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 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 the 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 relatively few 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 what we feel is 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, though an extremely useful primitive, is rather coarse-grained: a sender encrypts a message M with respect to a given 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 applications such as encrypting data a personal hard-drive or securing point-to-point communication, where encrypted data is intended for one particular user who is known in advance by the sender.

As the Internet and applications evolve, more complex types of data will be stored in distributed settings. These will present a different set of demands for the security of stored data. In many cases, the sender will want access to the encrypted data to be governed by more complex rules; the sender may no longer intend the encrypted data for any particular user, but might instead want to enable decryption by any user satisfying a certain sender-specified policy. For example, in a health care application a patient's records should perhaps be accessible only to a physician who has treated the patient in the past.

In other scenarios, we will want to go even further and limit what a particular user learns about a record. Consider a researcher studying the correlation between the occurrence of a particular disease and the age of people that the disease afflicts. The researcher might be authorized to learn whether a record matches for the disease and, if so, the age of the patient; however, he should not learn anything more. Another example is an email firewall that should only be able to evaluate the *predicate* of whether an encrypted email message is considered to be spam, but should learn nothing else about the encrypted message.

Applications such as those sketched above will require new cryptographic mechanisms that provide more fine-grained control over access to encrypted data. *Predicate encryption schemes* are one such tool. At a high level (formal definitions are given in Section 2), secret keys in a predicate encryption scheme correspond to predicates in some class $\mathcal{F}$, and ciphertexts are associated with attributes in some 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 $f(I) = 1$. In predicate encryption systems we require a strong definition of security where we will evaluate the predicate over the hidden data (or attributes) itself. For example, these predicate encryption schemes an adversary as above learns nothing about I beyond the fact that $f_1(I) = \cdots f_\ell(I) = 0$. Furthermore, even a legitimate user who holds SK_f and can recover the plaintext from some ciphertext associated with attribute I learns nothing about I other than the fact that $f(I) = 1$. (See Definition 2.2 for a formal statement.)

Related Work The best-known example of encryption systems with fine-grained capabilities is *Identity-Based Encryption* (IBE) [18, 6, 12]. Identity-Based Encryption can be viewed as a system where a ciphertext is associated with a certain attribute or identity, I, and a user can decrypt the underlying encrypted data M if and only if there is an *equality* match between the attribute assigned to the user's private key and that of the ciphertext. One limitation of IBE systems and more expressive generalizations such as *attribute-based encryption* (ABE) [17, 15, 3, 11, 16] is that they fall into a class of encryption systems that we informally informally refer to as "attribute revealing". Attribute-revealing encryption systems can be viewed as encryption systems that guarantee that an adversary in possession of secret keys $SK_{f_1}, \ldots, SK_{f_\ell}$ and given a ciphertext associated with the attribute I learns nothing about the underlying plaintext whenever $f_1(I) = \cdots = f_\ell(I) = 0$. However, the adversary may learn I itself. More generally, an attribute-revealing scheme offers no protection of I whatsoever; typical constructions satisfying this notion reveal I in the clear. In the context of, e.g., identity-based encryption, a attribute-revealing scheme corresponds to the standard notion of security while a predicate encryption (or attribute-hiding) scheme is also anonymous.

Unfortunately, current predicate encryption systems are rather limited in their expressiveness. Song, Wagner, and Perrig [20] and Goldreich and Ostrovsky [14] gave the first such encryption systems for equality predicates in the symmetric setting and Boneh et al. [5] showed how to compute equality tests (that later came to be known as Anonymous IBE) in the public key setting. Subsequently, Boyen and Waters [9] showed the first such Anonymous IBE scheme that didn't use random oracles and also had a hierarchical structure. Gentry [13] gave a different technique for removing random oracles.

Recently, Boneh and Waters [8] showed how to construct predicates that were a conjunction over subset of fields, specified by the private key that was used to realize subset and range queries. Shi et al. [19] showed how to do more efficient queries over a small number of ranges, but in a weaker security model where the decryptor learns extra information when the predicate evaluates to true. In other work researchers have looked at issues of correctness definitions in anonymous IBE [1] and security versus efficiency tradeoffs in equality predicates [2].

1.1 Our Results

An important research direction is to construct predicate encryption schemes for classes $\mathcal{F}$ that are as expressive as possible (with the ultimate goal, of course, being to construct a scheme where $\mathcal{F}$ contains all polynomial-time predicates). The main limitation of all prior work, listed above, is that existing techniques for building attribute-hiding schemes are essentially limited to enforcing *conjunctions* of equalities. Getting slightly more technical, this is because the underlying cryptographic mechanism in the above schemes is to set up "cancellations" between the private key components and the ciphertext components. If a particular field in the ciphertext does not match with the corresponding field in the private key, the result will be random and the decryptor will know that the predicate evaluates to false.

While these cancellation techniques are useful for conjunction predicates, it is apparent that very different cryptographic techniques are needed to support predicates that include a disjunctive component. As a simple example, suppose we want a system that supports a simple **OR** predicate over two fields (where the evaluation should not revealing which field satisfied it). Previous cancellation mechanisms do not readily support such queries; if a particular ciphertext field does not match a private key field, then the evaluation should depend upon the second field and not just be sent to a random group element. While **OR** queries is just one type of predicate, several other predicate classes depend on some type of disjunctive mechanism, such as thresholds and evaluating CNF or DNF formulas. In this work we aim to realize such predicates, so we will need to introduce new techniques.

Towards realizing this goad we focus on building a class of predicates corresponding to the computation of inner products of vectors over $\mathbb{Z}_N$ for some large integer N. By making our technical objective tied more closely to the underlying mathematics we will able to achieve our overall objectives. Specifically, we will take $\Sigma = \mathbb{Z}_N^n$ as our set of attributes, and our class of predicates will be $\mathcal{F} = \{f_{\vec{x}} \mid \vec{x} \in \mathbb{Z}_N^n\}$ where $f_{\vec{x}}(\vec{y}) = 1$ iff $\langle \vec{x}, \vec{y} \rangle = 0$. (We let$\langle \vec{x}, \vec{y} \rangle$ denote the value of the standard inner product $\sum_{i=1}^n x_i \cdot y_i \bmod N$ of two vectors $\vec{x}$ and $\vec{y}$.) This can be generalized in a completely straightforward manner to support more general predicates such as $f_{\vec{x},a}(\vec{y}) = 1$ iff $\langle \vec{x}, \vec{y} \rangle = a$.

Our main result is a construction of a scheme as described above in the standard model, 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"). We haven't done this and we justify them by showing that they hold in generic groups with a bilinear map. A pessimistic interpretation of our results would be that we prove security in the generic group model, but we believe it is of additional importance that we are able to distill our necessary assumptions to ones that are compact and falsifiable.

We view our main construction as a significant step toward increasing the expressiveness of predicate encryption in general. In particular, we stress that our desired result does not follow from any trivial adaptation or combination of the existing schemes highlighted above, and our construction uses new techniques including the fact that we work in a group whose order is a product of *three* primes (but see footnote 1). Moreover, we show that any predicate encryption scheme supporting "inner product" predicates as described earlier 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. As a consequence, our work implies all the results of [8].
- We can also construct predicate encryption schemes supporting polynomial evaluation. Here, we take $\mathbb{Z}_N$ as our set of attributes, and predicates corresponds 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 addressing the open question from [8] noted earlier. In the context of identity-based encryption, this gives the ability to issue secret keys corresponding to a *set* of identities that enables decryption whenever a ciphertext is encrypted to any identity in this set (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 for hiding an encrypted biometric in the "fuzzy IBE" setting of Sahai and Waters [17], which previously hid only a payload, but revealed the biometric it was encrypted under.

We defer further discussion regarding the above until Section 5.

2 Definitions

We define the syntax of predicate encryption, as well as the notion of "attribute hiding" mentioned previously. Our definitions follow the general framework of those given in [8]. 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 $\mathcal{F}$ might depend on the security parameter and/or the master public parameters; for simplicity, we leave this 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* $\mathsf{Setup}, \mathsf{GenKey}, \mathsf{Enc}, \mathsf{Dec}$ *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.*

We will also consider a variant of the above that we call a *predicate-only scheme*. Here, Enc takes only an attribute I (and no message); the correctness requirement is that if $f(I) = 1$ then $\mathsf{Dec}_{SK_f}(\mathsf{Enc}_{PK}(I)) = 1$ whereas if $f(I) = 0$ then $\mathsf{Dec}_{SK_f}(\mathsf{Enc}_{PK}(I)) = 0$. A scheme of this sort can serve as a useful building block toward a full-fledged predicate encryption scheme.

Our definition of security corresponds to the attribute-hiding notion described informally earlier. Here, an adversary may request keys corresponding to the predicates $f_1, \ldots, f_\ell$ and is then 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.

Once again, we note that when specialized to the case when $\mathcal{F}$ consists of equality queries, this notion corresponds to *anonymous* identity-based encryption. However, our definition actually uses the "selective" notion of security first introduced in [10].

Definition 2.2. *A predicate encryption scheme with respect to $\mathcal{F}$ and Σ is* attribute hiding *(or simple* secure*) 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, 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 $SK_i \leftarrow \mathsf{GenKey}_{SK}(f_i)$.*
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 we simply omit the messages in the above experiment. For convenience, we include in Appendix A a re-statement of the definition of security given above for the particular inner-product predicate we use in our main construction.

3 Background on Pairings and Complexity Assumptions

3.1 Bilinear Groups of Composite Order

We review some general notions about bilinear groups, with an emphasis on groups of *composite order*. We follow [7] in which composite-order bilinear groups were first introduced. In contrast to all prior work using composite-order bilinear groups, however, we present our results using groups whose order N is a product of *three* (distinct) primes. This is for simplicity only, since a variant of our construction can be proven secure based on a "decisional linear"-type assumption [4] in a group of composite order N which is a product of two primes.[1]

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:

[1] This is analogous to the "folklore" transformation (based on the decisional linear assumption) that converts any scheme using groups whose order N is a product of two primes, to a scheme that uses prime-order groups.

- (Bilinear) $\forall u, v \in \mathbb{G}$, $\forall a, b \in \mathbb{Z}$, $\hat{e}(u^a, v^b) = \hat{e}(u, v)^{ab}$.
- (Non-degenerate) $\exists g \in \mathbb{G}$ such that $\hat{e}(g, g)$ has order N in $\mathbb{G}_T$.

We assume that the group action 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 description of $\mathbb{G}$ and $\mathbb{G}_T$ includes 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, e.g., $h_p \in \mathbb{G}_p$ 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 $\alpha_2 = \log_{g^{pr}} h_q$. Similar rules hold whenever $\hat{e}$ is applied to elements in disjoint subgroups.

3.2 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 by proving that they hold in the generic group model under the assumption that factoring N, the order of the group, is hard. (These proofs are omitted from the present submission). At a minimum, then, our construction can be viewed as secure in the generic group model. Nevertheless, we state our assumptions explicitly since our assumptions are non-interactive and of fixed size, and we view this as an advantage.

Assumption 1. Let $\mathcal{G}$ be as in the previous section. 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 in the previous section. 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 $h, Q_1, Q_2 \in \mathbb{G}_q$, 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.

Note that both the above assumptions imply the hardness of factoring N.

4 Our Main Construction

Our main construction is a predicate-only scheme where the set of attributes is $\Sigma = \mathbb{Z}_N^n$, and the class of predicates is $\mathcal{F} = \{f_{\vec{x}} \mid \vec{x} \in \mathbb{Z}_N^n\}$ with $f_{\vec{x}}(\vec{y}) = 1$ iff $\langle \vec{x}, \vec{y} \rangle = 0 \bmod N$. In this section we provide our predicate-only construction and give some intuition about our proof. For space considerations the details of the proof are presented in Appendix B. In Appendix C we show how our scheme can be extended to give a full-fledged predicate encryption scheme.

Intuition 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 $\vec{x} \cdot \vec{v} = 0$, 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 [7], that a random element in any of these subgroups is indistinguishable from a random element of $\mathbb{G}$.[2]. 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. 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. $\mathbb{G}_p$ 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 ordering 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, 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.

We now proceed to the formal description of our scheme.

4.1 Scheme

$\mathsf{Setup}(1^n)$ 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

[2]This is only for intuition. Our actual computational assumption is given in Section 3.

$R_{1,i}, R_{2,i} \in \mathbb{G}_r$ 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:

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

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

$\mathsf{Enc}_{\mathrm{PK}}(\vec{x})$ 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 by choosing random $\delta \in \mathbb{Z}_N$ and setting $R = g_r^{\delta}$.) It outputs the ciphertext

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

$\mathsf{GenKey}_{\mathrm{SK}}(\vec{v})$ Let $\vec{v} = (v_1, \ldots, v_n)$, and recall $\mathrm{SK} = \big(p, q, r, g_q, \{h_{1,i},\ h_{2,i}\}_{i=1}^{n}\big)$. 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

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

$\mathsf{Dec}_{\mathrm{SK}_{\vec{v}}}(C)$ Let C and $\mathrm{SK}_{\vec{v}}$ 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. Let C and $\mathrm{SK}_{\vec{v}}$ be as above. Then

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

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 the above evaluates to an element other than the identity. It is possible 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 an adversary can cause this condition to occur with only negligible probability.

The reader might notice that there appears to be some redundancy in our construction. For instance, the $C_{1,i}$ and $C_{2,i}$ components play almost identical roles. In fact we can view the encryption system as two parallel sub-systems linked at the C_0 component (and the corresponding private key component). A natural question is whether this redundancy can be eliminated (i.e. remove the second sub-system) to achieve better performance. While such a construction appears to be secure, our current proof (that utilizes a *non-interactive* assumption) relies in an essential way on having two parallel subsystems.

4.2 Proof Intuition

The most challenging aspect to providing a proof of our scheme naturally arises from the disjunctive capabilities of our system. In previous conjunctive systems such as that of Boneh and Waters [8] the authors proved security by moving through a sequence of hybrid games, in which an encryption of a vector $\vec{x}$ was changed component-by-component to the encryption of a vector $\vec{y}$. In these systems no inner product functionality was given, the adversary could only ask for predicate capabilities that performed a conjunctive search. In the security game the adversary could only ask for queries that did not match either $\vec{x}$ or $\vec{y}$, or queries that did not "look at" the components in which $\vec{x}$ and $\vec{y}$ differed. There, is was relatively straightforward to perform hybrid experiments over the components of $\vec{x}$ and $\vec{y}$ that differed, since the private keys given to the adversary did not "look at" these components.

In our proof an adversary will again try to distinguish between encryption of two vectors $\vec{x}$ and $\vec{y}$. However, in this case the adversary can query for a vector $\vec{v}$ such that both $\langle \vec{x}, \vec{v} \rangle = 0$ and $\langle \vec{y}, \vec{v} \rangle = 0$; i.e., this key should enable correct decryption in either case. This is a perfectly legitimate query since the inner product capability associated with predicate $\vec{v}$ should not be able to distinguish between $\vec{x}$ and $\vec{y}$. However, this means that we cannot use the same hybrid proof strategy as in previous schemes. For example, if we change just one component at a time, then the "hybrid" vector used in an intermediate step will likely not be orthogonal to $\vec{v}$ (and the adversary will be able to detect this). Therefore, we need an approach that will enable us to use hybrids in which entire vectors are changed in one step, instead of changing the vector component-by-component.

To get around this we take advantage of the fact that, as noted earlier, our encryption scheme has two parallel sub-systems. In our proof we will use a sequence of hybrids where some of the hybrid challenge ciphertexts will be encryptions of one vector in the first sub-system and a *different* vector in the second sub-system. (Note that such a ciphertext will be ill-formed, since any valid ciphertext will always use the same vector in each sub-system.) Let $(\vec{a}, \vec{b})$ denote an encryption of vector $\vec{a}$ in the first sub-system and $\vec{b}$ in the second su-system. To prove indistinguishability when encrypting to $\vec{x}$ (which corresponds to $(\vec{x}, \vec{x})$) and when encrypting to $\vec{y}$ (which corresponds to $(\vec{y}, \vec{y})$), we will prove indistinguishability of the following sequence of hybrid games:

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

Forming our hybrid proof in this structure allows us to use a simulator that will essentially be able to evaluate one sub-system, but not know 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 a particular vector or the zero vector. Details of our proof and further discussion are given in Appendix B.

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.

For improved readability throughout this section, we use boldface to denote vectors. Given a vector $\mathbf{x} \in \mathbb{Z}_N^\ell$, we denote by $f_\mathbf{x} : \mathbb{Z}_N^\ell \to \{0,1\}$ the function such that $f_\mathbf{x}(\mathbf{y}) = 1$ iff $\langle \mathbf{x}, \mathbf{y} \rangle = 0$. We define $\mathcal{F}^\ell \overset{\text{def}}{=} \{f_\mathbf{x} \mid \mathbf{x} \in \mathbb{Z}_N^\ell\}$.

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 system over $\mathcal{F}^2$. 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 $\mathbf{I} := (1, I)$ and output the secret key corresponding to the predicate $f_\mathbf{I}$. To encrypt a message M for the identity $I \in \mathbb{Z}_N$, set $\bar{\mathbf{I}} := (-I, 1)$ and encrypt the message using the encryption algorithm of the underlying inner product encryption scheme and the attribute $\bar{\mathbf{I}}$. Since $\left\langle \mathbf{I}, \bar{\mathbf{J}} \right\rangle = 0$ iff $I = J$, both correctness and security follow.

5.2 Hidden-Vector Encryption

Given a set Σ, let $\Sigma_\star = \Sigma \cup \{\star\}$. Hidden-vector encryption (HVE) [8] corresponds to a predicate encryption scheme for the class of predicates $\Phi^{\mathsf{hve}} = \{\phi^{\mathsf{hve}}_{(a_1,\ldots,a_\ell)} \mid a_1, \ldots, a_\ell \in \Sigma_\star\}$, 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 $(\mathsf{Setup}, \mathsf{GenKey}, \mathsf{Enc}, \mathsf{Dec})$ for $\mathcal{F}^{2\ell}$:

- 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 $\mathbf{A} = (A_1, \ldots, A_{2\ell})$ 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 $\mathsf{GenKey}_{SK}(f_\mathbf{A})$.
- 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 $\mathbf{X_r} = (X_1, \ldots, X_{2\ell})$ as follows:

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

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

To see that correctness holds, let $(a_1, \ldots, a_\ell)$, $\mathbf{A}$, $(x_1, \ldots, x_\ell)$, $\mathbf{r}$, and $\mathbf{X_r}$ be as above and note that

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

Conversely, security holds since

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

which is negligible.

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 $\bar{\Phi}^{\mathsf{hve}}=\{\bar{\phi}^{\mathsf{hve}}_{(a_1,\ldots,a_\ell)} \mid a_1,\ldots,a_\ell\in\Sigma\}$ 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

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

for $x\in\mathbb{Z}_N$. Given an inner product encryption scheme $(\mathsf{Setup},\mathsf{GenKey},\mathsf{Enc},\mathsf{Dec})$ for $\mathcal{F}^{d+1}$, 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 $p=a_dx^d+\cdots+a_0x^0$, set $\mathbf{p}:=(a_d,\ldots,a_0)$ and output the key obtained by running $\mathsf{GenKey}_{SK}(f_{\mathbf{p}})$.
- To encrypt a message M for the attribute $w\in\mathbb{Z}_N$, set $\mathbf{w}:=(w^d \bmod N,\ldots,w^0 \bmod N)$ and output the ciphertext $C\leftarrow\mathsf{Enc}_{PK}(\mathbf{w},M)$.

Since $p(w)=0$ iff $\langle\mathbf{p},\mathbf{w}\rangle=0$, 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 $\overline{\mathsf{OR}}^{\mathsf{eq}}_{a_1,a_2}(x_1,x_2)=1$ 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[3] that $p''(x_1,x_2) \neq 0$.

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) of each variable.

5.5 Exact Thresholds

We conclude with an application that relies directly on predicate encryption schemes for computing inner products (rather than relying on the polynomial-based scheme outlined in Section 5.3). Here, we consider the setting of "fuzzy IBE" [17], 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 [17] show a construction of a predicate encryption scheme (satisfying a weaker notion of security than Definition 2.2) 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 satisfy the stronger definition of security given here.) Namely, set $\Phi' = \{\phi'_S \mid S \subseteq A\}$ with $\phi'_S(S') = 1$ iff $|S \cap S'| = t$. Then, given any predicate encryption scheme over $\mathcal{F}^{\ell+1}$:

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

$$\begin{aligned} \text{for } 1 \leq i \leq \ell: \qquad & v_i = 1 \text{ iff } i \in S \\ & v_{\ell+1} = 1. \end{aligned}$$

  Then output the key obtained by running $\mathsf{GenKey}_{SK}(f_{\mathbf{v}})$.
- To encrypt a message M for the attribute $S' \subseteq A$, define a vector $\mathbf{v}'$ as follows:

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

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

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

[3] In general, the secret key may leak the value of r in which case the adversary will be able to find x_1, x_2 such that $\mathsf{AND}_{I_1,I_2}(x_1,x_1) \neq 1$ yet $p''(x_1,x_2) = 0$. Since, however, we consider the "selective" notion of security (where the adversary must commit to x_1, x_2 at the outset of the experiment), this is not a problem in our setting.

References

[1] M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi. Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. In *Advances in Cryptology — Crypto 2005*.

[2] M. Bellare, A. Boldyreva, and A. O'Neill. Deterministic and efficiently searchable encryption. In *CRYPTO*, pages 535–552, 2007.

[3] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *IEEE Symposium on Security and Privacy*, pages 321–334, 2007.

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

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

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

[7] D. Boneh, E.-J. Goh, and K. Nissim. Evaluating 2-DNF formulas on ciphertexts. In J. Kilian, editor, *Proc. 2nd Theory of Cryptography Conference*, volume 3378 of *LNCS*, pages 325–342. Springer, 2005.

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

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

[10] R. Canetti, S. Halevi, and J. Katz. A forward-secure public-key encryption scheme. In *Advances in Cryptology — Eurocrypt 2003*.

[11] M. Chase. Multi-authority attribute based encryption. In *TCC*, pages 515–534, 2007.

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

[13] C. Gentry. Practical identity-based encryption without random oracles. In *EUROCRYPT*, pages 445–464, 2006.

[14] O. Goldreich and R. Ostrovsky. Software protection and simulation by oblivious rams. *JACM*, 1996.

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

[16] R. Ostrovsky, A. Sahai, and B. Waters. Attribute-based encryption with non-monotonic access structures. In *ACM Conf. on Computer and Communications Security*, 2007.

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

[18] A. Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology — Crypto '84*, volume 196 of *LNCS*, pages 47–53. Springer-Verlag, 1984.

[19] E. Shi, J. Bethencourt, H. T.-H. Chan, D. X. Song, and A. Perrig. Multi-dimensional range queries over encrypted data. In *IEEE Symp. Security and Privacy*, pages 35–0–364, 2007.

[20] D. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In *Proceedings of the 2000 IEEE symposium on Security and Privacy (S&P 2000)*, 2000.

A Security Definition for Inner-Product Encryption

Here, we re-state Definition 2.2 in the particular setting of our main construction. Our main construction is a predicate-only scheme (we show how to extend it to a full-fledged predicate encryption scheme in Appendix C) 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

$$f_{\vec{x}}(\vec{y}) = 1 \Leftrightarrow \langle \vec{x}, \vec{y} \rangle = 0.$$

where $\Sigma = \mathbb{Z}_N^n$ and $\mathcal{F} = \{f_{\vec{v}} \mid \vec{v} \in \mathbb{Z}_N^n\}$ with $f_{\vec{v}}(v') = 1$ iff $\langle \vec{v}, \vec{v}' \rangle = 0$:

Definition A.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*,* $\langle \vec{v}_i, \vec{x} \rangle = 0$ *if and only if* $\langle \vec{v}_i, \vec{y} \rangle = 0$*. 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* $1/2$*.*

B Proof of Security

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

[4]Technically speaking, both Σ and $\mathcal{F}$ depend on the public parameters (since N is generated as part of PK), but we ignore this technicality. We remark also that we consider vectors of length n, the security parameter, for convenience only.

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

Throughout, we will refer to the experiment as described in Definition A.1. 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 A.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 of 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; informally speaking, 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 $\langle \vec{v}, \vec{x} \rangle \neq 0 \neq \langle \vec{v}, \vec{y} \rangle$.

B.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 does not appear to be any way 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 a random element $QR \in \mathbb{G}_{qr} \overset{\text{def}}{=} \mathbb{G}_q \times \mathbb{G}_r$: this can be done 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 A.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 (a combination of an encryption of $\vec{x}$ and $\vec{0}$).

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 make use of this element from the assumption 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, the simulator will be implicitly setting:

$$\begin{aligned} r_{1,i} &= r_{1,i}' + v_i \cdot (af_1' - abf_2') &(1)\\ r_{2,i} &= r_{2,i}' + a\, f_2'\, v_i, &(2) \end{aligned}$$

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 it computes:

$$\begin{aligned} K_{1,i} &= \left(g_p^a g_q\right)^{f_1' v_i} \cdot \left(g_p^{ab} Q_1\right)^{-f_2' v_i} \cdot g_p^{r_{1,i}'} \\ &= g_p^{(af_1' - abf_2')\cdot v_i + r_{1,i}'} \cdot g_q^{(f_1' - df_2')\cdot v_i} \end{aligned}$$

and

$$\begin{aligned} K_{2,i} &= \left(g_p^a g_q\right)^{f_2' v_i} \cdot g_p^{r_{2,i}'} \\ &= g_p^{af_2' v_i + r_{2,i}'} \cdot g_q^{f_2' v_i}. \end{aligned}$$

Now, to construct the K element for 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 $\sum_i r_{1,i} b x_i$. But because of how we chose $r_{1,i}$, we have that $\sum_i r_{1,i} b x_i = k(abf_1' - ab^2 f_2) + \sum_i r_{1,i}' x_i$ where $k = \langle \vec{v}, \vec{x} \rangle$. A similar equation holds for the terms arising out of the $h_{2,i}$ parts of K, and allows us to cancel out all the ab^2 terms that arise in K. Thus, we can compute K as follows:

Let $k = \langle \vec{v}, \vec{x} \rangle$. Finally, the simulator 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

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

(using Eqs. (1) and (2)), and thus K_p has the correct distribution.

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 $\{R'_{7,i}, R'_{8,i}\}$ refer to 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. ■

B.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 $\mathbb{G}_r$ or $\mathbb{G}_{qr}$ 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 A.1. We distinguish two cases, depending on whether $\langle \vec{v}, \vec{x} \rangle$ and $\langle \vec{v}, \vec{y} \rangle$ are both 0 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

$$\begin{aligned}
K_p &= \prod_i g_p^{-a f_1 v_i w_{1,i} - a f_2 v_i w_{2,i}} \cdot g_p^{-w_{1,i} \cdot r'_{1,i} - w_{2,i} \cdot r'_{2,i}} \cdot h_p^{-x_i \cdot r'_{1,i}} \cdot k_p^{-y_i \cdot r'_{2,i}} \\
&= \prod_i h_p^{-a f_1 x_i v_i} \cdot k_p^{-a f_2 y_i v_i} \cdot g_p^{-a f_1 v_i w_{1,i} - a f_2 v_i w_{2,i}} \cdot g_p^{-w_{1,i} \cdot r'_{1,i} - w_{2,i} \cdot r'_{2,i}} \cdot h_p^{-x_i \cdot r'_{1,i}} \cdot k_p^{-y_i \cdot r'_{2,i}},
\end{aligned}$$

using the fact that $\prod_i h_p^{-a f_1 x_i v_i} = h_p^{-a f_1 \cdot \sum_i x_i v_i} = 1 = \prod_i k_p^{-a f_2 y_i v_i}$ (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

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

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

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

as well as $f_1 = f'_1 - c_y \cdot df'_2$ 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

$$\begin{aligned} K_p &= g_p^{-abc_x f'_1} \cdot \prod_i g_p^{-af'_1 v_i w_{1,i} - af'_2 c_x v_i w_{2,i}} \cdot g_p^{abf'_2 c_y v_i w_{1,i}} \cdot g_p^{-w_{1,i} \cdot r'_{1,i} - w_{2,i} \cdot r'_{2,i}} \cdot h_p^{-x_i \cdot r'_{1,i}} \cdot k_p^{-y_i \cdot r'_{2,i}} \\ &= \prod_i h_p^{-ax_i v_i f'_1} \cdot g_p^{-af'_1 v_i w_{1,i} - af'_2 c_x v_i w_{2,i}} \cdot g_p^{abf'_2 c_y v_i w_{1,i}} \cdot (h_{1,i})^{r'_{1,i}} (h_{2,i})^{r'_{2,i}} \\ &= h_p^{c_x c_y abf'_2} \cdot h_p^{-c_x c_y abf'_2} \prod_i g_p^{-af'_2 c_x v_i w_{2,i}} \cdot g_p^{abf'_2 c_y v_i w_{1,i}} \cdot (h_{1,i})^{-r'_{1,i} - av_i f'_1} (h_{2,i})^{-r'_{2,i}} \\ &= \prod_i h_p^{x_i v_i c_y abf'_2} \cdot k_p^{-c_x y_i v_i af'_2} \cdot g_p^{-af'_2 c_x v_i w_{2,i}} \cdot g_p^{abf'_2 c_y v_i w_{1,i}} \cdot (h_{1,i})^{-r'_{1,i} - av_i f'_1} (h_{2,i})^{-r'_{2,i}} \\ &= \prod_i (h_{1,i})^{-r'_{1,i} - av_i f'_1 + abf'_2 c_y v_i} (h_{2,i})^{-r'_{2,i} - ac_x v_i f'_2} \;=\; \prod_i (h_{1,i})^{-r_{1,i}} (h_{2,i})^{-r_{2,i}}, \end{aligned}$$

and so K_p has the right 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 elements of $\mathbb{G}_r$ whose 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. ■

B.3 Completing the Proof

Our scheme is symmetric with respect to the roles of $h_{1,i}$ and $h_{2,i}$. Thus, as mentioned earlier, 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 B.1) that Game_1 and Game_2 are indistinguishable. This concludes the proof of our theorem.

C A Full-Fledged Predicate Encryption Scheme

In Section 4, we showed a construction of a *predicate-only* scheme. Here, we extend the previous scheme to obtain a full-fledged predicate encryption scheme in the sense of Definition 2.1. The construction follows.

$\mathsf{Setup}(1^n)$ 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. It also chooses random $\gamma \in \mathbb{Z}_p$ and $h \in \mathbb{G}_p$. The public parameters include $(N = pqr, \mathbb{G}, \mathbb{G}_T, \hat{e})$ along with:

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

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

$\mathsf{Enc}_{\mathrm{PK}}(\vec{x}, M)$ View M as an element of $\mathbb{G}_T$, and 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. It outputs the ciphertext

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

$\mathsf{GenKey}_{\mathrm{SK}}(\vec{v})$ 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 $f_1, f_2 \in \mathbb{Z}_q$, random $R_5 \in \mathbb{G}_r$, and random $Q_6 \in \mathbb{G}_q$. It then outputs

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

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

$$C' \cdot \hat{e}(C_1, K) \cdot \prod_{i=1}^n \hat{e}(C_{1,i}, K_{1,i}) \cdot \hat{e}(C_{2,i}, K_{2,i}).$$

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 $\mathrm{SK}_{\vec{v}}$ be as above. Then

$$\begin{aligned}
& C' \cdot \hat{e}(C_1, K) \cdot \prod_{i=1}^{n} \hat{e}(C_{1,i}, K_{1,i}) \cdot \hat{e}(C_{2,i}, K_{2,i}) \\
&\quad = \quad M \cdot P^s \cdot \hat{e}\left(g_p^s, \quad R_5 Q_6 h^{-\gamma} \prod_{i=1}^{n} h_{1,i}^{-r_{1,i}} h_{2,i}^{-r_{2,i}}\right) \\
&\qquad\quad \cdot \prod_{i=1}^{n} \hat{e}\left(H_{1,i}^s Q^{\alpha \cdot x_i} R_{3,i}, \quad g_p^{r_{1,i}} g_q^{f_1 \cdot v_i}\right) \cdot \hat{e}\left(H_{2,i}^s Q^{\beta \cdot x_i} R_{4,i}, \quad g_p^{r_{2,i}} g_q^{f_2 \cdot v_i}\right) \\
&\quad = \quad M \cdot P^s \cdot \hat{e}\left(g_p^s, \quad h^{-\gamma} \prod_{i=1}^{n} h_{1,i}^{-r_{1,i}} h_{2,i}^{-r_{2,i}}\right) \cdot \prod_{i=1}^{n} \hat{e}\left(h_{1,i}^s\, g_q^{\alpha \cdot x_i}, \quad g_p^{r_{1,i}} g_q^{f_1 \cdot v_i}\right) \cdot \hat{e}\left(h_{2,i}^s\, g_q^{\beta \cdot x_i}, \quad g_p^{r_{2,i}} g_q^{f_2 \cdot v_i}\right) \\
&\quad = \quad M \cdot P^s \cdot \hat{e}(g_p, h)^{-\gamma s} \cdot \prod_{i=1}^{n} \hat{e}(g_q, g_q)^{(\alpha f_1 + \beta f_2) x_i v_i} \quad = \quad M \cdot \hat{e}(g_q, g_q)^{(\alpha f_1 + \beta f_2)\langle \vec{x}, \vec{v} \rangle}.
\end{aligned}$$

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 B.1 and B.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 A.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

$$C = \left(C' = M_0 \cdot P^s, \quad C_1 = g_p^s, \quad \left\{C_{1,i} = H_{1,i}^s Q^{\alpha x_i} R_{3,i}, \quad C_{2,i} = H_{2,i}^s Q^{\beta x_i} R_{4,i}\right\}_{i=1}^{n}\right).$$

Game_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

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

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

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

Note that this exactly parallels Game_3 in the proof of Theorem B.1.

Game_4 and Game_5: These games are defined symmetrically to Game_2 and Game_3, as in the proof of Theorem B.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 mentioned earlier, *mutatis mutandis* from the proofs in Sections B.1 and B.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

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

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 $C_1 = g_p^s$, 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. ■