Predicate Privacy in Encryption Systems

Emily Shen[1], Elaine Shi[2], and Brent Waters[3⋆]

[1] MIT
eshen@csail.mit.edu
[2] CMU/PARC
eshi@parc.com
[3] UT Austin
bwaters@cs.utexas.edu

Abstract. *Predicate encryption* is a new encryption paradigm which gives a master secret key owner fine-grained control over access to encrypted data. The master secret key owner can generate secret key tokens corresponding to predicates. An encryption of data x can be evaluated using a secret token corresponding to a predicate f; the user learns whether the data satisfies the predicate, i.e., whether $f(x) = 1$.
Prior work on public-key predicate encryption has focused on the notion of data or plaintext privacy, the property that ciphertexts reveal no information about the encrypted data to an attacker other than what is inherently revealed by the tokens the attacker possesses. In this paper, we consider a new notion called *predicate privacy*, the property that tokens reveal no information about the encoded query predicate. Predicate privacy is inherently impossible to achieve in the public-key setting and has therefore received little attention in prior work. In this work, we consider predicate encryption in the symmetric-key setting and present a symmetric-key predicate encryption scheme which supports inner product queries. We prove that our scheme achieves both plaintext privacy and predicate privacy.

1 Introduction

In traditional public-key encryption, a user encrypts a message under a public key, and only the owner of the corresponding secret key can decrypt the ciphertext. In some applications, however, the user may wish to have more fine-grained control over what is revealed about the encrypted data. For example, in a medical context an administrative assistant might only be able to learn whether an encrypted record was generated at a certain clinic. *Predicate encryption* is a new encryption paradigm which allows for such fine-grained control over access to encrypted data. In a predicate encryption scheme, the owner of a master secret key can create and issue secret key *tokens* to other users. Tokens are associated with

⋆ Supported by NSF CNS-0524252, CNS-0716199; the U.S. Department of Homeland Security under Grant Award Number 2006-CS-001-000001.

predicates which can be evaluated over encrypted data. Specifically, an encryption of a data x can be evaluated using a token TK_f associated with a predicate f to learn whether $f(x) = 1$.

Prior work on public-key predicate encryption [7, 12, 1, 9, 19, 11, 28, 27] has focused on the security property that ciphertexts reveal no information about the underlying plaintext or data other than what is implied by the tokens in one's possession. More specifically, an adversary in possession of tokens $TK_{f_1}, \ldots, TK_{f_\ell}$ for predicates $f_1, \ldots, f_\ell$ learns no information about the underlying plaintext x other than the values of $f_1(x), \ldots, f_\ell(x)$[4]. We refer to the above property as plaintext or data privacy.

In this work, we consider a different dimension of predicate encryption – *predicate privacy*. In addition to protecting the privacy of plaintexts, we would like to protect the description of the predicates encoded by tokens. Informally, predicate privacy says that a token hides all information about the encoded predicate other than what is implied by the ciphertexts in one's possession. Unfortunately, predicate privacy is inherently impossible to achieve in the public-key setting. Since encryption does not require a secret key, an adversary can encrypt any plaintext of his choice and evaluate a token on the resulting ciphertext to learn whether the plaintext satisfies the predicate associated with the token. In this way, an adversary can gain information about the predicate encoded in a token. Therefore, it does not make sense to consider the notion of predicate privacy for predicate encryption in the public-key setting.

However, it is interesting to consider predicate privacy in the symmetric-key setting, in applications where we want to hide information about the predicate being tested from the party evaluating a token. For example, suppose a user Alice uses a remote storage service to back up her files. Alice wishes to protect the privacy of her files by encrypting them using her secret key before sending them to the server. (Only Alice possesses her secret key.) Later on, Alice may wish to retrieve all files satisfying a certain predicate. To do this, Alice can create a token (using her secret key) for this predicate and issue the token to the server. The server can then evaluate the predicate on the encrypted files and return those files which satisfy the predicate. We want to guarantee that the server learns nothing about the predicate it evaluates on Alice's behalf.

1.1 Our Results

In this paper, we present formal definitions of security for predicate encryption in the symmetric-key setting, for general classes of predicates. We present a symmetric-key predicate encryption scheme that achieves both plaintext privacy and predicate privacy. Our construction supports the class of predicates corresponding to the evaluation of inner products. We take the set of plaintexts to

[4] In some works the authors also distinguish an extra "payload message" M such that in the case that one of $f_1(x), \ldots, f_\ell(x)$ evaluates to 1, the adversary learns the payload message M. In our work we solely consider the predicate encryption system property where the evaluation reveals $f(x)$.

be $\Sigma = \mathbb{Z}_N^n$ and the class of predicates to be $\mathcal{F} = \{f_{\boldsymbol{v}} | \boldsymbol{v} \in \mathbb{Z}_N^n\}$ where $f_{\boldsymbol{v}}(\boldsymbol{x}) = 1$ iff $\langle \boldsymbol{v}, \boldsymbol{x} \rangle = 0$, where $\langle \boldsymbol{v}, \boldsymbol{x} \rangle$ denotes the inner product $\sum_{i=1}^{n} v_i \cdot x_i \mod N$ of vectors $\boldsymbol{v}$ and $\boldsymbol{x}$. Our construction is based on the KSW construction [21], which uses bilinear groups whose order is the product of three primes. Our construction uses groups whose order is the product of four primes. Our complexity assumptions have all been introduced in prior work but for the case of groups whose order is the product of fewer than four primes.

Why Inner Product Queries? An important goal in predicate encryption is to support complex, expressive queries. Prior work has focused on achieving more expressive schemes, with the most expressive scheme to date being that of Katz, Sahai and Waters [21]. The KSW scheme supports inner product queries, which are strictly more expressive than conjunctive queries and, as shown in [21], imply conjunctions, disjunctions, CNF/DNF formulas, polynomial evaluation, and exact thresholds. Therefore, our goal in this work is to construct a symmetric-key predicate encryption scheme that supports inner product queries.

1.2 Related Work

Public-Key Predicate Encryption. The earliest examples of public-key predicate encryption are *anonymous identity-based encryption* (A-IBE) schemes with *keyword search* (which corresponds to an equality predicate) [7, 12, 1, 9]. Since then, more expressive schemes such as those supporting conjunctive queries [19, 11, 28] and multi-dimensional range queries [27] have been proposed. The most expressive scheme known to date is due to Katz, Sahai and Waters [21] and supports inner-product queries. As explained above, the KSW scheme is strictly more expressive than previously proposed predicate encryption schemes.

Searchable Encrypted Databases. A related line of research is secure searching on outsourced encrypted databases. The problem was considered by Goldreich and Ostrovsky [22, 18] when cast as a problem of oblivious RAM, and they provided general solutions. Song, Wagner, and Perrig [29] later gave more efficient solutions for equality searches, but made a tradeoff of letting a storage server learn the access pattern of a user. Curtmola et al. [17] considered stronger security definitions in a similar setting. While we do not directly address searchable encrypted databases in this work, our predicate encryption solution allows for more complex queries to be made in this particular application.

Identity-Based Encryption and Attribute-Based Encryption. Identity-based encryption (IBE) [26, 8, 16] can be viewed as a special, more limited, case of predicate encryption for the class of equality tests. In attribute-based encryption (ABE) [25, 20, 3, 24, 15, 23], a user can receive a capability representing an access control policy over the attributes of an encrypted record.

In both IBE and ABE schemes, the identity or attributes are not hidden in the ciphertext. In fact, access to the encrypted data itself is inherently "all-or-nothing." The important distinction between these systems and the ones we

consider is that they only hide a "payload message" M. In particular, the ciphertext is associated with a payload message M and some extra structure x (e.g., the "identity" or set of attributes associated with the ciphertext). The security guarantee of these systems is that M remains hidden as along as the attacker does not have a secret key associated with a predicate function f such that $f(x) = 1$; however, there is no guarantee about hiding the structure of x, which in general might be leaked to the attacker. One advantage, however, is that this relaxation might allow for more expressive access predicates.

2 Definitions

In this section, we formally define symmetric-key predicate encryption and its security. For simplicity, we consider the *predicate-only* variant, in which evaluating a token on a ciphertext outputs a bit indicating whether the encrypted plaintext satisfies the predicate corresponding to the token. We note that a predicate-only scheme can easily be extended to obtain a full-fledged predicate encryption scheme, in which evaluating a token on a ciphertext outputs the encrypted plaintext if the plaintext satisfies the predicate corresponding to the token, using techniques from prior work such as [11, 27, 21].

We give definitions for the general case of an arbitrary set of plaintexts Σ and an arbitrary set of predicates $\mathcal{F}$. Our construction in Section 4 will be for the specific case of $\Sigma = \mathbb{Z}_N^n$ and $\mathcal{F} = \{f_{\boldsymbol{v}} | \boldsymbol{v} \in \mathbb{Z}_N^n\}$ with $f_{\boldsymbol{x}}(\boldsymbol{v}) = 1$ iff $\langle \boldsymbol{x}, \boldsymbol{v} \rangle = 0 \mod N$, where $\langle \boldsymbol{x}, \boldsymbol{v} \rangle$ denotes the inner product $\sum_{i=1}^{n} x_i \cdot v_i \mod N$ of vectors $\boldsymbol{x}$ and $\boldsymbol{v}$. We follow the notation of [21].

2.1 Symmetric-Key Predicate-Only Encryption

Let Σ denote a finite set of plaintexts, and let $\mathcal{F}$ denote a finite set of predicates $f : \Sigma \to \{0, 1\}$. We say that $x \in \Sigma$ satisfies a predicate f if $f(x) = 1$.

Definition 1 (Symmetric-Key Predicate-Only Encryption Scheme). *A* symmetric-key predicate-only encryption scheme *for the class of predicates $\mathcal{F}$ over the set of attributes Σ consists of the following probabilistic polynomial time (PPT) algorithms.*

Setup(1^λ)**:** *Takes as input a security parameter 1^λ and outputs a secret key SK.*

Encrypt(SK, x)**:** *Takes as input a secret key SK and a plaintext $x \in \Sigma$ and outputs a ciphertext CT.*

GenToken(SK, f)**:** *Takes as input a secret key SK and a description of a predicate $f \in \mathcal{F}$ and outputs a token TK_f.*

Query(TK_f, CT)**:** *Takes as input a token TK_f for a predicate f and a ciphertext CT. It outputs either 0 or 1, indicating the value of the predicate f evaluated on the underlying plaintext.*

Correctness. For correctness, we require the following condition. For all λ, all $x \in \Sigma$, and all $f \in \mathcal{F}$, letting $SK \leftarrow Setup(1^\lambda)$, $TK_f \leftarrow GenToken(SK, f)$, and $CT \leftarrow Encrypt(SK, x)$,

- *If $f(x) = 1$, then $Query(TK_f, CT) = 1$.*
- *If $f(x) = 0$, then $\Pr[Query(TK_f, CT) = 0] > 1 - \epsilon(\lambda)$ where ϵ is a negligible function.*

2.2 Security Definitions

We now give formal definitions of security for a symmetric-key predicate-only encryption scheme. We first define *full security*, which, roughly speaking, says that given a set of tokens for predicates $f_1, \ldots, f_k$ and a set of encryptions of plaintexts $x_1, \ldots, x_\ell$, an adversary $\mathcal{A}$ gains no information about any of the predicates $f_1, \ldots, f_k$ or the plaintexts $x_1, \ldots, x_\ell$ (other than the value of each of the predicates evaluated on each of the plaintexts).

However, the full security notion turns out to be difficult to work with in our proofs of security. Therefore, we introduce a second security notion called *single challenge security*, which resembles the security notions used in previous work such as [11, 21]. As we show later, full security implies single challenge security, and, for the specific case of inner product predicates, single challenge security implies full security in the sense that, given a single challenge secure scheme for inner product predicates over $\Sigma = \mathbb{Z}_N^{2n}$, we can construct a fully secure scheme for inner product predicates over $\Sigma = \mathbb{Z}_N^n$. Therefore, for our construction it suffices to consider the single challenge security definition. To prove the security of our construction, we will use the *selective* relaxation of single challenge security. The notion of selective security was first introduced by [13] and has been used widely in the literature [13, 14, 5, 11, 12, 27].

Full Security We define full security of a symmetric-key predicate-only encryption scheme using the following game between an adversary $\mathcal{A}$ and a challenger.

Setup: The challenger runs $Setup(1^\lambda)$ and keeps SK to itself. The challenger picks a random bit b.

Queries: $\mathcal{A}$ adaptively issues queries, where each query is of one of two types:

- Ciphertext query. On the jth ciphertext query, $\mathcal{A}$ outputs a bit $t = 0$ (indicating a ciphertext query) and two plaintexts $x_{j,0}, x_{j,1} \in \Sigma$. The challenger responds with $Encrypt(SK, x_{j,\mathsf{b}})$.
- Token query. On the ith token query, $\mathcal{A}$ outputs a bit $t = 1$ (indicating a token query) and descriptions of two predicates $f_{i,0}, f_{i,1} \in \mathcal{F}$. The challenger responds with $GenToken(SK, f_{i,\mathsf{b}})$.

$\mathcal{A}$'s queries are subject to the restriction that, for all ciphertext queries $(x_{j,0}, x_{j,1})$ and all predicate queries $(f_{i,0}, f_{i,1})$, $f_{i,0}(x_{j,0}) = f_{i,1}(x_{j,1})$.

Guess: $\mathcal{A}$ outputs a guess b' of b.

The advantage of $\mathcal{A}$ is defined as $\mathrm{Adv}_{\mathcal{A}} = \left|\Pr[\mathsf{b}' = \mathsf{b}] - \frac{1}{2}\right|$.

Definition 2 (Full Security). *A symmetric-key predicate-only encryption scheme is* fully secure *if, for all PPT adversaries $\mathcal{A}$, the advantage of $\mathcal{A}$ in winning the above game is negligible in λ.*

Single Challenge Security In order to prove the security of our construction, we will need to introduce a second security definition, which we refer to as *single challenge security*. Whereas in the full security game, each of the adversary's queries is considered part of the challenge, in the single challenge security game, the challenge consists of only one pair of plaintexts or predicates. The single challenge security game resembles security games used previously in the IBE and predicate encryption literature. The game proceeds as follows.

Setup: The challenger runs $\mathit{Setup}(1^\lambda)$ and keeps SK to itself.
Query Phase 1: $\mathcal{A}$ adaptively issues queries, where each query is of one of two types:
- Ciphertext query. On the jth ciphertext query, $\mathcal{A}$ outputs a bit $t = 0$ (indicating a ciphertext query) and a plaintext x_j. The challenger responds with $\mathit{Encrypt}(SK, x_j)$.
- Token query. On the jth token query, $\mathcal{A}$ outputs a bit $t = 1$ (indicating a token query) and a description of a predicate f_j. The challenger responds with $\mathit{GenToken}(SK, f_j)$.

Challenge: $\mathcal{A}$ outputs a request for one of the following:
- Ciphertext challenge. $\mathcal{A}$ outputs a bit $t = 0$ (indicating a ciphertext challenge) and two plaintexts x_0^* and x_1^* such that, for all previous token queries f_j, $f_j(x_0^*) = f_j(x_1^*)$. The challenger picks a random bit b and responds with $\mathit{Encrypt}(SK, x_\mathsf{b}^*)$.
- Token challenge. $\mathcal{A}$ outputs a bit $t = 1$ (indicating a token challenge) and descriptions of two predicates f_0^* and f_1^* such that, for all previous ciphertext queries x_j, $f_0^*(x_j) = f_1^*(x_j)$. The challenger picks a random bit b and responds with $\mathit{GenToken}(SK, f_\mathsf{b}^*)$.

Query Phase 2: $\mathcal{A}$ adaptively issues additional queries as in Query Phase 1, subject to the same restriction with respect to the challenge as above.
Guess: $\mathcal{A}$ outputs a guess b' of b.

The advantage of $\mathcal{A}$ is defined as $\mathrm{Adv}_\mathcal{A} = \left|\Pr[\mathsf{b}' = \mathsf{b}] - \frac{1}{2}\right|$.

Definition 3 (Single Challenge Security). *A symmetric-key predicate-only encryption scheme is* single challenge secure *if, for all PPT adversaries $\mathcal{A}$, the advantage of $\mathcal{A}$ in winning the above game is negligible in λ.*

Selective Single Challenge Security. We will need to use the *selective* variant of single challenge security, defined below. The notion of selective security was first introduced by [13] and has been used previously by [13, 14, 5, 11, 12, 27].

Definition 4 (Selective Single Challenge Security). *In the selective single challenge security game, the adversary $\mathcal{A}$ outputs the challenge strings at the*

start of the game during an ***Init*** *phase (instead of during a* ***Challenge*** *phase). The rest of the game proceeds in the same way as in the single challenge security game. We say that a symmetric-key predicate-only encryption scheme is* selective single challenge secure *if, for all PPT adversaries $\mathcal{A}$, the advantage of $\mathcal{A}$ in winning the selective single challenge game is negligible in λ.*

For our proofs of security, it will be useful to define separate notions of plaintext privacy and predicate privacy, which correspond to a ciphertext challenge and a token challenge, respectively, in the selective single challenge security game.

Definition 5 (Plaintext Privacy). *A symmetric-key predicate-only encryption scheme has* selective single challenge plaintext privacy *(*plaintext privacy*, for short) if, for all PPT adversaries $\mathcal{A}$, the advantage of $\mathcal{A}$ in winning the selective single challenge game for a ciphertext challenge is negligible in λ.*

Definition 6 (Predicate Privacy). *A symmetric-key predicate-only encryption scheme has* selective single challenge predicate privacy *(*predicate privacy*, for short) if, for all PPT adversaries $\mathcal{A}$, the advantage of $\mathcal{A}$ in winning the selective single challenge game for a token challenge is negligible in λ.*

We note that plaintext privacy and predicate privacy, together, are equivalent to selective single challenge security.

Relationship Between Single Challenge Security and Full Security It is useful to consider the relationship between the security definitions introduced above. The full security notion implies single challenge security. For the specific case of inner product query predicates, a single challenge secure scheme for vectors of length $2n$ can be used to construct a fully secure scheme for vectors of length n. Therefore, we consider single challenge security to be a sufficiently strong notion of security for our construction.

These relationships are stated formally in the following theorems.

Theorem 1. *If a symmetric-key predicate-only encryption scheme is fully secure, then it is single challenge secure.*

Proof. Suppose an adversary $\mathcal{A}$ wins the single challenge security game with advantage ϵ. We can define an adversary $\mathcal{B}$ that wins the full security game with advantage ϵ as follows. When $\mathcal{A}$ makes a ciphertext query $\boldsymbol{x}$, $\mathcal{B}$ in turn makes the ciphertext query $(\boldsymbol{x}, \boldsymbol{x})$ to $\mathcal{B}$'s challenger and responds to $\mathcal{A}$ with the ciphertext it receives. Similarly, when $\mathcal{A}$ makes a token query $\boldsymbol{v}$, $\mathcal{B}$ in turn makes the token query $(\boldsymbol{v}, \boldsymbol{v})$ to $\mathcal{B}$'s challenger and responds to $\mathcal{A}$ with the token it receives. When $\mathcal{A}$ issues its challenge request, $\mathcal{B}$ outputs the challenge request as a query to its challenger and responds to $\mathcal{A}$ with the answer it receives. $\mathcal{B}$ outputs the same guess b' as $\mathcal{A}$ does. It is clear that all of $\mathcal{B}$'s responses to $\mathcal{A}$ are properly constructed, and $\mathcal{B}$ wins the full security game with the same advantage ϵ with which $\mathcal{A}$ wins the single challenge security game.

Theorem 2. *Let* SCHEME_{2n} *denote a single challenge secure symmetric-key predicate-only encryption scheme for inner product queries, where plaintext and predicate vectors have length* $2n$*. Then* SCHEME_{2n} *can be used to construct a fully secure symmetric-key predicate-only encryption scheme* SCHEME_n *for inner product queries, where plaintext and predicate vectors have length* n*.*

The proof of this theorem is deferred to Appendix A.

3 Background and Complexity Assumptions

Our symmetric-key predicate encryption scheme uses bilinear groups of composite order, first introduced by [10]. While the public-key predicate encryption scheme of [21] uses bilinear groups whose order is the product of three distinct primes, we use bilinear groups whose order is the product of four distinct primes.

We briefly review some facts about bilinear groups and then state the assumptions we use to prove security of our construction.

3.1 Bilinear Groups of Composite Order

Let $\mathcal{G}$ denote a group generator algorithm that takes as input a security parameter 1^λ and outputs a tuple $(p, q, r, s, \mathbb{G}, \mathbb{G}_T, e)$ where p, q, r, s are distinct primes, $\mathbb{G}$ and $\mathbb{G}_T$ are two cyclic groups of order $N = pqrs$, and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ satisfies the following properties:

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

We assume that group operations in $\mathbb{G}$ and $\mathbb{G}_T$ as well as the bilinear map e can be computed in time polynomial in λ.

We use the notation $\mathbb{G}_p, \mathbb{G}_q, \mathbb{G}_r, \mathbb{G}_s$ to denote the subgroups of $\mathbb{G}$ having order p, q, r, s, respectively.

We will use the following facts about bilinear groups of composite order. Although these facts are stated in terms of $\mathbb{G}_p$ and $\mathbb{G}_q$, similar facts hold in general for distinct subgroups of a composite order bilinear group.

- Let $a_p \in \mathbb{G}_p, b_q \in \mathbb{G}_q$ denote two elements from distinct subgroups. Then $e(a_p, b_q) = 1$.
- Let $\mathbb{G}_{pq} = \mathbb{G}_p \times \mathbb{G}_q$, $a, b \in \mathbb{G}_{pq}$. a and b can be rewritten (uniquely) as $a = a_p a_q$, $b = b_p b_q$, where $a_p, b_q \in \mathbb{G}_p$, and $a_q, b_q \in \mathbb{G}_q$. Then $e(a, b) = e(a_p, b_p)e(a_q, b_q)$.

3.2 Our Assumptions

The security of our symmetric-key predicate-only encryption scheme relies on three assumptions. All of these assumptions have been introduced previously but in groups whose order is the product of at most three distinct primes. Specifically,

Assumption 1 involves 3 subgroups, C3DH involves 2 subgroups, and DL involves 1 subgroup. We assume that these assumptions hold when the relevant subgroups are contained in a larger group whose order is the product of four distinct primes. Note that the naming of subgroups is not significant in our assumptions; that is, the assumptions are the same if the subgroups are renamed.

Assumption 1 We use Assumption 1 of KSW [21], which was used for bilinear groups whose order is the product of three distinct primes. We restate the assumption in the context of a bilinear group whose order is the product of four distinct primes.

Let $\mathcal{G}$ be a group generator algorithm as above. Run $\mathcal{G}(1^\lambda)$ to obtain $(p, q, r, s, \mathbb{G}, \mathbb{G}_T, e)$. Let $N = pqrs$ and let g_p, g_q, g_r, g_s be random generators of $\mathbb{G}_p, \mathbb{G}_q, \mathbb{G}_r, \mathbb{G}_s$, respectively. Choose random $Q_1, Q_2, Q_3 \in \mathbb{G}_q$, random R_1, R_2, $R_3 \in \mathbb{G}_r$, random $a, b, c \in \mathbb{Z}_p$, and a random bit b. If $\mathsf{b} = 0$, let $T = g_p^{b^2 c} R_3$; if $\mathsf{b} = 1$, let $T = g_p^{b^2 c} Q_3 R_3$. Give the adversary $\mathcal{A}$ the description of the bilinear group, $(N, \mathbb{G}, \mathbb{G}_T, e)$, along with the following values:

$$\Big(g_p,\ \ g_r,\ \ g_s,\ \ g_q R_1,\ \ g_p^b,\ \ g_p^{b^2},\ \ g_p^a g_q,\ \ g_p^{ab} Q_1,\ \ g_p^c,\ \ g_p^{bc} Q_2 R_2,\ \ T\Big)$$

The adversary $\mathcal{A}$ outputs a guess b' of b. The advantage of $\mathcal{A}$ is defined as $\mathrm{Adv}_{\mathcal{A}} = \left|\Pr[\mathsf{b}' = \mathsf{b}] - \frac{1}{2}\right|$.

Definition 7. *We say that $\mathcal{G}$ satisfies Assumption 1 if, for all PPT algorithms $\mathcal{A}$, the advantage of $\mathcal{A}$ in winning the above game is negligible in the security parameter λ.*

We note that Assumption 1 implies the hardness of finding a non-trivial factor of N.

Generalized 3-Party Diffie-Hellman Assumption (C3DH). We use the composite 3-party Diffie-Hellman assumption first introduced by [11]. We restate the assumption in the context of a bilinear group whose order is the product of four distinct primes.

Let $\mathcal{G}$ be a group generator algorithm as above. Run $\mathcal{G}(1^\lambda)$ to obtain $(p, q, r, s, \mathbb{G}, \mathbb{G}_T, e)$. Let $N = pqrs$ and let g_p, g_q, g_r, g_s be random generators of $\mathbb{G}_p, \mathbb{G}_q, \mathbb{G}_r, \mathbb{G}_s$, respectively. Choose random $R_1, R_2, R_3 \in \mathbb{G}_r$, random $a, b, c \in \mathbb{Z}_N$, and a random bit b. If $\mathsf{b} = 0$, let $T = g_p^c \cdot R_3$; if $\mathsf{b} = 1$, let T be a random element in $\mathbb{G}_{pr} = \mathbb{G}_p \times \mathbb{G}_r$. Give the adversary $\mathcal{A}$ the description of the bilinear group, $(N, \mathbb{G}, \mathbb{G}_T, e)$, along with the following values:

$$\big(g_p,\ \ g_q,\ \ g_r,\ \ g_s,\ \ g_p^a,\ \ g_p^b,\ \ g_p^{ab} \cdot R_1,\ \ g_p^{abc} \cdot R_2,\ \ T\big)$$

The adversary $\mathcal{A}$ outputs a guess b' of b. The advantage of $\mathcal{A}$ is defined as $\mathrm{Adv}_{\mathcal{A}} = \left|\Pr[\mathsf{b}' = \mathsf{b}] - \frac{1}{2}\right|$.

Definition 8. *We say that $\mathcal{G}$ satisfies the C3DH assumption if for all PPT algorithms $\mathcal{A}$, the advantage of $\mathcal{A}$ in winning the above game is negligible in the security parameter λ.*

We note that the C3DH assumption implies the hardness of finding a non-trivial factor N.

Decisional Linear assumption (DLinear). We use the Decisional Linear assumption introduced by [6]. We restate the assumption in the context of a bilinear group whose order is the product of four distinct primes.

Let $\mathcal{G}$ be a group generator algorithm as above. Run $\mathcal{G}(1^\lambda)$ to obtain $(p, q, r, s, \mathbb{G}, \mathbb{G}_T, e)$. Let $N = pqrs$ and let g_p, g_q, g_r, g_s be random generators of $\mathbb{G}_p, \mathbb{G}_q, \mathbb{G}_r, \mathbb{G}_s$, respectively. Choose random $z_1, z_2, z_3, z_4 \in \mathbb{Z}_p$ and a random bit b. If $\mathsf{b} = 0$, let $Z = g_p^{z_3+z_4}$; if $\mathsf{b} = 1$, let Z be a random element in $\mathbb{G}_p$. Give the adversary $\mathcal{A}$ the description of the bilinear group, $(N, \mathbb{G}, \mathbb{G}_T, e)$, along with the following values:

$$(g_p,\ \ g_q,\ \ g_r,\ \ g_s,\ \ g_p^{z_1},\ \ g_p^{z_2},\ \ g_p^{z_1 z_3},\ \ g_p^{z_2 z_4},\ \ Z)$$

The adversary $\mathcal{A}$ outputs a guess b' of b. The advantage of $\mathcal{A}$ is defined as $\mathrm{Adv}_{\mathcal{A}} = \left|\Pr[\mathsf{b}' = \mathsf{b}] - \frac{1}{2}\right|$.

Definition 9. *We say that $\mathcal{G}$ satisfies the DLinear assumption if for all PPT algorithms $\mathcal{A}$, the advantage of $\mathcal{A}$ in winning the above game is negligible in the security parameter n.*

4 Construction

Our goal is to construct a symmetric-key predicate encryption scheme supporting inner product queries that has both plaintext privacy and predicate privacy. The KSW construction [21] is a public-key predicate encryption scheme supporting inner product queries that has plaintext privacy. A natural first attempt might be to convert the KSW scheme into a symmetric-key scheme simply by withholding the public key. Such a scheme would immediately inherit plaintext privacy from the KSW construction. However, it is difficult to prove the predicate privacy of such a scheme. Our primary challenge is to achieve predicate privacy.

To achieve predicate privacy, we use the observation that, for inner product queries, ciphertexts and tokens play symmetric roles in the scheme and the security definitions. In particular, a token and a ciphertext each encode a vector in $\mathbb{Z}_N^n$, and the inner product $\langle \boldsymbol{x}, \boldsymbol{v} \rangle$ is commutative. Furthermore, for inner products, ciphertexts and tokens have symmetric roles in the security definitions. One way to interpret this observation is to view a ciphertext as an encryption of a plaintext vector and a token as an encryption of a predicate vector.

Based on this observation, our general approach is to start from a construction that resembles the KSW construction, so that we can prove plaintext privacy in a relatively straightforward manner. We then show through a series of modifications to our construction that it is indistinguishable from one in which ciphertexts and tokens are formed symmetrically. Using this symmetry, we can leverage the plaintext privacy proven for our main construction to achieve predicate privacy as well. Taken all together, the "native" formation of our system gives us

plaintext privacy by a KSW type of approach, and the indistinguishability of our construction from one in which ciphertexts and tokens are symmetrically formed implies that our construction also has predicate privacy.

4.1 A Symmetric-Key Predicate Encryption Scheme

Our main construction is a symmetric-key predicate-only encryption scheme supporting inner product queries. We take the class of plaintexts to be $\Sigma = \mathbb{Z}_N^n$ and the class of predicates to be $\mathcal{F} = \{f_{\boldsymbol{v}} | \boldsymbol{v} \in \mathbb{Z}_N^n\}$ with $f_{\boldsymbol{x}}(\boldsymbol{v}) = 1$ iff $\langle \boldsymbol{x}, \boldsymbol{v} \rangle = 0 \mod N$.

We now describe our construction in detail.

Setup(1^λ)**:** The setup algorithm runs $\mathcal{G}(1^\lambda)$ to obtain $(p, q, r, s, \mathbb{G}, \mathbb{G}_T, e)$ with $\mathbb{G} = \mathbb{G}_p \times \mathbb{G}_q \times \mathbb{G}_r \times \mathbb{G}_s$. Next it picks generators g_p, g_q, g_r, g_s of $\mathbb{G}_p, \mathbb{G}_q, \mathbb{G}_r, \mathbb{G}_s$, respectively. It chooses $h_{1,i}, h_{2,i}, u_{1,i}, u_{2,i} \in \mathbb{G}_p$ uniformly at random for $i = 1$ to n. The secret key is

$$SK = \left(g_p,\ g_q,\ g_r,\ g_s,\ \{h_{1,i}, h_{2,i}, u_{1,i}, u_{2,i}\}_{i=1}^n\right).$$

Encrypt$(SK, \boldsymbol{x})$**:** Let $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{Z}_N^n$. The encryption algorithm chooses random $y, z, \alpha, \beta \in \mathbb{Z}_N$, random $S, S_0 \in \mathbb{G}_s$, and random $R_{1,i}, R_{2,i} \in \mathbb{G}_r$ for $i = 1$ to n. It outputs the ciphertext

$$CT = \begin{pmatrix} C = S \cdot g_p^y, & C_0 = S_0 \cdot g_p^z, \\ \left\{C_{1,i} = h_{1,i}^y \cdot u_{1,i}^z \cdot g_q^{\alpha x_i} \cdot R_{1,i}, & C_{2,i} = h_{2,i}^y \cdot u_{2,i}^z \cdot g_q^{\beta x_i} \cdot R_{2,i}\right\}_{i=1}^n \end{pmatrix}.$$

GenToken$(SK, \boldsymbol{v})$**:** Let $\boldsymbol{v} = (v_1, \ldots, v_n) \in \mathbb{Z}_N^n$. The token generation algorithm chooses random $f_1, f_2 \in \mathbb{Z}_N$, random $r_{1,i}, r_{2,i} \in \mathbb{Z}_N$ for $i = 1$ to n, random $R, R_0 \in \mathbb{G}_r$, and random $S_{1,i}, S_{2,i} \in \mathbb{G}_s$ for $i = 1$ to n. It outputs the token

$$TK_{\boldsymbol{v}} = \begin{pmatrix} K = R \cdot \prod_{i=1}^n h_{1,i}^{-r_{1,i}} \cdot h_{2,i}^{-r_{2,i}}, & K_0 = R_0 \cdot \prod_{i=1}^n u_{1,i}^{-r_{1,i}} \cdot u_{2,i}^{-r_{2,i}}, \\ \left\{K_{1,i} = g_p^{r_{1,i}} \cdot g_q^{f_1 v_i} \cdot S_{1,i}, & K_{2,i} = g_p^{r_{2,i}} \cdot g_q^{f_2 v_i} \cdot S_{2,i}\right\}_{i=1}^n \end{pmatrix}.$$

Query$(TK_{\boldsymbol{v}}, CT)$: Let $CT = (C, C_0, \{C_{1,i}, C_{2,i}\}_{i=1}^n)$ and $TK_{\boldsymbol{v}} = (K, K_0, \{K_{1,i}, K_{2,i}\}_{i=1}^n)$ as above. The query algorithm outputs 1 iff

$$e(C, K) \cdot e(C_0, K_0) \cdot \prod_{i=1}^n e(C_{1,i}, K_{1,i}) \cdot e(C_{2,i}, K_{2,i}) \stackrel{?}{=} 1.$$

Correctness. Let CT and $TK_{\boldsymbol{v}}$ be as above. Then

$$
\begin{aligned}
& e(C,K)\cdot e(C_0,K_0)\cdot\prod_{i=1}^{n} e(C_{1,i},K_{1,i})\cdot e(C_{2,i},K_{2,i}) \\
&= e(S\cdot g_p^{y},\ R\cdot\prod_{i=1}^{n} h_{1,i}^{-r_{1,i}}\cdot h_{2,i}^{-r_{2,i}})\cdot e(S_0\cdot g_p^{z},\ R_0\cdot\prod_{i=1}^{n} u_{1,i}^{-r_{1,i}}\cdot u_{2,i}^{-r_{2,i}}) \\
&\quad\cdot\prod_{i=1}^{n} e(h_{1,i}^{y}\cdot u_{1,i}^{z}\cdot g_q^{\alpha x_i}\cdot R_{1,i},\ g_p^{r_{1,i}}\cdot g_q^{f_1 v_i}\cdot S_{1,i}) \\
&\quad\cdot e(h_{2,i}^{y}\cdot u_{2,i}^{z}\cdot g_q^{\beta x_i}\cdot R_{2,i},\ g_p^{r_{2,i}}\cdot g_q^{f_2 v_i}\cdot S_{2,i}) \\
&= e(g_p^{y},\ \prod_{i=1}^{n} h_{1,i}^{-r_{1,i}}\cdot h_{2,i}^{-r_{2,i}})\cdot e(g_p^{z},\ \prod_{i=1}^{n} u_{1,i}^{-r_{1,i}}\cdot u_{2,i}^{-r_{2,i}}) \\
&\quad\cdot\prod_{i=1}^{n} e(h_{1,i}^{y}\cdot u_{1,i}^{z}\cdot g_q^{\alpha x_i},\ g_p^{r_{1,i}}\cdot g_q^{f_1 v_i})\cdot e(h_{2,i}^{y}\cdot u_{2,i}^{z}\cdot g_q^{\beta x_i},\ g_p^{r_{2,i}}\cdot g_q^{f_2 v_i}) \\
&= \prod_{i=1}^{n} e(g_q,g_q)^{(\alpha f_1+\beta f_2)x_i v_i} = e(g_q,g_q)^{(\alpha f_1+\beta f_2 \mod q)\langle \boldsymbol{x},\boldsymbol{v}\rangle}
\end{aligned}
$$

If $\langle \boldsymbol{x},\boldsymbol{v}\rangle = 0 \mod N$, then the above expression evaluates to 1. If $\langle \boldsymbol{x},\boldsymbol{v}\rangle \neq 0 \mod N$, then there are two cases. If $\langle \boldsymbol{x},\boldsymbol{v}\rangle = 0 \mod q$, then the above expression evaluates to 1; however, this case would reveal a non-trivial factor of N and, therefore, this case occurs with negligible probability. If $\langle \boldsymbol{x},\boldsymbol{v}\rangle \neq 0 \mod q$, then with all but negligible probability $\alpha f_1+\beta f_2 \neq 0 \mod q$ and the above expression does not evaluate to 1.

4.2 Discussion

To understand our construction, it is useful to examine the role of each of the subgroups $\mathbb{G}_p, \mathbb{G}_q, \mathbb{G}_r, \mathbb{G}_s$.

The $\mathbb{G}_q$ subgroup is used to encode the plaintext vector $\boldsymbol{x}$ in the $C_{1,i}$ and $C_{2,i}$ terms of the ciphertext and the predicate vector $\boldsymbol{v}$ in the $K_{1,i}$ and $K_{2,i}$ terms of the token. When a token for $\boldsymbol{v}$ is applied to an encryption of $\boldsymbol{x}$, the computation of the inner product $\langle \boldsymbol{x},\boldsymbol{v}\rangle$ is evaluated in the exponent of the $\mathbb{G}_q$ subgroup.

The $\mathbb{G}_p$ subgroup is used to prevent an adversary from manipulating components of either a ciphertext or a token and then evaluating a query on the improperly formed inputs. The $\mathbb{G}_p$ subgroup encodes an equation which will evaluate to 0 in the exponent if the inputs to the query algorithm are properly formed.

The $\mathbb{G}_r$ subgroup is used for to hide factors from other subgroups and ensure plaintext privacy. In an analogous manner, the $\mathbb{G}_s$ subgroup is used to ensure predicate privacy. Also, the additional subgroup $\mathbb{G}_s$ allows us to construct our scheme in a slightly different manner from KSW. For example, the $\mathbb{G}_s$ subgroup allows us to eliminate the factor Q from the $\mathbb{G}_q$ subgroup in the K term of the token.

As discussed earlier, in our proofs of security we will need to show that our main construction is computationally indistinguishable from a scheme in which ciphertexts and tokens are formed symmetrically. In the KSW construction, all terms in the ciphertext have the same exponent y in the $\mathbb{G}_p$ subgroup. In our construction, we introduce an additional degree of randomness using the exponent z. Terms in the ciphertext now contain two degrees of randomness, y and z, in the $\mathbb{G}_p$ subgroup. This change is necessary to ensure symmetry of the ciphertext and the token in the $\mathbb{G}_p$ subgroup.

To see why this is the case, recall that Decisional Diffie-Hellman is easy in bilinear groups. That is, for a random vector $g_p^{\alpha_1}, g_p^{\alpha_2}, \ldots, g_p^{\alpha_k}$, it is easy to decide whether the exponents $(\alpha_1, \alpha_2, \ldots, \alpha_k)$ are picked independently at random or picked from a prescribed one-dimensional subspace. On the other hand, an informal interpretation of the Decisional Linear assumption tells us that it is computationally hard to decide whether the exponents $(\alpha_1, \alpha_2, \ldots, \alpha_k)$ are picked independently at random or picked randomly from a prescribed 2-dimensional subspace. The reason for introducing the extra randomness z in the ciphertext is to ensure that the exponents in the $\mathbb{G}_p$ subgroup are picked from a 2-dimensional subspace instead of a 1-dimensional subspace.

Similarly to [11, 12, 21], our construction consists of two parallel sub-systems. Note that $C_{1,i}$ and $C_{2,i}$ (similarly, $K_{1,i}$ and $K_{2,i}$) play identical roles. Our proof of security will rely on having these two parallel sub-systems.

For comparison, we provide a review of the KSW construction in Appendix B.

4.3 Proof Overview

Our main security statement is the following theorem.

Theorem 3. *Under the generalized Assumption 1 of the KSW construction, the generalized C3DH assumption, and the Decisional Linear assumption, the symmetric-key predicate-only encryption scheme presented in Section 4.1 is selectively single challenge secure.*

Our proof technique consists of two steps. First, we prove that our construction achieves plaintext privacy. Second, we prove that, for our construction, plaintext privacy implies predicate privacy. Taken together, these results imply the security of our scheme.

Our construction defined above, which we call SchemeReal, does not immediately yield a proof of these two properties. In order to argue these properties, we define two different schemes that are computationally indistinguishable from our original construction. That is, no adversary can tell whether tokens and ciphertexts are generated from our actual system or from one of the two defined for the purposes of the proof.

We first define a system that we call SchemeQ, which very closely follows the KSW construction. We reduce the plaintext privacy of SchemeQ to the plaintext privacy of the KSW construction.

Next we define a system that we call SchemeSym, in which ciphertexts and tokens are formed symmetrically. For this system it is straightforward to argue that plaintext privacy implies predicate privacy.

We observe that since our main construction and the two variants defined are all computationally indistinguishable (from an adversary's view), it is actually possible to define any of them as the "real" construction that we will actually use. We chose the variant described above due to (relative) notational simplicity and slight efficiency advantages. Details of our proof and further discussion are given in the full version of our paper.

5 Conclusions

We examined the idea of protecting the privacy of predicates in predicate encryption systems. While this turns out to be an inherently unachievable in a public-key system, we showed that there exist solutions in the symmetric-key setting. We first provided security definitions for predicate encryption schemes in the symmetric-key setting and then presented a construction supporting inner product queries, which are the most expressive queries supported by currently known schemes.

While semantic security of predicates is inherently impossible in the public-key setting, in the future we might wish to consider relaxations of public-key encryption. For example, is it possible to find interesting systems where predicates are drawn from a high entropy distribution, in a fashion similar to recent work on deterministic encryption [4, 2]? Another open direction is to consider "partial public-key encryption," in which a public key might allow a user to generate only a subset of valid ciphertexts. (The rest may be generated from a secret key or other public keys kept hidden from an attacker.) Thus, certain predicates might be indistinguishable given the partial public keys published.

Acknowledgments

We thank Philippe Golle for helpful discussions. The second author thanks Adrian Perrig for his support while part of this research was conducted.

References

1. Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. In *Advances in Cryptology - Proceedings of CRYPTO '05*, pages 205–222. Springer-Verlag, August 2005.
2. Mihir Bellare, Marc Fischlin, Adam O'Neill, and Thomas Ristenpart. Deterministic encryption: Definitional equivalences and constructions without random oracles. In *CRYPTO*, pages 360–378, 2008.

3. John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *SP '07: Proceedings of the 2007 IEEE Symposium on Security and Privacy*, pages 321–334, Washington, DC, USA, 2007. IEEE Computer Society.
4. Alexandra Boldyreva, Serge Fehr, and Adam O'Neill. On notions of security for deterministic encryption, and efficient constructions without random oracles. In *CRYPTO*, pages 335–359, 2008.
5. Dan Boneh and Xavier Boyen. Efficient selective-ID secure identity based encryption without random oracles. In *Advances in Cryptology—EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 223–238. Berlin: Springer-Verlag, 2004. Available at `http://www.cs.stanford.edu/~xb/eurocrypt04b/`.
6. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *CRYPTO*, pages 41–55, 2004.
7. Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In *EUROCRYPT*, pages 506–522, 2004.
8. Dan Boneh and Matt Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *Proceedings of Crypto 2001*, volume 2139 of *LNCS*, pages 213–29. Springer-Verlag, 2001.
9. Dan Boneh, Craig Gentry, and Michael Hamburg. Space-efficient identity based encryption without pairings. In *FOCS*, 2007.
10. Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In Joe Kilian, editor, *Proceedings of Theory of Cryptography Conference 2005*, volume 3378 of *LNCS*, pages 325–342. Springer, 2005.
11. Dan Boneh and Brent Waters. A fully collusion resistant broadcast trace and revoke system with public traceability. In *ACM Conference on Computer and Communication Security (CCS)*, 2006.
12. Xavier Boyen and Brent Waters. Anonymous hierarchical identity-based encryption (without random oracles). In *CRYPTO*, 2006.
13. Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In *EUROCRYPT*, pages 255–271, 2003.
14. Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In *EUROCRYPT*, pages 207–222, 2004.
15. Melissa Chase. Multi-authority attribute based encryption. In *TCC*, pages 515–534, 2007.
16. Clifford Cocks. An identity based encryption scheme based on quadratic residues. In *Proceedings of the 8th IMA International Conference on Cryptography and Coding*, pages 360–363, London, UK, 2001. Springer-Verlag.
17. Reza Curtmola, Juan Garay, Seny Kamara, and Rafail Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. In *CCS '06: Proceedings of the 13th ACM conference on Computer and communications security*, 2006.
18. O. Goldreich and R. Ostrovsky. Software protection and simulation by oblivious rams. *JACM*, 1996.
19. Philippe Golle, Jessica Staddon, and Brent Waters. Secure conjunctive keyword search over encrypted data. In *Proc. of the 2004 Applied Cryptography and Network Security Conference*, 2004.
20. Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *CCS '06: Proceedings of the 13th ACM conference on Computer and communications security*, pages 89–98, New York, NY, USA, 2006. ACM Press.

21. Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *Eurocrypt*, 2008.
22. Rafail Ostrovsky. *Software protection and simulation on oblivious RAMs*. PhD thesis, M.I.T, 1992. Preliminary version in STOC 1990.
23. Rafail Ostrovsky, Amit Sahai, and Brent Waters. Attribute-based encryption with non-monotonic access structures. In *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security*, 2007.
24. Matthew Pirretti, Patrick Traynor, Patrick McDaniel, and Brent Waters. Secure attribute-based systems. In *CCS '06: Proceedings of the 13th ACM conference on Computer and communications security*, pages 99–112, New York, NY, USA, 2006.
25. Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, pages 457–473, 2005.
26. Adi Shamir. Identity-based cryptosystems and signature schemes. In *Proceedings of Crypto '84*, volume 196 of *LNCS*, pages 47–53. Springer-Verlag, 1984.
27. Elaine Shi, John Bethencourt, T-H. Hubert Chan, Dawn Song, and Adrian Perrig. Multi-dimensional range query over encrypted data. In *IEEE Symposium on Security and Privacy*, May 2007.
28. Elaine Shi and Brent Waters. Delegating capabilities in predicate encryption systems. In *Proceedings of ICALP*, 2008. Full version can be found online at `http://sparrow.ece.cmu.edu/~elaine/docs/delegation.pdf`.
29. Dawn Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In *Proceedings of the 2000 IEEE symposium on Security and Privacy (S&P 2000)*, 2000.

A Proof of Theorem 2

Here, we prove that a single challenge secure symmetric-key predicate-only encryption scheme supporting inner product queries for vectors of length $2n$ can be used to construct a fully secure symmetric-key predicate-only encryption scheme supporting inner product queries for vectors length n. Our proof is inspired by the hybrid argument used by [21].

Proof. Let SCHEME_{2n} be a single challenge secure symmetric-key predicate-only encryption scheme supporting inner product queries over $\mathbb{Z}_N^{2n}$. We construct a fully secure symmetric-key predicate-only encryption scheme SCHEME_n supporting inner product queries over $\mathbb{Z}_N^n$.

For any two vectors $\boldsymbol{x} = (x_1, \ldots x_n), \boldsymbol{y} = (y_1, \ldots, y_n) \in \mathbb{Z}_N^n$, define $\boldsymbol{x} \| \boldsymbol{y} = (x_1, \ldots, x_n, y_1, \ldots, y_n)$ to be the vector obtained by concatenating $\boldsymbol{x}$ and $\boldsymbol{y}$.

Informally, SCHEME_n works as follows. To encrypt a vector $\boldsymbol{x} \in \mathbb{Z}_N^n$, encrypt the vector $\boldsymbol{x} \| \boldsymbol{x} \in \mathbb{Z}_N^{2n}$ using SCHEME_{2n}. Similarly, to construct a token for the vector $\boldsymbol{v} \in \mathbb{Z}_N^n$, use SCHEME_{2n} to construct a token for the vector $\boldsymbol{v} \| \boldsymbol{v} \in \mathbb{Z}_N^{2n}$. The algorithms of Scheme_n are defined as follows.

$\text{SCHEME}_n.\mathit{Setup}(1^\lambda)$: Run $\text{SCHEME}_{2n}.\mathit{Setup}(1^\lambda)$. The secret key SK is the same as that generated by SCHEME_{2n}.
$\text{SCHEME}_n.\mathit{Encrypt}(SK, \boldsymbol{x})$: Output $\text{SCHEME}_{2n}.\mathit{Encrypt}(SK, \boldsymbol{x} \| \boldsymbol{x})$.
$\text{SCHEME}_n.\mathit{GenToken}(SK, \boldsymbol{v})$: Output $\text{SCHEME}_{2n}.\mathit{GenToken}(SK, \boldsymbol{v} \| \boldsymbol{v})$.
$\text{SCHEME}_n.\mathit{Query}(TK_{\boldsymbol{v}}, CT)$: Output $\text{SCHEME}_{2n}.\mathit{Query}(TK_{\boldsymbol{v}}, CT)$.

The correctness of SCHEME_n results from the fact that for vectors $\boldsymbol{x}, \boldsymbol{v} \in \mathbb{Z}_N^n$,

$$\langle \boldsymbol{x}, \boldsymbol{v} \rangle = 0 \quad \textit{iff} \quad \langle \boldsymbol{x} \| \boldsymbol{x},\ \boldsymbol{v} \| \boldsymbol{v} \rangle = 0.$$

We now show that SCHEME_n is fully secure. Recall the full security game defined in Section 2.2. First, the challenger picks a random bit b. Next, the adversary $\mathcal{A}$ adaptively issues queries to the challenger. If a query is a ciphertext query $(\boldsymbol{x}_{j,0}, \boldsymbol{x}_{j,1})$, the challenger responds with an encryption of $\boldsymbol{x}_{j,\mathsf{b}}$. If a query is a token query $(\boldsymbol{v}_{i,0}, \boldsymbol{v}_{i,1})$, the challenger responds with a token for $\boldsymbol{v}_{i,\mathsf{b}}$. $\mathcal{A}$'s queries are subject to the restriction that, for all ciphertext queries $(x_{j,0}, x_{j,1})$ and all predicate queries $(f_{i,0}, f_{i,1})$, $f_{i,0}(x_{j,0}) = f_{i,1}(x_{j,1})$. At the end of the game, $\mathcal{A}$ outputs a guess b' of b and wins if $\mathsf{b}' = \mathsf{b}$.

Suppose that the adversary $\mathcal{A}$ makes c ciphertext queries, $(\boldsymbol{x}_{1,0},\ \boldsymbol{x}_{1,1})$, …, $(\boldsymbol{x}_{c,0},\ \boldsymbol{x}_{c,1})$, and t token queries, $(\boldsymbol{v}_{1,0}, \boldsymbol{v}_{1,1}), \ldots, (\boldsymbol{v}_{t,0}, \boldsymbol{v}_{t,1})$.

Our task is to show that $\mathcal{A}$ cannot distinguish between two experiments: one where the challenger constructs ciphertexts for $\boldsymbol{x}_{1,0}, \ldots, \boldsymbol{x}_{c,0}$ and tokens for $\boldsymbol{v}_{1,0}, \ldots, \boldsymbol{v}_{t,0}$ (call this Game 0), and one where the challenger constructs ciphertexts for $\boldsymbol{x}_{1,1}, \ldots, \boldsymbol{x}_{c,1}$ and tokens for $\boldsymbol{v}_{1,1}, \ldots, \boldsymbol{v}_{t,1}$ (call this Game 1). To do this, we construct a series of hybrid games as follows.

Game 0 : The challenger calls SCHEME_{2n} and computes ciphertexts for $\boldsymbol{x}_{1,0}\|\boldsymbol{x}_{1,0}$, $\boldsymbol{x}_{2,0}\|\boldsymbol{x}_{2,0}$, …, $\boldsymbol{x}_{c,0}\|\boldsymbol{x}_{c,0}$ and tokens for $\boldsymbol{v}_{1,0}\|\boldsymbol{v}_{1,0}$, $\boldsymbol{v}_{2,0}\|\boldsymbol{v}_{2,0}$, …, $\boldsymbol{v}_{t,0}\|\boldsymbol{v}_{t,0}$.

Game A : The challenger calls SCHEME_{2n} and computes ciphertexts for $\boldsymbol{x}_{1,0}\|\boldsymbol{0}$, $\boldsymbol{x}_{2,0}\|\boldsymbol{0}$, …, $\boldsymbol{x}_{c,0}\|\boldsymbol{0}$ and tokens for $\boldsymbol{v}_{1,0}\|\boldsymbol{v}_{1,0}$, $\boldsymbol{v}_{2,0}\|\boldsymbol{v}_{2,0}$, …, $\boldsymbol{v}_{t,0}\|\boldsymbol{v}_{t,0}$.

Game B : The challenger calls SCHEME_{2n} and computes ciphertexts for $\boldsymbol{x}_{1,0}\|\boldsymbol{0}$, $\boldsymbol{x}_{2,0}\|\boldsymbol{0}$, …, $\boldsymbol{x}_{c,0}\|\boldsymbol{0}$ and tokens for $\boldsymbol{v}_{1,0}\|\boldsymbol{v}_{1,1}$, $\boldsymbol{v}_{2,0}\|\boldsymbol{v}_{2,1}$, …, $\boldsymbol{v}_{t,0}\|\boldsymbol{v}_{t,1}$.

Game M : The challenger picks a random $\alpha \xleftarrow{\text{R}} \mathbb{Z}_N$, calls SCHEME_{2n} and computes ciphertexts for $\boldsymbol{x}_{1,0}\|\alpha\boldsymbol{x}_{1,1}$, $\boldsymbol{x}_{2,0}\|\alpha\boldsymbol{x}_{2,1}$, …, $\boldsymbol{x}_{c,0}\|\alpha\boldsymbol{x}_{c,1}$ and tokens for $\boldsymbol{v}_{1,0}\|\boldsymbol{v}_{1,1}$, $\boldsymbol{v}_{2,0}\|\boldsymbol{v}_{2,1}$, …, $\boldsymbol{v}_{t,0}\|\boldsymbol{v}_{t,1}$.

Notice that in the above sequence of hybrid games, the outcomes of the predicates corresponding to the generated tokens on the plaintexts in $\mathbb{Z}_N^{2n}$ encrypted by the challenger remain the same between all pairs of adjacent games, except with negligible probability.

Claim. If SCHEME_{2n} is single challenge secure, then no PPT adversary $\mathcal{A}$ has more than negligible advantage in distinguishing between any pair of adjacent games in the above sequence of games.

Proof. By a hybrid argument.

Similarly, we can construct a sequence of hybrid games connecting Game M and Game 1. Using a hybrid argument, we conclude that no PPT adversary has more than negligible advantage in distinguishing between Game 0 and Game 1.

B KSW Predicate Encryption Scheme

To aid in the understanding of our construction and the proof of security, we review the KSW public key predicate-only encryption scheme for inner product queries [21].

Let $\mathcal{G}'$ denote a group generator algorithm for a bilinear group whose order is the product of three distinct primes.

Setup(1^λ): The setup algorithm runs $\mathcal{G}'(1^\lambda)$ to obtain $(p, q, r, \mathbb{G}, \mathbb{G}_T, e)$ with $\mathbb{G} = \mathbb{G}_p \times \mathbb{G}_q \times \mathbb{G}_r$. Next it picks generators g_p, g_q, g_r from subgroups $\mathbb{G}_p, \mathbb{G}_q, \mathbb{G}_r$, respectively. It then chooses, uniformly at random, $h_{1,i}, h_{2,i} \in \mathbb{G}_p$, $R_{1,i}, R_{2,i} \in \mathbb{G}_r$ for $i = 1$ to n, and $R_0 \in \mathbb{G}_r$.
The public key consists of:

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

The secret key is set to:

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

Encrypt$(PK, \boldsymbol{x})$: Let $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{Z}_N^n$. The encryption algorithm first picks random exponents y, α, β from $\mathbb{Z}_N$, and it chooses random $R_{3,i}, R_{4,i} \in \mathbb{G}_r$ for $i = 1$ to n. It outputs the ciphertext

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

GenToken$(SK, \boldsymbol{v})$: Let $\boldsymbol{v} = (v_1, \ldots, v_n) \in \mathbb{Z}_N^n$. The token generation algorithm chooses random $f_1, f_2, \{r_{1,i}, r_{2,i}\}_{i=1}^n$ from $\mathbb{Z}_N$, random $R_5 \in \mathbb{G}_r$, and random $Q_6 \in \mathbb{G}_q$. It outputs the token

$$TK_{\boldsymbol{v}} = \left(\begin{array}{c} K = R_5 \cdot Q_6 \cdot \prod_{i=1}^n h_{1,i}^{-r_{1,i}} \cdot h_{2,i}^{-r_{2,i}}, \\ \left\{K_{1,i} = g_p^{r_{1,i}} \cdot g_q^{f_1 v_i},\ K_{2,i} = g_p^{r_{2,i}} \cdot g_q^{f_2 v_i}\right\}_{i=1}^n \end{array}\right).$$

Query$(TK_{\boldsymbol{v}}, CT)$: Let $CT = (C, \{C_{1,i}, C_{2,i}\}_{i=1}^n)$ and $TK_{\boldsymbol{v}} = (K, \{K_{1,i}, K_{2,i}\}_{i=1}^n)$ as above. The query algorithm outputs 1 iff

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