, we need to use a sequence of hybrid games in which an entire vector used in the challenge ciphertext is changed in one step, instead of using a sequence of hybrid games where the vector is changed component-by-component. To do this we take advantage of the fact that, as noted at the end of the previous section, our encryption scheme contains two parallel sub-systems. In our proof we will use hybrid games where a challenge ciphertext will be encrypted with respect to one vector in the first sub-system and a *different* vector in the second sub-system. (Note that such a ciphertext is ill-formed, since any honestly-generated ciphertext will always use the same vector in each sub-system.) Let $(\vec{a}, \vec{b})$ denote a ciphertext encrypted using vector $\vec{a}$ in the first sub-system and $\vec{b}$ in the second sub-system. To prove indistinguishability between the case when the challenge ciphertext is associated with $\vec{x}$ (which corresponds to $(\vec{x}, \vec{x})$) and the case when the challenge ciphertext is associated with $\vec{y}$ (which corresponds to $(\vec{y}, \vec{y})$), we use a sequence of intermediate hybrid games $(\vec{x}, \vec{0})$, $(\vec{x}, \vec{y})$, $(\vec{0}, \vec{y})$, showing indistinguishability in each case. That is, we show

$$(\vec{x}, \vec{x}) \approx (\vec{x}, \vec{0}) \approx (\vec{x}, \vec{y}) \approx (\vec{0}, \vec{y}) \approx (\vec{y}, \vec{y}),$$

proving our desired result. (We use the 0-vector since it is orthogonal to everything.) Using this structure in our proof allows us to use a simulator that will essentially work in one sub-system without "knowing" what is happening in the other one. The simulator embeds a "subgroup decision-like" assumption into the challenge ciphertext for each experiment. The structure of the challenge will determine whether a sub-system encrypts the given vector or the zero vector. Details of our proof and further discussion are given in the following section.

4.4 Proof of Security

This section is devoted to a proof of the following theorem:

Theorem 4.1. *If $\mathcal{G}$ satisfies Assumption 1 then the scheme described in Section 4 is an attribute-hiding, predicate-only encryption scheme.*

We use only Assumption 1 for proving security of our predicate-only scheme; we additionally rely on Assumption 2 when proving security of our full-fledged predicate encryption scheme in Appendix C.

For convenience, we include in Appendix B a re-statement of the definition of security for the particular inner-product predicate we use. Note that the particular predicate we use introduces a slight change in the definition, since the attribute space depends on the master public key (but, in Definition 2.2 the adversary is supposed to output I_0, I_1 before receiving the public key). We adapt the definition in the natural way by giving $\mathcal{A}$ the modulus N first, then requiring it to output I_0, I_1 before being given the rest of the public key. See Definition B.1 for the details.

We establish the theorem using a sequence of games, defined as follows:

Game_1: The challenge ciphertext is generated as a proper encryption using $\vec{x}$. (Recall from Definition B.1 that we let $\vec{x}, \vec{y}$ denote the two vectors output by the adversary.) That is, we choose random $s, \alpha, \beta \in \mathbb{Z}_N$ and random $\{R_{3,i}, R_{4,i}\} \in \mathbb{G}_r$ and compute the ciphertext as

$$C = \left(C_1 = g_p^s, \quad \left\{ C_{1,i} = H_{1,i}^s Q^{\alpha x_i} R_{3,i}, \quad C_{2,i} = H_{2,i}^s Q^{\beta x_i} R_{4,i} \right\}_{i=1}^n \right).$$

Game_2: We now generate the $\{C_{2,i}\}$ components as if encryption were done using $\vec{0}$. That is, we choose random $s, \alpha, \beta \in \mathbb{Z}_N$ and random $\{R_{3,i}, R_{4,i}\} \in \mathbb{G}_r$ and compute the ciphertext as

$$C = \left(C_1 = g_p^s, \quad \{ C_{1,i} = H_{1,i}^s Q^{\alpha x_i} R_{3,i}, \quad C_{2,i} = H_{2,i}^s R_{4,i} \}_{i=1}^n \right).$$

Game_3: We now generate the $\{C_{2,i}\}$ components using vector $\vec{y}$. That is, we choose random $s, \alpha, \beta \in \mathbb{Z}_N$ and random $\{R_{3,i}, R_{4,i}\} \in \mathbb{G}_r$ and compute the ciphertext as

$$C = \left(C_1 = g_p^s, \quad \left\{ C_{1,i} = H_{1,i}^s Q^{\alpha x_i} R_{3,i}, \quad C_{2,i} = H_{2,i}^s Q^{\beta y_i} R_{4,i} \right\}_{i=1}^n \right).$$

Game_4 and Game_5: These games are defined symmetrically to Game_2 and Game_3: In Game_4 the $\{C_{i,1}\}$ components are generated using $\vec{0}$. That is, we choose random $s, \alpha, \beta \in \mathbb{Z}_N$ and random $\{R_{3,i}, R_{4,i}\} \in \mathbb{G}_r$ and compute the ciphertext as

$$C = \left(C_1 = g_p^s, \quad \left\{ C_{1,i} = H_{1,i}^s R_{3,i}, \quad C_{2,i} = H_{2,i}^s Q^{\beta y_i} R_{4,i} \right\}_{i=1}^n \right).$$

In Game_5, the $\{C_{i,1}\}$ components are generated using $\vec{y}$. I.e., we choose random $s, \alpha, \beta \in \mathbb{Z}_N$ and random $\{R_{3,i}, R_{4,i}\} \in \mathbb{G}_r$ and compute the ciphertext as

$$C = \left(C_1 = g_p^s, \quad \left\{ C_{1,i} = H_{1,i}^s Q^{\alpha y_i} R_{3,i}, \quad C_{2,i} = H_{2,i}^s Q^{\beta y_i} R_{4,i} \right\}_{i=1}^n \right).$$

In Game_5 the challenge ciphertext is a proper encryption with respect to the vector $\vec{y}$. So, the proof of the theorem is concluded once we show that the adversary cannot distinguish between Game_i and Game_{i+1} for each i.

As discussed in Section 4.2, it is difficult to proceed directly from a game in which the challenge ciphertext is generated as a proper encryption using $\vec{x}$, to a game in which the challenge ciphertext is generated as a proper encryption using $\vec{y}$. (Indeed, this is the reason our construction uses two "sub-systems" to begin with.) That is why our proof proceeds via the intermediate Game_3 where half the challenge ciphertext corresponds to an encryption using $\vec{x}$ and the other half corresponds to an encryption using $\vec{y}$. Intermediate games Game_2 and Game_4 are used to simplify the proof; it helps when part of the ciphertext corresponds to an encryption using $\vec{0}$ since this vector is orthogonal to everything.

The main difficulty in our proofs will be to answer queries for decryption keys. In considering the indistinguishability of Game_1 and Game_2 (and, symmetrically, Game_4 and Game_5), we will actually be able to construct *all* decryption keys (i.e., even keys that would allow the adversary to distinguish an encryption relative to $\vec{x}$ from an encryption relative to $\vec{y}$). In essence, we will be showing that even such keys cannot be used to distinguish a well-formed encryption of $\vec{x}$ (or $\vec{y}$) from a badly-formed one.

On the other hand, in considering the indistinguishability of Game_2 and Game_3 (and, symmetrically, Game_3 and Game_4) we will not be able to construct all decryption keys. Instead, we will deal separately with the problems of (1) providing keys for vectors $\vec{v}$ with $\langle \vec{v}, \vec{x} \rangle = 0 = \langle \vec{v}, \vec{y} \rangle$ and (2) providing keys for vectors $\vec{v}$ with $\langle \vec{v}, \vec{x} \rangle \neq 0 \neq \langle \vec{v}, \vec{y} \rangle$.

4.4.1 Indistinguishability of Game_1 and Game_2

Fix an adversary $\mathcal{A}$. We describe a simulator who is given $(N = pqr, \mathbb{G}, \mathbb{G}_T, \hat{e})$ along with the elements g_p, g_r, $g_q R_1$, $h_p = g_p^b$, $k_p = g_p^{b^2}$, $g_p^a g_q$, $g_p^{ab} Q_1$, g_p^s, $g_p^{bs} Q_2 R_2$, and an element $T = g_p^{b^2 s} g_q^{\beta} g_r^{R_3}$ where β is either 0 or uniform in $\mathbb{Z}_q$ (cf. Assumption 1).

Before describing the simulation in detail, we observe that the simulator can sample a random element $R \in \mathbb{G}_r$ by choosing random $\delta \in \mathbb{Z}_N$ and setting $R = g_r^{\delta}$. Although there is no direct wat for the simulator to sample a random element of $\mathbb{G}_q$ (since g_q is not provided to the simulator), it is possible for the simulator to choose an independent random element $QR \in \mathbb{G}_{qr} \stackrel{\text{def}}{=} \mathbb{G}_q \times \mathbb{G}_r$ by choosing random $\delta_1, \delta_2 \in \mathbb{Z}_N$ and setting $QR = (g_q R_1)^{\delta_1} \cdot g_r^{\delta_2}$. Henceforth, we simply describe the simulator as sampling uniformly from $\mathbb{G}_r$ and $\mathbb{G}_{qr}$ with the understanding that such sampling is done in this way.

Public parameters. The simulator begins by giving N to $\mathcal{A}$, who outputs vectors $\vec{x}, \vec{y}$. The simulator chooses random $\{w_{1,i}, w_{2,i}\} \in \mathbb{Z}_N$ and random $\{R_{1,i}, R_{2,i}\} \in \mathbb{G}_r$, includes $(N, \mathbb{G}, \mathbb{G}_T, \hat{e})$ in the public parameters, and sets the remaining values as follows:

$$PK = \left(g_p,\ g_r,\ g_q R_1,\ \left\{H_{1,i} = (h_p)^{x_i} g_p^{w_{1,i}} R_{1,i}, \quad H_{2,i} = (k_p)^{x_i} g_p^{w_{2,i}} R_{2,i}\right\}\right).$$

By doing so, the simulator is implicitly setting $h_{1,i} = h_p^{x_i} g_p^{w_{1,i}}$ and $h_{2,i} = k_p^{x_i} g_p^{w_{2,i}}$. Note that PK has the appropriate distribution.

Key derivation. We now describe how the simulator prepares the secret key corresponding to the vector $\vec{v} = (v_1, \ldots, v_n)$. We stress that although Definition B.1 restricts the vectors $\vec{v}$ for which the adversary is allowed to request secret keys, we do not rely on this restriction here. This is because the purpose of this hybrid proof is to show that the adversary cannot distinguish between properly formed encryptions of $\vec{x}$ and improperly formed encryptions (that are a combination of an encryption of $\vec{x}$ and an encryption of $\vec{0}$).

We begin with some intuition. We must construct the $K_{1,i}$ and $K_{2,i}$ components of the key. Note that we do not have access to g_q, but we do have $g_q g_p^a$. We will use this element here. This will give rise to terms containing a in the exponent of g_p. Note, however, that we will later have to construct the K component of the key, whose purpose is to cancel out terms in the G_p subgroup. If $\langle \vec{v}, \vec{x} \rangle \neq 0$, then additional terms involving ab and ab^2 will have to appear in K. However, we do not have access to $g_p^{ab^2}$; indeed if we did, the assumption would be false and we could easily distinguish between Game_1 and Game_2. We deal with this problem by adding a term (using the $g_p^{ab} g_q^d$ term given in the assumption) to the $K_{1,i}$ components that will allow us to cancel out the ab^2 terms that will appear in K due to the $K_{2,i}$ components.

The simulator begins by choosing random $f_1', f_2', \{r_{1,i}'\}, \{r_{2,i}'\} \in \mathbb{Z}_N$. In constructing the key, implicitly the simulator will be setting:

$$\begin{aligned} r_{1,i} &= r_{1,i}' + v_i \cdot (a f_1' - ab f_2') && (1) \\ r_{2,i} &= r_{2,i}' + a\, f_2'\, v_i, && (2) \end{aligned}$$

as well as $f_1 = f_1' - d\, f_2'$ and $f_2 = f_2'$, where we set $d = \log_{g_q} Q_1$. Note that these values are each independently and uniformly distributed in $\mathbb{Z}_N$, just as they would be in actual secret key components.

Next, for all i the simulator computes:

$$\begin{aligned} K_{1,i} &= \left(g_p^a g_q\right)^{f_1' v_i} \cdot \left(g_p^{ab} Q_1\right)^{-f_2' v_i} \cdot g_p^{r_{1,i}'} \\ &= g_p^{(a f_1' - ab f_2') \cdot v_i + r_{1,i}'} \cdot g_q^{(f_1' - d f_2') \cdot v_i} \end{aligned}$$

and

$$\begin{aligned} K_{2,i} &= \left(g_p^a g_q\right)^{f_2' v_i} \cdot g_p^{r_{2,i}'} \\ &= g_p^{a f_2' v_i + r_{2,i}'} \cdot g_q^{f_2' v_i}. \end{aligned}$$

The simulator will next construct the K element of the decryption key. Recall that $h_{1,i} = (g_p)^{bx_i} g_p^{w_{1,i}}$. Therefore, the exponents in K will contain a term of the form $\sum_i r_{1,i} bx_i$. But because of how we chose $r_{1,i}$, we have $\sum_i r_{1,i} bx_i = k(abf'_1 - ab^2 f_2) + \sum_i r'_{1,i} x_i$ where $k = \langle \vec{v}, \vec{x} \rangle$. A similar equation holds for the terms arising from the $h_{2,i}$ parts of K, and allows the simulator to cancel out all the ab^2 terms that arise in K.

The simulator computes K as follows: Let $k = \langle \vec{v}, \vec{x} \rangle$. The simulator then chooses random $QR \in \mathbb{G}_{qr}$ and computes

$$\begin{aligned} K &= QR \cdot \left(g_p^{ab} Q_1\right)^{-k \cdot f'_1} \\ &\quad \cdot \prod_i \left(g_p^a g_q\right)^{-f'_1 v_i w_{1,i} - f'_2 v_i w_{2,i}} \cdot \left(g_p^{ab} Q_1\right)^{f'_2 v_i w_{1,i}} \cdot g_p^{-w_{1,i} \cdot r'_{1,i} - w_{2,i} \cdot r'_{2,i}} \cdot h_p^{-x_i \cdot r'_{1,i}} \cdot k_p^{-x_i \cdot r'_{2,i}}. \end{aligned}$$

The simulator then hands the adversary $SK_{\vec{v}} = (K, \{K_{1,i}, K_{2,i}\}_{i=1}^n)$ as the key.

To see formally that the K component has the correct distribution, let K_p, K_q, and K_r denote the projections of K in $\mathbb{G}_p, \mathbb{G}_q$, and $\mathbb{G}_r$, respectively. It is easy to see that K_q and K_r are independently and uniformly distributed, as required. Furthermore,

$$\begin{aligned} K_p &= g_p^{-abkf'_1} \cdot \prod_i g_p^{-af'_1 v_i w_{1,i} - af'_2 v_i w_{2,i}} g_p^{abf'_2 v_i w_{1,i}} g_p^{-w_{1,i} r'_{1,i} - w_{2,i} r'_{2,i}} h_p^{-x_i r'_{1,i}} k_p^{-x_i r'_{2,i}} \\ &= h_p^{-akf'_1} \prod_i \left(h_p^{-x_i r'_{1,i}} g_p^{-w_{1,i} r'_{1,i}} g_p^{-w_{1,i} v_i (af'_1 - abf'_2)} \right) \cdot \left(k_p^{-x_i r'_{2,i}} g_p^{-w_{2,i} r'_{2,i}} g_p^{-w_{2,i} af'_2 v_i} \right) \\ &= \prod_i h_p^{-ax_i v_i f'_1} \cdot \left(h_p^{-x_i r'_{1,i}} g_p^{-w_{1,i} r'_{1,i}} g_p^{-w_{1,i} v_i (af'_1 - abf'_2)} \right) \cdot \left(h_p^{abx_i v_i f'_2} \cdot h_p^{-abx_i v_i f'_2} \right) \\ &\qquad \cdot \left(k_p^{-x_i r'_{2,i}} g_p^{-w_{2,i} r'_{2,i}} g_p^{-w_{2,i} af'_2 v_i} \right), \end{aligned}$$

using the fact that $k = \langle \vec{x}, \vec{v} \rangle = \sum_i x_i, v_i$. Using simple (but tedious) algebra, we obtain

$$\begin{aligned} &K_p \\ &= \prod_i \left(h_p^{-x_i r'_{1,i}} g_p^{-w_{1,i} r'_{1,i}} h_p^{-x_i v_i \cdot (af'_1 - abf'_2)} g_p^{-w_{1,i} v_i (af'_1 - abf'_2)} \right) \cdot \left(k_p^{-x_i r'_{2,i}} g_p^{-w_{2,i} r'_{2,i}} k_p^{-x_i af'_2 v_i} g_p^{-w_{2,i} af'_2 v_i} \right) \\ &= \prod_i \left(h_p^{x_i} g_p^{w_{1,i}}\right)^{-r_{1,i}} \left(k_p^{x_i} g_p^{w_{2,i}}\right)^{-r_{2,i}} \quad = \quad \prod_i h_{1,i}^{-r_{1,i}} h_{2,i}^{-r_{2,i}} \end{aligned}$$

(using Eqs. (1) and (2)), and thus K_p has the correct distribution. We conclude that K has the correct distribution.

The challenge ciphertext. The challenge ciphertext is generated in a straightforward way, as follows. The simulator chooses $\{R_{7,i}, R_{8,i}\} \in \mathbb{G}_r$ at random, sets C_1 equal to g_p^s, and computes:

$$\begin{aligned} C_{1,i} &= \left(g_p^{bs} Q_2 R_2\right)^{x_i} \cdot (g_p^s)^{w_{1,i}} \cdot R_{7,i} \\ &= h_p^{x_i s} g_p^{w_{1,i} s} Q_2^{x_i} R'_{7,i} \\ &= (h_{1,i})^s Q_2^{x_i} R'_{7,i} \\ C_{2,i} &= T^{x_i} \cdot (g_p^s)^{w_{2,i}} \cdot R_{8,i} \\ &= (h_{2,i})^s \left(g_q^{\beta}\right)^{x_i} R'_{8,i}, \end{aligned}$$

where $\{R'_{7,i}, R'_{8,i}\}$ are random elements of $\mathbb{G}_r$ whose exact values are unimportant.

Analysis. By examining the projections of the components of the challenge ciphertext in the groups $\mathbb{G}_p$, $\mathbb{G}_q$, and $\mathbb{G}_r$, it can be verified that when β is random the challenge ciphertext is distributed exactly as in Game_1, whereas if $\beta = 0$ the challenge ciphertext is distributed exactly as in Game_2. We conclude that, under Assumption 1, these two games are indistinguishable. ∎

4.4.2 Indistinguishability of Game_2 and Game_3

Fix again some adversary $\mathcal{A}$. We describe a simulator who is given $(N = pqr, \mathbb{G}, \mathbb{G}_T, \hat{e})$ along with the elements g_p, g_r, $g_q R_1$, $h_p = g_p^b$, $k_p = g_p^{b^2}$, $g_p^a g_q$, $g_p^{ab} Q_1$, g_p^s, $g_p^{bs} Q_2 R_2$, and an element $T = g_p^{b^2 s} g_q^{\beta} g_r^{R_3}$ where β is either 0 or uniform in $\mathbb{Z}_q$. Recall that sampling uniform elements from $\mathbb{G}_r$ and $\mathbb{G}_{qr}$ can be done efficiently. The simulator interacts with $\mathcal{A}$ as we now describe.

Public parameters. The simulator begins by giving N to $\mathcal{A}$, who outputs vectors $\vec{x}, \vec{y}$. The simulator chooses random $\{w_{1,i}, w_{2,i}\} \in \mathbb{Z}_N$ and random $\{R_{1,i}, R_{2,i}\} \in \mathbb{G}_r$, includes $(N, \mathbb{G}, \mathbb{G}_T, \hat{e})$ in the public parameters, and sets the public parameters as follows:

$$PK = \left(g_p,\ g_r,\ g_q R_1,\ \left\{H_{1,i} = (h_p)^{x_i} g_p^{w_{1,i}} R_{1,i} \quad H_{2,i} = (k_p)^{y_i} g_p^{w_{2,i}} R_{2,i}\right\}\right).$$

By doing so, the simulator is implicitly setting $h_{1,i} = h_p^{x_i} g_p^{w_{1,i}}$ and $h_{2,i} = k_p^{y_i} g_p^{w_{2,i}}$. Note that PK has the appropriate distribution.

Key derivation. The adversary $\mathcal{A}$ may request secret keys corresponding to different vectors, and we now describe how the simulator prepares the secret key corresponding to the vector $\vec{v} = (v_1, \ldots, v_n)$. Here, the simulator will only be able to produce the appropriate secret key when the vector $\vec{v}$ satisfies the restriction imposed by Definition B.1. We distinguish two cases, depending on whether $\langle \vec{v}, \vec{x} \rangle$ and $\langle \vec{v}, \vec{y} \rangle$ are both zero or whether they are both non-zero.

Case 1. We first consider the case where $\langle \vec{v}, \vec{x} \rangle = 0 = \langle \vec{v}, \vec{y} \rangle$. The simulator begins by choosing random $f_1, f_2, \{r'_{1,1}\}, \{r'_{2,1}\} \in \mathbb{Z}_N$. Then for all i it computes:

$$\begin{aligned} K_{1,i} &= \left(g_p^a g_q\right)^{f_1 v_i} \cdot (g_p)^{r'_{1,i}} \\ &= g_p^{a f_1 v_i + r'_{1,i}} \cdot g_q^{f_1 v_i} \\ K_{2,i} &= \left(g_p^a g_q\right)^{f_2 v_i} \cdot (g_p)^{r'_{2,i}} \\ &= g_p^{a f_2 v_i + r'_{2,i}} \cdot g_q^{f_2 v_i}. \end{aligned}$$

Finally, the simulator chooses random $QR \in \mathbb{G}_{qr}$ and computes

$$K = QR \cdot \prod_i \left(g_p^a g_q\right)^{-f_1 v_i w_{1,i} - f_2 v_i w_{2,i}} \cdot g_p^{-w_{1,i} \cdot r'_{1,i} - w_{2,i} \cdot r'_{2,i}} \cdot h_p^{-x_i \cdot r'_{1,i}} \cdot k_p^{-y_i \cdot r'_{2,i}}.$$

The simulator then hands the adversary $SK_{\vec{v}} = (K, \{K_{1,i}, K_{2,i}\})$ as the key.

To see that this key has the correct distribution, note that by construction of the $\{K_{1,i}, K_{2,i}\}$ the values f_1, f_2 are random; furthermore, the simulator implicity sets

$$\begin{aligned} r_{1,i} &= r'_{1,i} + a f_1 v_i \\ r_{2,i} &= r'_{2,i} + a f_2 v_i, \end{aligned}$$

which are uniformly distributed as well. Looking at K_p, the projection of K in $\mathbb{G}_p$ (as in the proof in the previous section), we see that

$$\begin{aligned} K_p &= \prod_i g_p^{-af_1 v_i w_{1,i} - af_2 v_i w_{2,i}} \cdot g_p^{-w_{1,i}\cdot r'_{1,i} - w_{2,i}\cdot r'_{2,i}} \cdot h_p^{-x_i \cdot r'_{1,i}} \cdot k_p^{-y_i\cdot r'_{2,i}} \\ &= \prod_i h_p^{-af_1 x_i v_i} \cdot k_p^{-af_2 y_i v_i} \cdot g_p^{-af_1 v_i w_{1,i} - af_2 v_i w_{2,i}} \cdot g_p^{-w_{1,i}\cdot r'_{1,i} - w_{2,i}\cdot r'_{2,i}} \cdot h_p^{-x_i\cdot r'_{1,i}} \cdot k_p^{-y_i \cdot r'_{2,i}}, \end{aligned}$$

using the fact that $\prod_i h_p^{-af_1 x_i v_i} = h_p^{-af_1\cdot \sum_i x_i v_i} = 1 = \prod_i k_p^{-af_2 y_i v_i}$ (because $\langle \vec{v}, \vec{x} \rangle = 0 = \langle \vec{v}, \vec{y} \rangle$). Algebraic manipulation as in the previous section shows that K_p has the correct distribution.

Case 2. Here, we consider the case where $\langle \vec{v}, \vec{x}\rangle = c_x \neq 0$ and $\langle \vec{v}, \vec{y} \rangle = c_y \neq 0$. The simulator begins by choosing random $f'_1, f'_2, \{r'_{1,1}\}, \{r'_{2,1}\} \in \mathbb{Z}_N$. Next, for all i it computes

$$\begin{aligned} K_{1,i} &= \left(g_p^a g_q\right)^{f'_1 v_i} \left(g_p^{ab} Q_1\right)^{-c_y\cdot f'_2 v_i} \cdot (g_p)^{r'_{1,i}} \\ &= g_p^{(af'_1 - abc_y f'_2)\cdot v_i + r'_{1,i}} \cdot g_q^{(f'_1 - c_y d f'_2)\cdot v_i} \\ K_{2,i} &= \left(g_p^a g_q\right)^{c_x \cdot f'_2 v_i} \cdot (g_p)^{r'_{2,i}} \\ &= g_p^{ac_x f'_2 v_i + r'_{2,i}} \cdot g_q^{c_x\cdot f'_2 v_i}, \end{aligned}$$

where we set $d = \log_{g_q} Q_1$ as in the previous proof. Finally, the simulator chooses random $QR \in \mathbb{G}_{qr}$ and computes

$$\begin{aligned} K &= QR \cdot (g_p^{ab} Q_1)^{-c_x f'_1} \\ &\quad \cdot \prod_i \left(g_p^a g_q\right)^{-f'_1 v_i w_{1,i} - f'_2 c_x v_i w_{2,i}} \cdot (g_p^{ab} Q_1)^{f'_2 c_y v_i w_{1,i}} \cdot g_p^{-w_{1,i}\cdot r'_{1,i} - w_{2,i}\cdot r'_{2,i}} \cdot h_p^{-x_i \cdot r'_{1,i}} \cdot k_p^{-y_i \cdot r'_{2,i}}. \end{aligned}$$

The simulator then hands the key $SK_{\vec{v}} = (K, \{K_{1,i}, K_{2,i}\})$ to the adversary.

To see that this key has the correct distribution, note that by construction of the $\{K_{1,i}, K_{2,i}\}$ the simulator implicity sets

$$\begin{aligned} r_{1,i} &= r'_{1,i} + (af'_1 - c_y ab f'_2)\cdot v_i \\ r_{2,i} &= r'_{2,i} + ac_x f'_2 v_i, \end{aligned}$$

as well as $f_1 = f'_1 - c_y \cdot d f'_2$ and $f_2 = c_x \cdot f'_2$. It is clear that f_1 and the $\{r_{1,i}, r_{2,i}\}$ are independently and uniformly distributed in $\mathbb{Z}_N$. The value f_2 is also uniformly distributed in $\mathbb{Z}_N$ as long as $\gcd(c_x, N) = 1$. (If $\gcd(c_x, N) \neq 1$, then the adversary has found a non-trivial factor of N. This occurs with negligible probability under Assumption 1.)

As for element K of the secret key, it is once again easy to see that the projection of K in $\mathbb{G}_{qr}$ is uniformly distributed. Looking at K_p, the projection of K in $\mathbb{G}_p$ (as in the previous section), we see that

$$
\begin{aligned}
K_p &= g_p^{-abc_x f_1'} \cdot \prod_i g_p^{-af_1' v_i w_{1,i} - af_2' c_x v_i w_{2,i}} \cdot g_p^{abf_2' c_y v_i w_{1,i}} \cdot g_p^{-w_{1,i} \cdot r_{1,i}' - w_{2,i} \cdot r_{2,i}'} \cdot h_p^{-x_i \cdot r_{1,i}'} \cdot k_p^{-y_i \cdot r_{2,i}'} \\
&= \prod_i h_p^{-ax_i v_i f_1'} \cdot g_p^{-af_1' v_i w_{1,i} - af_2' c_x v_i w_{2,i}} \cdot g_p^{abf_2' c_y v_i w_{1,i}} \cdot (h_{1,i})^{r_{1,i}'} (h_{2,i})^{r_{2,i}'} \\
&= h_p^{c_x c_y abf_2'} \cdot h_p^{-c_x c_y abf_2'} \prod_i g_p^{-af_2' c_x v_i w_{2,i}} \cdot g_p^{abf_2' c_y v_i w_{1,i}} \cdot (h_{1,i})^{-r_{1,i}' - av_i f_1'} (h_{2,i})^{-r_{2,i}'} \\
&= \prod_i h_p^{x_i v_i c_y abf_2'} \cdot k_p^{-c_x y_i v_i af_2'} \cdot g_p^{-af_2' c_x v_i w_{2,i}} \cdot g_p^{abf_2' c_y v_i w_{1,i}} \cdot (h_{1,i})^{-r_{1,i}' - av_i f_1'} (h_{2,i})^{-r_{2,i}'} \\
&= \prod_i (h_{1,i})^{-r_{1,i}' - av_i f_1' + abf_2' c_y v_i} (h_{2,i})^{-r_{2,i}' - ac_x v_i f_2'} \quad = \quad \prod_i (h_{1,i})^{-r_{1,i}} (h_{2,i})^{-r_{2,i}},
\end{aligned}
$$

and so K_p has the right distribution. We conclude that K has the correct distribution.

The challenge ciphertext. The challenge ciphertext is generated in a straightforward way. The simulator chooses $\{R_{7,i}, R_{8,i}\} \in \mathbb{G}_r$ at random, sets $C_1 = g_p^s$, and computes:

$$
\begin{aligned}
C_{1,i} &= \left(g_p^{bs} Q_2 R_2\right)^{x_i} \cdot (g_p^s)^{w_{1,i}} \cdot R_{7,i} \\
&= (h_{1,i})^s Q_2^{x_i} R_{7,i}' \\
C_{2,i} &= T^{y_i} (g_p^s)^{w_{2,i}} R_{8,i} \\
&= (h_{2,i})^s \left(g_q^\beta\right)^{y_i} R_{8,i}',
\end{aligned}
$$

where $\{R_{7,i}', R_{8,i}'\}$ again refer to random elements of $\mathbb{G}_r$ whose exact values are unimportant.

Analysis. By examining the projections of the components of the challenge ciphertext in the groups $\mathbb{G}_p$, $\mathbb{G}_q$, and $\mathbb{G}_r$, it can be verified that when β is random the challenge ciphertext is distributed exactly as in Game_3, whereas if $\beta = 0$ the challenge ciphertext is distributed exactly as in Game_2. We conclude that, under Assumption 1, these two games are indistinguishable. ■

4.4.3 Completing the Proof

Our scheme is symmetric with respect to the roles of $h_{1,i}$ and $h_{2,i}$. Thus, the proof that Game_3 and Game_4 are indistinguishable exactly parallels the proof (given in the previous section) that Game_2 and Game_3 are indistinguishable, while the proof that Game_4 and Game_5 are indistinguishable exactly parallels the proof (given in Section 4.4.1) that Game_1 and Game_2 are indistinguishable. This concludes the proof of our theorem.

5 Applications of Our Main Construction

In this section we discuss some applications of predicate encryption schemes of the type constructed in this paper. Our treatment here is general and can be based on any predicate encryption scheme supporting "inner product" queries; we do not rely on any specific details of our construction.

Given a vector $\vec{x} \in \mathbb{Z}_N^\ell$, we denote by $f_{\vec{x}} : \mathbb{Z}_N^\ell \to \{0,1\}$ the function such that $f_{\vec{x}}(\vec{y}) = 1$ iff $\langle \vec{x}, \vec{y} \rangle = 0 \bmod N$. We define $\mathcal{F}_\ell \stackrel{\text{def}}{=} \{f_{\vec{x}} \mid \vec{x} \in \mathbb{Z}_N^\ell\}$. An *inner product encryption scheme of dimension* ℓ is an attribute-hiding predicate encryption scheme for the class of predicates $\mathcal{F}_\ell$.

5.1 Anonymous Identity-Based Encryption

As a warm-up, we show how anonymous identity-based encryption (IBE) can be recovered from any inner product encryption scheme of dimension 2. To generate the master public and secret keys for the IBE scheme, simply run the setup algorithm of the underlying inner product encryption scheme. To generate secret keys for the identity $I \in \mathbb{Z}_N$, set $\vec{I} := (1, I)$ and output the secret key for the predicate $f_{\vec{I}}$. To encrypt a message M for the identity $J \in \mathbb{Z}_N$, set $\vec{J} := (-J, 1)$ and encrypt the message using the encryption algorithm of the underlying inner product encryption scheme and the attribute $\vec{J}$. Since $\left\langle \vec{I}, \vec{J} \right\rangle = 0$ iff $I = J$, correctness and security follow.

5.2 Hidden-Vector Encryption

Given a set Σ, let $\Sigma_\star = \Sigma \cup \{\star\}$. Hidden-vector encryption (HVE) [12] corresponds to a predicate encryption scheme for the class of predicates $\Phi_\ell^{\mathsf{hve}} = \{\phi^{\mathsf{hve}}_{(a_1,\ldots,a_\ell)} \mid a_1, \ldots, a_\ell \in \Sigma_\star\}$, where

$$\phi^{\mathsf{hve}}_{(a_1,\ldots,a_\ell)}(x_1, \ldots, x_\ell) = \begin{cases} 1 & \text{if, for all } i, \text{ either } a_i = x_i \text{ or } a_i = \star \\ 0 & \text{otherwise} \end{cases} .$$

A generalization of the ideas from the previous section can be used to realize hidden-vector encryption with $\Sigma = \mathbb{Z}_N$ from any inner product encryption scheme $(\mathsf{Setup}, \mathsf{GenKey}, \mathsf{Enc}, \mathsf{Dec})$ of dimension 2ℓ:

- The setup algorithm is unchanged.
- To generate a secret key corresponding to the predicate $\phi^{\mathsf{hve}}_{(a_1,\ldots,a_\ell)}$, first construct a vector $\vec{A} = (A_1, \ldots, A_{2\ell})$ as follows:

$$\begin{array}{lll} \text{if } a_i \neq \star: & A_{2i-1} := 1, & A_{2i} := a_i \\ \text{if } a_i = \star: & A_{2i-1} := 0, & A_{2i} := 0. \end{array}$$

  Then output the key obtained by running $\mathsf{GenKey}_{SK}(f_{\vec{A}})$.
- To encrypt a message M for the attribute $x = (x_1, \ldots, x_\ell)$, choose random $r_1, \ldots, r_\ell \in \mathbb{Z}_N$ and construct a vector $\vec{X}_{\vec{r}} = (X_1, \ldots, X_{2\ell})$ as follows:

$$X_{2i-1} := -r_i \cdot x_i, \quad X_{2i} := r_i$$

  (multiplication is done modulo N). Then output the ciphertext $C \leftarrow \mathsf{Enc}_{PK}(\vec{X}_{\vec{r}}, M)$.

To see that correctness holds, let $(a_1, \ldots, a_\ell)$, $\vec{A}$, $(x_1, \ldots, x_\ell)$, $\vec{r}$, and $\vec{X}_{\vec{r}}$ be as above. Then:

$$\phi^{\mathsf{hve}}_{(a_1,\ldots,a_\ell)}(x_1, \ldots, x_\ell) = 1 \;\Rightarrow\; \forall \vec{r}: \left\langle \vec{A}, \vec{X}_{\vec{r}} \right\rangle = 0 \;\Rightarrow\; \forall \vec{r}: f_{\vec{A}}(\vec{X}_{\vec{r}}) = 1.$$

Furthermore, assuming $\gcd(a_i - x_i, N) = 1$ for all i:

$$\phi^{\mathsf{hve}}_{(a_1,\ldots,a_\ell)}(x_1, \ldots, x_\ell) = 0 \;\Rightarrow\; \Pr_{\vec{r}}\left[\left\langle \vec{A}, \vec{X}_{\vec{r}} \right\rangle = 0\right] = 1/N \;\Rightarrow\; \Pr_{\vec{r}}\left[f_{\vec{A}}(\vec{X}_{\vec{r}}) = 1\right] = 1/N,$$

which is negligible. Using this, one can prove security of the construction as well.

A straightforward modification of the above gives a scheme that is the "dual" of HVE, where the set of attributes is $(\Sigma_\star)^\ell$ and the class of predicates is $\bar{\Phi}_\ell^{\mathsf{hve}} = \{\bar{\phi}^{\mathsf{hve}}_{(a_1,\ldots,a_\ell)} \mid a_1, \ldots, a_\ell \in \Sigma\}$ with

$$\bar{\phi}^{\mathsf{hve}}_{(a_1,\ldots,a_\ell)}(x_1, \ldots, x_\ell) = \begin{cases} 1 & \text{if, for all } i, \text{ either } a_i = x_i \text{ or } x_i = \star \\ 0 & \text{otherwise} \end{cases} .$$

5.3 Predicate Encryption Schemes Supporting Polynomial Evaluation

We can also construct predicate encryption schemes for classes of predicates corresponding to polynomial evaluation. Let $\Phi_{\leq d}^{\text{poly}} = \{f_p \mid p \in \mathbb{Z}_N[x], \deg(p) \leq d\}$, where

$$f_p(x) = \begin{cases} 1 & \text{if } p(x) = 0 \bmod N \\ 0 & \text{otherwise} \end{cases}$$

for $x \in \mathbb{Z}_N$. Given an inner product encryption scheme ($\mathsf{Setup}, \mathsf{GenKey}, \mathsf{Enc}, \mathsf{Dec}$) of dimension $d+1$, we can construct a predicate encryption scheme for $\Phi_{\leq d}^{\text{poly}}$ as follows:

- The setup algorithm is unchanged.
- To generate a secret key corresponding to the polynomial $p = a_d x^d + \cdots + a_0 x^0$, set $\vec{p} := (a_d, \ldots, a_0)$ and output the key obtained by running $\mathsf{GenKey}_{SK}(f_{\vec{p}})$.
- To encrypt a message M for the attribute $w \in \mathbb{Z}_N$, set $\vec{w} := (w^d \bmod N, \ldots, w^0 \bmod N)$ and output the ciphertext $C \leftarrow \mathsf{Enc}_{PK}(\vec{w}, M)$.

Since $p(w) = 0$ iff $\langle \vec{p}, \vec{w} \rangle = 0$, correctness and security follow.

The above shows that we can construct predicate encryption schemes where predicates correspond to univariate polynomials whose degree d is polynomial in the security parameter. This can be generalized to the case of polynomials in t variables, and degree at most d in each variable, as long as d^t is polynomial in the security parameter.

We can also construct schemes that are the "dual" of the above, in which attributes correspond to polynomials and predicates involve the evaluation of the input polynomial at some fixed point.

5.4 Disjunctions, Conjunctions, and Evaluating CNF and DNF Formulas

Given the polynomial-based constructions of the previous section, we can fairly easily build predicate encryption schemes for disjunctions of equality tests. For example, the predicate OR_{I_1,I_2}, where $\mathsf{OR}_{I_1,I_2}(x) = 1$ iff either $x = I_1$ or $x = I_2$, can be encoded as the univariate polynomial

$$p(x) = (x - I_1) \cdot (x - I_2),$$

which evaluates to 0 iff the relevant predicate evaluates to 1. Similarly, the predicate $\overline{\mathsf{OR}}_{a_1,a_2}$, where $\overline{\mathsf{OR}}_{a_1,a_2}(x_1, x_2) = 1$ iff either $x_1 = I_1$ or $x_2 = I_2$, can be encoded as the bivariate polynomial

$$p'(x_1, x_2) = (x_1 - I_1) \cdot (x_2 - I_2).$$

Conjunctions can be handled in a similar fashion. Consider, for example, the predicate AND_{I_1,I_2} where $\mathsf{AND}_{I_1,I_2}(x_1, x_1) = 1$ if both $x_1 = I_1$ and $x_2 = I_2$. Here, we determine the relevant secret key by choosing a random $r \in \mathbb{Z}_N$ and letting the secret key correspond to the polynomial

$$p''(x_1, x_2) = r \cdot (x_1 - I_1) + (x_2 - I_2).$$

Note that if $\mathsf{AND}_{I_1,I_2}(x_1, x_1) = 1$ then $p''(x_1, x_2) = 0$, whereas if $\mathsf{AND}_{I_1,I_2}(x_1, x_1) = 0$ then, with all but negligible probability over choice of r, it will hold[2] that $p''(x_1, x_2) \neq 0$.

[2] In general, the secret key may leak the value of r in which case the adversary will be able to find I_1', I_2' such that $\mathsf{AND}_{I_1,I_2}(I_1', I_2') \neq 1$ yet $p''(I_1', I_2') = 0$. However, this is not a problem when considering the "selective" notion of security where the adversary must commit to I_1', I_2' at the outset of the experiment.

The above ideas extend to more complex combinations of disjunctions and conjunctions, and for boolean variables this means we can handle arbitrary CNF or DNF formulas. (For non-boolean variables we do not know how to directly handle negation.) As pointed out in the previous section, the complexity of the resulting scheme depends polynomially on d^t, where t is the number of variables and d is the maximum degree (of the resulting polynomial) in each variable.

5.5 Exact Thresholds

We conclude with an application that relies directly on inner product encryption. Here, we consider the setting of "fuzzy IBE" [23], which can be mapped to the predicate encryption framework as follows: fix a set $A = \{1, \ldots, \ell\}$ and let the set of attributes be all subsets of A. Predicates take the form $\Phi = \{\phi_S \mid S \subseteq A\}$ where $\phi_S(S') = 1$ iff $|S \cap S'| \geq t$, i.e., S and S' overlap in *at least* t positions. Sahai and Waters [23] show a construction of a payload-hiding predicate encryption scheme for this class of predicates.

We can construct a scheme where the attribute space is the same as before, but the class of predicates corresponds to overlap in *exactly* t positions. (Our scheme will also be attribute hiding.) Namely, set $\Phi' = \{\phi'_S \mid S \subseteq A\}$ with $\phi'_S(S') = 1$ iff $|S \cap S'| = t$. Then, given any inner product encryption scheme of dimension $\ell + 1$, we construct a scheme as follows:

- The setup algorithm is unchanged.
- To generate a secret key for the predicate ϕ'_S, first define a vector $\vec{v} \in \mathbb{Z}_N^{\ell+1}$ as follows:

$$\text{for } 1 \leq i \leq \ell\text{:} \quad v_i = 1 \text{ iff } i \in S$$
$$v_{\ell+1} = 1.$$

  Then output the key obtained by running $\mathsf{GenKey}_{SK}(f_{\vec{v}})$.

- To encrypt a message M for the attribute $S' \subseteq A$, define a vector $\vec{v}'$ as follows:

$$\text{for } 1 \leq i \leq \ell\text{:} \quad v_i = 1 \text{ iff } i \in S'$$
$$v_{\ell+1} = -t \bmod N.$$

  Then output the ciphertext $C \leftarrow \mathsf{Enc}_{PK}(\vec{v}', M)$.

Since $|S \cap S'| = t$ exactly when $\langle \vec{v}, \vec{v}' \rangle = 0$, correctness and security follow.

References

[1] S. Al-Riyami, J. Malone-Lee, and N. Smart. Escrow-free encryption supporting cryptographic workflow. *Intl. J. Information Security*, 5(4):217–229, 2006.

[2] W. Bagga and R. Molva. Policy-based cryptography and applications. In *Financial Cryptography*, 2005.

[3] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *IEEE Symposium on Security and Privacy*, 2007.

[4] D. Boneh and X. Boyen. Efficient selective-ID identity based encryption without random oracles. In *Advances in Cryptology — Eurocrypt 2004*.

[5] D. Boneh and X. Boyen. Secure identity based encryption without random oracles. In *Advances in Cryptology — Crypto 2004*.

[6] D. Boneh and X. Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *J. Cryptology*, 21(2):149–177, 2008.

[7] D. Boneh, X. Boyen, and E. Goh. Hierarchical identity based encryption with constant size ciphertexts. In *Advances in Cryptology — Eurocrypt 2005*.

[8] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *Advances in Cryptology — Crypto 2004*.

[9] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano. Public-key encryption with keyword search. In *Advances in Cryptology — Eurocrypt 2004*.

[10] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. *SIAM J. Computing*, 32(3):586–615, 2003.

[11] D. Boneh, E.-J. Goh, and K. Nissim. Evaluating 2-DNF formulas on ciphertexts. In *Theory of Cryptography Conference*, 2005.

[12] D. Boneh and B. Waters. Conjunctive, subset, and range queries on encrypted data. In *Theory of Cryptography Conference*, 2007.

[13] X. Boyen and B. Waters. Anonymous hierarchical identity-based encryption (without random oracles). In *Advances in Cryptology — Crypto 2006*.

[14] J. Camenisch and A. Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *Advances in Cryptology – Crypto 2004*.

[15] R. Canetti, S. Halevi, and J. Katz. A forward-secure public-key encryption scheme. *J. Cryptology*, 20(3):265–294, 2007.

[16] C. Cocks. An identity based encryption scheme based on quadratic residues. In *Proc. IMA Intl. Conf. on Cryptography and Coding*, 2001.

[17] C. Gentry. Practical identity-based encryption without random oracles. In *Advances in Cryptology — Eurocrypt 2006*.

[18] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM CCCS*, 2006.

[19] A. Joux. A one round protocol for tripartite Diffie-Hellman. In *Proceedings of ANTS IV*, volume 1838 of *LNCS*, pages 385–394. Springer-Verlag, 2000.

[20] A. Joux and K. Nguyen. Separating decision Diffie-Hellman from Diffie-Hellman in cryptographic groups. *J. of Cryptology*, 16(4):239–247, 2003. Early version in Cryptology ePrint Archive, Report 2001/003.

[21] V. Nechaev. Complexity of a determinate algorithm for the discree logarithm. *Math. Notes*, 55(2):165–172, 1974.

[22] R. Ostrovsky, A. Sahai, and B. Waters. Attribute-based encryption with non-monotonic access structures. In *ACM CCCS*, 2007.

[23] A. Sahai and B. Waters. Fuzzy identity-based encryption. In *Advances in Cryptology — Eurocrypt 2005*.

[24] A. Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology — Crypto '84*.

[25] E. Shi, J. Bethencourt, H. T.-H. Chan, D. X. Song, and A. Perrig. Multi-dimensional range queries over encrypted data. In *IEEE Symposium on Security and Privacy*, 2007.

[26] V. Shoup. Lower bounds for discrete logarithms and related problems. In *Proceedings of Eurocrypt 1997*. Springer-Verlag, 1997.

[27] B. Waters. Efficient identity-based encryption without random oracles. In *Advances in Cryptology — Eurocrypt 2005*.

A Justifying our Assumptions in the Generic Group Model

We justify Assumptions 1 and 2 by showing that they hold in generic bilinear groups of composite order N, as long as finding a non-trivial factor of N is hard. In doing so, we first prove two "master theorems" for hardness in generic groups of composite order. These theorems generalize the result by Boneh, Boyen, and Goh [7] in two ways: in addition to handling groups of composite order, they can be used for proving soundness of assumptions where the target element is in the bilinear group $\mathbb{G}$ (instead of the target group $\mathbb{G}_T$). Thus, it also applies to assumptions such as the linear assumption introduced by Boneh, Boyen, and Shacham [8] and the subgroup decision assumption introduced by Boneh, Goh, and Nissim [11].

A.1 The Generic Group Model: an Overview

The generic group model was introduced in [21, 26], and has been extended to the case of bilinear groups in [7]. This model provides a way to study "generic" group algorithms that act "independently" of the group representation (and therefore apply to any group, as long as the group operation itself can be computed in polynomial time), in a way made more precise below. It is important to qualify that various *non*-generic group algorithms are known for specific groups, and so a proof of security in the generic group model does not guarantee any security when the group is instantiated in some concrete fashion. It is, in part, for this reason that we have proved security of our constructions relative to our stated assumptions (and now justify the assumptions in the generic group model), rather than aiming for a direct proof that our constructions are secure in the generic group model.

In the generic group model, algorithms are not given any "actual" representations of group elements but are instead only given access to group elements via their "handles". So, for example, an element g may be represented by the handle "1" and h by the handle "2"; an algorithm can multiply these two elements by explicitly requesting mult("1", "2"). In response to this instruction, the group element gh is computed. If element gh has not already been assigned a handle, a new handle is assigned and returned to the algorithm; if gh has already been assigned a handle, that handle is returned. (So, for example, if g were the identity element then the instruction mult(“1”, “2”) would simply return “2”.) Note that this formalism allows the algorithm to check equality of elements, since two elements are equal iff they have the same handle. We also allow an exponentiation operation exp which takes as input an element's handle and an integer and returns the handle of the given element raised to the given power. (We allow negative exponents, so that inverses can also be computed.) In general, we restrict the algorithm to only using as input handles that it has already been given.[3]

In the setting of bilinear groups, we have two groups each with their own multiplication and exponentiation instructions and whose elements all have distinct handles. We also have a pairing instruction that takes as input two handles of elements from the first group and outputs the handle of an element from the second (“target”) group.

A.2 A “Master Theorem” for Hardness in Composite Order Bilinear Groups

Before stating our theorems, we introduce some notation. We will consider cyclic bilinear groups of order N, where $N = \prod_{i=1}^{m} p_i$ is the product of m distinct primes, each larger than 2^n. Let $\mathbb{G}$ denote the “base group” and let $\mathbb{G}_T$ denote the “target group”; i.e., the bilinear map $\hat{e}$ is from $\mathbb{G} \times \mathbb{G}$ to $\mathbb{G}_T$. Each element $g \in \mathbb{G}$ can be written as $g = g_{p_1}^{a_1} g_{p_2}^{a_2} \cdots g_{p_m}^{a_m}$, where $a_i \in \mathbb{Z}_{p_i}$ and g_{p_i} denotes some fixed generator of the subgroup of order p_i. We can therefore represent each element $g \in \mathbb{G}$ as an m-tuple $(a_1, \ldots, a_m)$. We can do the same with elements in $\mathbb{G}_T$ (with respect to the generators $\hat{e}(p_i, p_i)$), and will represent elements in $\mathbb{G}_T$ as bracketed tuples $[a_1, \ldots, a_m]$.

Using the above notation, the product of $(a_1, \ldots, a_m)$ and $(b_1, \ldots, b_m)$ is the element $(a_1 + b_1, \ldots, a_m + b_m)$, where addition in component i is done modulo $\mathbb{Z}_{p_i}$. Similarly $(a_1, \ldots, a_m)$ raised to the power $\gamma \in \mathbb{Z}$ is the element $(\gamma a_1, \ldots, \gamma a_m)$. (Analogous results hold for elements of $\mathbb{G}_T$.) It will be therefore be convenient to treat these tuples as “vectors” where vector addition corresponds to multiplication in the group and vector multiplication by a scalar corresponds to group exponentiation. The pairing of $(a_1, \ldots, a_m), (b_1, \ldots, b_m) \in \mathbb{G}$ gives the element $[a_1 b_1, \ldots, a_m b_m] \in \mathbb{G}_T$.

In an experiment involving the generic group, we will present an algorithm with a set of elements generated at random according to some distribution. We will describe these random variables using formal variables (written using capital letters) that are each chosen independently and uniformly at random from the appropriate domain. For example, a random element of $\mathbb{G}$ would be described as $(X_1, \ldots, X_m)$, where each X_i is chosen uniformly from $\mathbb{Z}_{p_i}$. We say a random variable expressed in this way has *degree* t if the maximum degree of any variable is t. Dependencies are made explicit by re-using the same formal variable; for example, a random “Diffie-Hellman-like” tuple (with $m = 2$) would be described by the three elements $(X_1, X_2), (Y_1, Y_2)$, and $(X_1 Y_1, X_2 Y_2)$. Random variables taking values in $\mathbb{G}_T$ are expressed in the same way, but using the bracket notation.

Given random variables $X, B_1, \ldots, B_\ell$ (expressed as above) over the same group, we say that X *is dependent on* $\{B_i\}$ if there exist $\gamma_i \in \mathbb{Z}_N^*$ such that $X = \sum_i \gamma_i B_i$, where equality refers to equality in terms of the underlying formal variables. If no such $\{\gamma_i\}$ exist, then X is said to be *independent* of $\{B_i\}$.

Given a random variable $A = (X_1, \ldots, X_m)$, when we say that an algorithm is given A we mean that random $x_1, \ldots, x_m$ are chosen appropriately and the adversary is given (the handle for) the element $(x_1, \ldots, x_m)$.

[3]Another way to ensure this is to use randomly-generated handles that the adversary will be unable to guess except with negligible probability.

We may now state our theorems.

Theorem A.1. *Let $N = \prod_{i=1}^{m} p_i$ be a product of distinct primes, each greater than 2^n. Let $\{A_i\}$ be random variables over $\mathbb{G}$, and let $\{B_i\}, T_0, T_1$ be random variables over $\mathbb{G}_T$, where all random variable have degree at most t. Consider the following experiment in the generic group model:*

> *An algorithm is given N, $\{A_i\}$, and $\{B_i\}$. A random bit b is chosen, and the adversary is given T_b. The algorithm outputs a bit b', and succeeds if $b' = b$. The algorithm's advantage is the absolute value of the difference between its success probability and $1/2$.*

Say each of T_0 and T_1 is independent of $\{B_i\} \cup \{\hat{e}(A_i, A_j)\}$. Then given any algorithm $\mathcal{A}$ issuing at most q instructions and having advantage δ in the above experiment, $\mathcal{A}$ can be used to find a non-trivial factor of N (in time polynomial in n and the running time of $\mathcal{A}$) with probability at least $\delta - O(q^2 t/2^n)$.

Thus, if N is generated in such a way that it is hard to find a non-trivial factor of N, the advantage of any polynomial-time algorithm $\mathcal{A}$ is negligible in n.

Proof In the original game, each of the random variables $\{A_i\}$, $\{B_i\}$, T_0, T_1 are instantiated by choosing random values for each of the formal variables and giving the handles of $\{A_i\}$, $\{B_i\}$, and T_b to the algorithm $\mathcal{A}$. The algorithm then issues a sequence of multiplication, exponentiation, and pairing instructions, and is given in return the appropriate handles. Finally, the algorithm outputs a bit b' and its advantage is measured as defined above.

We next define a second game in which the formal variables are never concretely instantiated, but instead the game only keeps track of the formal polynomials themselves. Furthermore, the game now uses identical handles for two elements only if these elements are equal *as formal polynomials* in each of their components. (So, in the original game the random variables $X = (X_1, \ldots, X_m)$ and $Y = (Y_1, \ldots, Y_m)$ could be assigned the same handle if it happened to be the case that $X_i = Y_i$ for all i. In this game, however, these two tuples of formal polynomials are always treated as different.) This only introduces a difference in case it happens during the course of the experiment that two different formal polynomials would take on the same value. For any particular pair of elements, the probability that this occurs is bounded by $2t/2^n$ (since the maximum degree of any polynomial constructed during the course of the experiment is $2t$). Summing over all pairs of elements produced during the course of the experiment shows that the statistical difference between these experiments is $O(q^2 \cdot t/2^n)$.

In the third game, we record the formal polynomials as before except that now all computation, in each of the m components, is done modulo N rather than modulo the appropriate p_i. Now, two elements are assigned identical handles only if they are equivalent as (tuples of) formal polynomials over $\mathbb{Z}_N$. This only introduces a difference in case two polynomials are generated during the course of the experiment that are different modulo N but would have been identical modulo one of the p_i. But whenever this occurs, a non-trivial factor of N can be recovered from the coefficients of any two such polynomials.

Finally, we observe that in the third game the only possible way in which the algorithm can distinguish whether it is given T_0 or T_1 is if the algorithm is able to generate a formal polynomial that would be symbolically equivalent to some previously-generated polynomial for one value of b but not the other. But in this case, we can write (for some b)

$$\gamma \cdot T_b = \sum_{i,j} \gamma_{i,j} \cdot \hat{e}(A_i, A_j) + \sum_i \gamma_i \cdot B_i$$

where $\gamma \neq 0$ and equality denotes symbolic equality in terms of the formal variables constituting the different random variables. By assumption of independence of T_0, T_1, it must be the case that one of the coefficients of the above equation is not in $\mathbb{Z}_N^*$. But then a non-trivial factor of N can be recovered. ■

Theorem A.2. *Let $N = \prod_{i=1}^m p_i$ be a product of distinct primes, each greater than 2^n. Let $\{A_i\}, T_0, T_1$ be random variables over $\mathbb{G}$, and let $\{B_i\}$ be random variables over $\mathbb{G}_T$, where all random variable have degree at most t. Consider the same experiment as in Theorem A.1.*

Let $S \stackrel{\text{def}}{=} \{i \mid \hat{e}(T_0, A_i) \neq \hat{e}(T_1, A_i)\}$ (where inequality refers to inequality as formal polynomials). Say each of T_0 and T_1 is independent of $\{A_i\}$, and furthermore that for all $k \in S$ it holds that $\hat{e}(T_0, A_k)$ is independent of $\{B_i\} \cup \{\hat{e}(A_i, A_j)\} \cup \{\hat{e}(T_0, A_i)\}_{i \neq k}$, and $\hat{e}(T_1, A_k)$ is independent of $\{B_i\} \cup \{\hat{e}(A_i, A_j)\} \cup \{\hat{e}(T_1, A_i)\}_{i \neq k}$. Then given any algorithm $\mathcal{A}$ issuing at most q instructions and having advantage δ, the algorithm can be used to find a non-trivial factor of N (in time polynomial in n and the running time of $\mathcal{A}$) with probability at least $\delta - O(q^2 t/2^n)$.

Thus, if N is generated in such a way that it is hard to find a non-trivial factor of N, the advantage of any polynomial-time algorithm $\mathcal{A}$ is negligible in n.

Proof The proof is identical to the proof of Theorem A.1 except for the analysis of the third game. As in the earlier proof, in the third game the only possible way in which the algorithm can distinguish whether it is given T_0 or T_1 is if the algorithm is able to generate a formal polynomial that would be symbolically equivalent to some previously-generated polynomial for one value of b but not the other. But then we either have (for some b)

$$\gamma \cdot T_b = \sum_i \gamma_i A_i$$

(with $\gamma \neq 0$), or else we have

$$\sum_{i \in S} \alpha_i \cdot \hat{e}(T_b, A_i) + \sum_{i \notin S} \beta_i \cdot \hat{e}(T_b, A_i) = \sum_i \gamma_i \cdot B_i + \sum_{i,j} \gamma_{i,j} \cdot \hat{e}(A_i, A_j),$$

where $\alpha_i \neq 0$ for at least one $i \in S$ (otherwise, symbolic equality would hold for both values of b). By the independence assumptions, this implies that a non-trivial factor of N can be recovered. ■

A.3 Applying the Master Theorem to Our Assumptions

We now show how to apply the theorems of the previous section to prove that our assumptions hold in the generic group model.

Assumption 2. We begin with Assumption 2 (since it corresponds to the simpler Theorem A.1). Using the notation of the previous section, our second assumption may be written as:

$$\begin{gathered} A_1 = (1,0,0), \;\; A_2 = (0,1,0), \;\; A_3 = (0,0,1), \;\; A_4 = (X,0,0) \\ A_5 = (S,0,0), \;\; A_6 = (XS, Y_1, 0), \;\; A_7 = (\Gamma, Y_2, 0), \;\; B_1 = [X\,\Gamma, 0, 0]. \\ T_0 = [X\,\Gamma S, 0, 0], \;\; T_1 = [Z_1, Z_2, Z_3] \end{gathered}$$

It is immediate that T_1 is independent of $B_1 \cup \{\hat{e}(A_i, A_j)\}$. As for T_0, the only way a dependence can occur is if there is an element of $\mathbb{G}_T$ with first component equal to $X\,\Gamma S$; this occurs only in

$\hat{e}(A_6, A_7)$, but in that element there is an additional component Y_1Y_2 in the second component that cannot be canceled.

Assumption 1. Assumption 1 may be written as:

$$\begin{gathered} A_1 = (1,0,0), \;\; A_2 = (0,0,1), \;\; A_3 = (0,1,Y_1), \\ A_4 = (B,0,0), \;\; A_5 = (B^2,0,0), \;\; A_6 = (A,1,0), \\ A_7 = (AB,Y_2,0), \;\; A_8 = (S,0,0), \;\; A_9 = (BS,Y_3,Y_4) \\ T_0 = (B^2S,0,Z_1), \;\; T_1 = (B^2S,Z_2,Z_1) \end{gathered}.$$

It is not difficult to see that both T_0 and T_1 are independent of $\{A_i\}$. Using the notation of Theorem A.2, we have $S = \{3,6,7,9\}$. Considering T_0 first, we obtain the following tuples:

$$\begin{aligned} C_3 &\stackrel{\text{def}}{=} \hat{e}(T_0, A_3) = [0,0,Z_1Y_1] & C_6 &\stackrel{\text{def}}{=} \hat{e}(T_0, A_6) = [AB^2S,0,0] \\ C_7 &\stackrel{\text{def}}{=} \hat{e}(T_0, A_7) = [AB^3S,0,0] & C_9 &\stackrel{\text{def}}{=} \hat{e}(T_0, A_9) = [B^3S^2,0,Z_1Y_4]. \end{aligned}$$

It is clear that C_3 and C_9 are independent of anything else, since an element in $\mathbb{G}_T$ whose third component contains Z_1Y_1 (resp, Z_1Y_4) cannot be generated any other way. As for C_6, the only other way to obtain an element whose third component contains AB^2S is by computing $\hat{e}(A_7, A_9)$, which yields the element $[AB^2S, Y_2Y_3, 0]$. But there is no other way to generate an element whose second component is Y_2Y_3, and hence no way to cancel that term. Finally, considering C_7, there is no other way to obtain an element containing a term of the form B^3S^2. Thus, each of the above elements satisfy the independence requirement of Theorem A.1. Exactly analogous arguments apply for the case of T_1.

B Security Definition for Inner-Product Encryption

Here, we re-state Definition 2.2 in the particular setting of our main construction, which is a predicate-only scheme where the set of attributes[4] is $\Sigma = \mathbb{Z}_N^n$ and the class of predicates is $\mathcal{F} = \{f_{\vec{x}} \mid \vec{x} \in \mathbb{Z}_N^n\}$ such that $f_{\vec{x}}(\vec{y}) = 1$ iff $\langle \vec{x}, \vec{y} \rangle = 0 \bmod N$.

Definition B.1. *A predicate-only encryption scheme for $\Sigma, \mathcal{F}$ as above is* attribute-hiding *if for all* PPT *adversaries $\mathcal{A}$, the advantage of $\mathcal{A}$ in the following experiment is negligible in the security parameter n:*

1. $\mathsf{Setup}(1^n)$ *is run to generate keys PK, SK. This defines a value N which is given to $\mathcal{A}$.*
2. *$\mathcal{A}$ outputs $\vec{x}, \vec{y} \in \mathbb{Z}_N^n$, and is then given PK.*
3. *$\mathcal{A}$ may adaptively request keys corresponding to the vectors $\vec{v}_1, \ldots, \vec{v}_\ell \in \mathbb{Z}_N^n$, subject to the restriction that, for all i, $\langle \vec{v}_i, \vec{x} \rangle = 0 \bmod N$ if and only if $\langle \vec{v}_i, \vec{y} \rangle = 0 \bmod N$. In response, $\mathcal{A}$ is given the corresponding keys $SK_{\vec{v}_i} \leftarrow \mathsf{GenKey}_{SK}(f_{\vec{v}_i})$.*
4. *A random bit b is chosen. If $b = 0$ then $\mathcal{A}$ is given $C \leftarrow \mathsf{Enc}_{PK}(\vec{x})$, and if $b = 1$ then $\mathcal{A}$ is given $C \leftarrow \mathsf{Enc}_{PK}(\vec{y})$.*
5. *The adversary may continue to request keys for additional vectors, subject to the same restriction as before.*
6. *$\mathcal{A}$ outputs a bit b', and succeeds if $b' = b$.*

The advantage of $\mathcal{A}$ is the absolute value of the difference between its success probability and $1/2$.

[4] We consider vectors of length n, the security parameter, for convenience only.

C A Full-Fledged Predicate Encryption Scheme

In Section 4, we showed a construction of a *predicate-only* scheme. Here, we extend that scheme to obtain a full-fledged predicate encryption scheme in the sense of Definition 2.1. The additions in the present scheme are boxed for the reader's convenience.

$\mathsf{Setup}(1^n)$ The setup algorithm first runs $\mathcal{G}(1^n)$ to obtain $(p, q, r, \mathbb{G}, \mathbb{G}_T, \hat{e})$ with $\mathbb{G} = \mathbb{G}_p \times \mathbb{G}_q \times \mathbb{G}_r$. Next, it computes g_p, g_q, and g_r as generators of $\mathbb{G}_p, \mathbb{G}_q$, and $\mathbb{G}_r$, respectively. It then chooses $R_{1,i}, R_{2,i} \in \mathbb{G}_r$ and $h_{1,i}, h_{2,i} \in \mathbb{G}_p$ uniformly at random for $i = 1$ to n, and $R_0 \in \mathbb{G}_r$ uniformly at random. $\boxed{\text{It also chooses random } \gamma \in \mathbb{Z}_p \text{ and } h \in \mathbb{G}_p.}$ The public parameters include $(N = pqr, \mathbb{G}, \mathbb{G}_T, \hat{e})$ along with:

$$PK = \Big(g_p, \quad g_r, \quad Q = g_q \cdot R_0, \quad \boxed{P = \hat{e}(g_p, h)^\gamma}, \quad \{H_{1,i} = h_{1,i} \cdot R_{1,i}, \quad H_{2,i} = h_{2,i} \cdot R_{2,i}\}_{i=1}^n\Big).$$

The master secret key SK is $\Big(p, q, r, g_q, \boxed{h^{-\gamma}}, \{h_{1,i},\ h_{2,i}\}_{i=1}^n\Big)$.

$\mathsf{Enc}_{PK}(\vec{x}, M)$ Let $\vec{x} = (x_1, \ldots, x_n)$ with $x_i \in \mathbb{Z}_N$, and view M as an element of $\mathbb{G}_T$. This algorithm chooses random $s, \alpha, \beta \in \mathbb{Z}_N$ and $R_{3,i}, R_{4,i} \in \mathbb{G}_r$ for $i = 1$ to n. It outputs the ciphertext

$$C = \left(\boxed{C' = M \cdot P^s}, \ C_0 = g_p^s, \quad \left\{C_{1,i} = H_{1,i}^s \cdot Q^{\alpha \cdot x_i} \cdot R_{3,i}, \quad C_{2,i} = H_{2,i}^s \cdot Q^{\beta \cdot x_i} \cdot R_{4,i}\right\}_{i=1}^n\right).$$

$\mathsf{GenKey}_{SK}(\vec{v})$ Let $\vec{v} = (v_1, \ldots, v_n)$. This algorithm chooses random $r_{1,i}, r_{2,i} \in \mathbb{Z}_p$ for $i = 1$ to n, random $R_5 \in \mathbb{G}_r$, random $f_1, f_2 \in \mathbb{Z}_q$, and random $Q_6 \in \mathbb{G}_q$. It then outputs

$$SK_{\vec{v}} = \left(K = R_5 \cdot Q_6 \cdot h^{-\gamma} \cdot \prod_{i=1}^n h_{1,i}^{-r_{1,i}} \cdot h_{2,i}^{-r_{2,i}}, \quad \left\{K_{1,i} = g_p^{r_{1,i}} \cdot g_q^{f_1 \cdot v_i}, \quad K_{2,i} = g_p^{r_{2,i}} \cdot g_q^{f_2 \cdot v_i}\right\}_{i=1}^n\right).$$

$\mathsf{Dec}_{SK_{\vec{v}}}(C)$ Let C and $SK_{\vec{v}}$ be as above. The decryption algorithm outputs

$$\boxed{C'} \cdot \hat{e}(C_0, K) \cdot \prod_{i=1}^n \hat{e}(C_{1,i}, K_{1,i}) \cdot \hat{e}(C_{2,i}, K_{2,i}).$$

As we have described it, decryption never returns an error (i.e., even when $\langle \vec{v}, \vec{x} \rangle \neq 0$). We will show below that when $\langle \vec{v}, \vec{x} \rangle \neq 0$ then the output is essentially a random element in the order-q subgroup of $\mathbb{G}_T$. By restricting the message space to some efficiently-recognizable set of negligible density in this subgroup, we recover the desired semantics by returning an error if the recovered message does not lie in this space.

Correctness. Let C and $SK_{\vec{v}}$ be as above. Then

$$
\begin{aligned}
& C' \cdot \hat{e}(C_0, K) \cdot \prod_{i=1}^{n} \hat{e}(C_{1,i}, K_{1,i}) \cdot \hat{e}(C_{2,i}, K_{2,i}) \\
&= M \cdot P^s \cdot \hat{e}\left(g_p^s, \quad R_5 Q_6 h^{-\gamma} \prod_{i=1}^{n} h_{1,i}^{-r_{1,i}} h_{2,i}^{-r_{2,i}}\right) \\
&\quad \cdot \prod_{i=1}^{n} \hat{e}\left(H_{1,i}^s Q^{\alpha \cdot x_i} R_{3,i}, \quad g_p^{r_{1,i}} g_q^{f_1 \cdot v_i}\right) \cdot \hat{e}\left(H_{2,i}^s Q^{\beta \cdot x_i} R_{4,i}, \quad g_p^{r_{2,i}} g_q^{f_2 \cdot v_i}\right) \\
&= M \cdot P^s \cdot \hat{e}\left(g_p^s, \quad h^{-\gamma} \prod_{i=1}^{n} h_{1,i}^{-r_{1,i}} h_{2,i}^{-r_{2,i}}\right) \cdot \prod_{i=1}^{n} \hat{e}\left(h_{1,i}^s\, g_q^{\alpha \cdot x_i}, \ \ g_p^{r_{1,i}} g_q^{f_1 \cdot v_i}\right) \cdot \hat{e}\left(h_{2,i}^s\, g_q^{\beta \cdot x_i}, \ \ g_p^{r_{2,i}} g_q^{f_2 \cdot v_i}\right) \\
&= M \cdot P^s \cdot \hat{e}(g_p, h)^{-\gamma s} \cdot \prod_{i=1}^{n} \hat{e}(g_q, g_q)^{(\alpha f_1 + \beta f_2) x_i v_i} \quad = \quad M \cdot \hat{e}(g_q, g_q)^{(\alpha f_1 + \beta f_2)\langle \vec{x}, \vec{v} \rangle}.
\end{aligned}
$$

If $\langle \vec{x}, \vec{v} \rangle = 0 \bmod N$, then the above evaluates to M. If $\langle \vec{x}, \vec{v} \rangle \neq 0 \bmod N$ there are two cases: if $\langle \vec{x}, \vec{v} \rangle \neq 0 \bmod q$ then the above evaluates to an element whose distribution is statistically close to uniform in the order-q subgroup of $\mathbb{G}_T$. (Recall that α, β are chosen at random.) It is possible that $\langle \vec{x}, \vec{v} \rangle = 0 \bmod q$, in which case the above always evaluates to M; however, this reveals a non-trivial factor of N and so an adversary can cause this condition to occur with only negligible probability.

C.1 Proof of Security

Theorem C.1. *If $\mathcal{G}$ satisfies Assumptions 1 and 2 then the scheme described in the previous section is an attribute-hiding predicate encryption scheme.*

We prove that the scheme described in the previous section satisfies Definition 2.2. In proving this, we distinguish two cases: when $M_0 = M_1$ and when $M_0 \neq M_1$. We show that the adversary's probability of success conditioned on the occurrence of each case is negligibly-close to 1/2.

A proof for the case $M_0 = M_1$ follows *mutatis mutandis* from the proof given in Section 4. Specifically, if $M_0 = M_1 = M$ then the adversary gets no advantage from the extra term $M \cdot P^s$ included in the challenge ciphertext and so the only point to verify is that, throughout the proofs in Sections 4.4.1 and 4.4.2, the simulator can compute the value P^s (so that it can construct the additional element $C' = M \cdot P^s$). This is easy to do if the simulator computes P exactly as in the Setup algorithm, and stores $h^{-\gamma}$. We omit the straightforward details.

Given the above, we concentrate here on proving security under the assumption that $M_0 \neq M_1$. Since we are considering only this case, we will assume the adversary is restricted to requesting keys corresponding to vectors $\vec{v}$ for which $\langle \vec{v}, \vec{x} \rangle \neq 0$ and $\langle \vec{v}, \vec{y} \rangle \neq 0$, where $\vec{x}, \vec{y}$ are the vectors output by the adversary at the outset of the experiment (cf. Definition B.1). We establish the result in this case using a sequence of games, defined as follows.

Game_0: The challenge ciphertext is generated as a proper encryption of M_0 using $\vec{x}$. That is, we choose random $s, \alpha, \beta \in \mathbb{Z}_N$ and random $\{R_{3,i}, R_{4,i}\} \in \mathbb{G}_r$, and compute the ciphertext as

$$
C = \left(C' = M_0 \cdot P^s, \quad C_0 = g_p^s, \quad \left\{C_{1,i} = H_{1,i}^s Q^{\alpha x_i} R_{3,i}, \quad C_{2,i} = H_{2,i}^s Q^{\beta x_i} R_{4,i}\right\}_{i=1}^{n}\right).
$$

Game_1: We now generate the challenge ciphertext as a proper encryption of a random element of $\mathbb{G}_T$, but still using $\vec{x}$. I.e., the ciphertext is formed as above except that C' is chosen uniformly from $\mathbb{G}_T$.

Game_2: We now generate the $\{C_{2,i}\}$ components as if encryption were done using $\vec{0}$. That is, we choose random $s, \alpha, \beta \in \mathbb{Z}_N$, random $\{R_{3,i}, R_{4,i}\} \in \mathbb{G}_r$, and random $C' \in \mathbb{G}_T$, and compute the ciphertext as

$$C = \Big(C', \quad C_0 = g_p^s, \quad \{\, C_{1,i} = H_{1,i}^s Q^{\alpha x_i} R_{3,i}, \quad C_{2,i} = H_{2,i}^s\, R_{4,i}\}_{i=1}^n\Big)\,.$$

Note that this exactly parallels Game_2 in the proof of Theorem 4.1.

Game_3: We now generate the $\{C_{2,i}\}$ components using vector $\vec{y}$. That is, we choose random $s, \alpha, \beta \in \mathbb{Z}_N$, random $\{R_{3,i}, R_{4,i}\} \in \mathbb{G}_r$, and random $C' \in \mathbb{G}_T$, and compute the ciphertext as

$$C = \Big(C', \quad C_0 = g_p^s, \quad \Big\{ C_{1,i} = H_{1,i}^s Q^{\alpha x_i} R_{3,i}, \quad C_{2,i} = H_{2,i}^s Q^{\beta y_i} R_{4,i}\Big\}_{i=1}^n\Big)\,.$$

Note that this exactly parallels Game_3 in the proof of Theorem 4.1.

Game_4 and Game_5: These games are defined symmetrically to Game_2 and Game_3, as in the proof of Theorem 4.1. We continue to let C' be a random element of $\mathbb{G}_T$. Note that Game_5 corresponds to a proper encryption of a random element of $\mathbb{G}_T$ using $\vec{y}$.

Game_6: The challenge ciphertext is generated as a proper encryption of M_1 using $\vec{y}$.

In the next section we prove that, under Assumption 2, Game_0 and Game_1 are indistinguishable. Indistinguishability of Game_1 and Game_5 follows, as earlier, *mutatis mutandis* from the proofs in Sections 4.4.1 and 4.4.2. The proof that Game_5 and Game_6 are indistinguishable is symmetric to the proof that Game_0 and Game_1 are indistinguishable, and is therefore omitted.

C.1.1 Indistinguishability of Game_0 and Game_1

Fix an adversary $\mathcal{A}$. We describe a simulator who is given $(N = pqr, \mathbb{G}, \mathbb{G}_T, \hat{e})$ along with the elements $g_p,\ g_q,\ g_r,\ h,\ g_p^s,\ h^s Q_1,\ g_p^\gamma Q_2,\ \hat{e}(g_p, h)^\gamma$, and an element T which is either equal to $\hat{e}(g_p, h)^{\gamma s}$ or is uniformly distributed in $\mathbb{G}_T$. Note that the simulator is now able to sample uniformly from $\mathbb{G}_q$ and $\mathbb{G}_r$ using g_q and g_r, respectively. In particular, the simulator can sample uniformly from $\mathbb{G}_{qr} = \mathbb{G}_q \times \mathbb{G}_r$. The simulator interacts with $\mathcal{A}$ as we now describe.

Public parameters. The simulator begins by giving N to $\mathcal{A}$, who outputs vectors $\vec{x}, \vec{y}$. The simulator chooses random $\{w_{1,i}, w_{2,i}\} \in \mathbb{Z}_N$ and random $\{R_{1,i}, R_{2,i}\}, R_0 \in \mathbb{G}_r$, includes $(N, \mathbb{G}, \mathbb{G}_T, \hat{e})$ in the public parameters, and sets the remainder of the parameters as follows:

$$PK = \big(g_p,\ g_r,\ Q = g_q R_0,\ P = \hat{e}(g_p, h)^\gamma, \quad \big\{H_{1,i} = h^{x_i} g_p^{w_{1,i}} R_{1,i}, \quad H_{2,i} = h^{x_i} g_p^{w_{2,i}} R_{2,i}\big\}_{i=1}^n\big)\,.$$

The simulator is implicitly setting $h_{1,i} = h^{x_i} g_p^{w_{1,i}}$ and $h_{2,i} = h^{x_i} g_p^{w_{2,i}}$. Note that PK has the appropriate distribution.

Key derivation. The adversary $\mathcal{A}$ may request secret keys corresponding to different vectors $\vec{v}$, as long as $\langle \vec{v}, \vec{x}\rangle \neq 0$ (we do not use the fact that $\langle \vec{v}, \vec{y}\rangle \neq 0$ here). We now describe how the simulator prepares the secret key corresponding to any such vector.

Say the adversary requests the secret key for vector $\vec{v}$, and let $k = 1/2 \cdot \langle \vec{x}, \vec{v} \rangle \bmod N$. (If $\gcd(\langle \vec{x}, \vec{v} \rangle, N) \neq 1$) then the adversary has factored N; this occurs with negligible probability.) The simulator first chooses random $f_1', f_2', \{r_{1,i}', r_{2,i}'\} \in \mathbb{Z}_N$. Next, for all i it computes:

$$\begin{aligned} K_{1,i} &= \left(g_p^\gamma Q_2\right)^{-kv_i} \cdot g_q^{f_1' v_i} \cdot g_p^{r_{1,i}'} \\ &= g_p^{-kv_i\gamma + r_{1,i}'} \cdot g_q^{(f_1' - kc)\cdot v_i} \end{aligned}$$

(where we set $c = \log_{g_q} Q_2$), and

$$\begin{aligned} K_{2,i} &= \left(g_p^\gamma Q_2\right)^{-kv_i} \cdot g_q^{f_2' v_i} \cdot g_p^{r_{2,i}'} \\ &= g_p^{-kv_i\gamma + r_{2,i}'} \cdot g_q^{(f_2' - kc)\cdot v_i}. \end{aligned}$$

The simulator then chooses random $QR \in \mathbb{G}_{qr}$ and computes:

$$K = QR \cdot \prod_{i=1}^{n} \left(\left(g_p^{w_{1,i}} h^{x_i}\right)^{-r_{1,i}'} \cdot \left(g_p^\gamma Q_2\right)^{kv_i w_{1,i}} \right) \cdot \left(\left(g_p^{w_{2,i}} h^{x_i}\right)^{-r_{2,i}'} \cdot \left(g_p^\gamma Q_2\right)^{kv_i w_{2,i}} \right).$$

Finally, the simulator hands the adversary $SK_{\vec{v}} = (K, \{K_{1,i}, K_{2,i}\}_{i=1}^n)$ as the key.

To see that this key has the correct distribution, note that by construction of the $\{K_{1,i}, K_{2,i}\}$ the simulator is implicitly setting $f_1 = f_1' - kc$ and, for all i, $r_{1,i} = -k\gamma v_i + r_{1,i}'$ (and analogously for f_2 and the $\{r_{2,i}\}$). These values are all uniformly and independently distributed in $\mathbb{Z}_N$. Next, note that

$$\begin{aligned} \prod_{i=1}^{n} \left(g_p^{w_{1,i}} h^{x_i}\right)^{-r_{1,i}'} \cdot \left(g_p^\gamma\right)^{kv_i w_{1,i}} &= \prod_{i=1}^{n} g_p^{-w_{1,i} r_{1,i}' + k\gamma v_i w_{1,i}} \cdot h^{-x_i r_{1,i}'} \\ &= \prod_{i=1}^{n} g_p^{-w_{1,i}\cdot(r_{1,i} + k\gamma v_i) + k\gamma v_i w_{1,i}} \cdot h^{-x_i \cdot (r_{1,i} + k\gamma v_i)} \\ &= \prod_{i=1}^{n} \left(h^{x_i} g_p^{w_{1,i}}\right)^{-r_{1,i}} \cdot h^{-\gamma k v_i x_i} \quad = \quad h^{-\gamma/2} \cdot \prod_{i=1}^{n} h_{1,i}^{-r_{1,i}}, \end{aligned}$$

using the fact that $\langle \vec{v}, \vec{x} \rangle = 1/2k \bmod N$. Thus, looking at K_p (the projection of K in $\mathbb{G}_p$) we see that

$$\begin{aligned} K_p &= \prod_{i=1}^{n} \left(\left(g_p^{w_{1,i}} h^{x_i}\right)^{-r_{1,i}'} \cdot \left(g_p^\gamma\right)^{kv_i w_{1,i}} \right) \cdot \left(\left(g_p^{w_{2,i}} h^{x_i}\right)^{-r_{2,i}'} \cdot \left(g_p^\gamma\right)^{kv_i w_{2,i}} \right) \\ &= h^{-\gamma} \cdot \prod_{i=1}^{n} h_{1,i}^{-r_{1,i}} \cdot h_{2,i}^{-r_{2,i}}, \end{aligned}$$

and so K_p (and hence K) is distributed appropriately.

The challenge ciphertext. The challenge ciphertext is generated as follows. The simulator chooses random $\{R_{7,i}, R_{8,i}\} \in \mathbb{G}_r$ and $Q_1' \in \mathbb{G}_q$, sets $C' = M_0 \cdot T$, sets $C_0 = g_p^s$, and computes:

$$\begin{aligned} C_{1,i} &= \left(g_p^s\right)^{w_{1,i}} \cdot (h^s Q_1)^{x_i} \cdot R_{7,i} \\ &= \left(h^{x_i} g_p^{w_{1,i}}\right)^s \cdot Q_1^{x_i} \cdot R_{7,i} \\ C_{2,i} &= \left(g_p^s\right)^{w_{2,i}} \cdot (h^s Q_1)^{x_i} \cdot (Q_1')^{x_i} \cdot R_{8,i} \\ &= \left(h^{x_i} g_p^{w_{2,i}}\right)^s \cdot (Q_1 Q_1')^{x_i} \cdot R_{8,i}. \end{aligned}$$

Analysis. By examining the projections of the components of the challenge ciphertext in the groups $\mathbb{G}_p$, $\mathbb{G}_q$, and $\mathbb{G}_r$, it can be verified that when $T = \hat{e}(g_p, h)^{\gamma s}$ the challenge ciphertext is distributed exactly as in Game_0, whereas if T is chosen uniformly from $\mathbb{G}_T$ the challenge ciphertext is distributed exactly as in Game_1. We conclude that, under Assumption 2, these two games are indistinguishable. ■