Function-Hiding Inner Product Encryption is Practical

Sam Kim[1], Kevin Lewi*[2], Avradip Mandal[3],
Hart Montgomery[3], Arnab Roy[3], and David J. Wu[1]

[1]Stanford University
[2]Facebook
[3]Fujitsu Laboratories of America

Abstract

In a functional encryption scheme, secret keys are associated with functions and ciphertexts are associated with messages. Given a secret key for a function f, and a ciphertext for a message x, a decryptor learns $f(x)$ and nothing else about x. Inner product encryption is a special case of functional encryption where both secret keys and ciphertext are associated with vectors. The combination of a secret key for a vector $\mathbf{x}$ and a ciphertext for a vector $\mathbf{y}$ reveal $\langle \mathbf{x}, \mathbf{y} \rangle$ and nothing more about $\mathbf{y}$. An inner product encryption scheme is function-hiding if the keys and ciphertexts reveal no additional information about both $\mathbf{x}$ and $\mathbf{y}$ beyond their inner product.

In the last few years, there has been a flurry of works on the construction of function-hiding inner product encryption, starting with the work of Bishop, Jain, and Kowalczyk (Asiacrypt 2015) to the more recent work of Tomida, Abe, and Okamoto (ISC 2016). In this work, we focus on the practical applications of this primitive. First, we show that the parameter sizes and the run-time complexity of the state-of-the-art construction can be further reduced by another factor of 2, though we compromise by proving security in the generic group model. We then show that function privacy enables a number of applications in biometric authentication, nearest-neighbor search on encrypted data, and single-key two-input functional encryption for functions over small message spaces. Finally, we evaluate the practicality of our encryption scheme by implementing our function-hiding inner product encryption scheme. Using our construction, encryption and decryption operations for vectors of length 50 complete in a tenth of a second in a standard desktop environment.

1 Introduction

Traditionally, encryption schemes have provided an all-or-nothing approach to data access: users can either fully recover the data, or recover none at all. In the last fifteen years, numerous primitives such as identity-based encryption [BF01, Coc01], attribute-based encryption [SW05, BSW07, GVW13] and predicate encryption [KSW08, OT09, GVW15] have been introduced to provide more fine-grained control to encrypted data. Recently, these works have been unified under the general umbrella of functional encryption (FE) [SS10, BSW11, O'N10]. In a functional encryption scheme, the holder of the master secret key is able to delegate arbitrary decryption keys that allow users to learn specific functions of the data, and nothing else. Specifically, given an encryption of a message x and a secret key for a function f, a decryptor only learns the value $f(x)$.

*Work done while at Stanford University.

Functional encryption for inner products. In the last few years, a considerable amount of effort has been dedicated to constructing functional encryption. Currently, we can realize FE for general functions in a restricted setting (i.e., security against "bounded collusions") [SS10, GVW13, GKP+13] from standard assumptions as well as fully-secure FE for general functions [GGH+13, Wat15, GGHZ16]. However, all of these aforementioned works have focused on the theoretical *feasibility* or *existence* of FE. And in fact, all of these general-purpose schemes are far too inefficient to be viable for practical scenarios. Thus, there is currently a significant gap between the kinds of FE that are realizable in theory and what practitioners for concrete applications. In this paper, we take a step towards bridging this gap. We focus on building *practical* functional encryption for a particular class of functionalities, namely the inner product functionality [ABCP15, ALS15, BJK15, DDM16], and show that our construction is both efficient enough for practical deployments while remaining expressive enough to support many interesting applications.

In an inner product encryption (IPE) scheme, both secret keys and ciphertexts are associated with vectors $\mathbf{x} \in \mathbb{Z}_q^n$ and $\mathbf{y} \in \mathbb{Z}_q^n$ of length n over a ring $\mathbb{Z}_q$. Given a secret key $\mathsf{sk}_{\mathbf{x}}$ for $\mathbf{x}$ and a ciphertext $\mathsf{ct}_{\mathbf{y}}$ for $\mathbf{y}$, the decryption function outputs the *value* $\langle \mathbf{x}, \mathbf{y} \rangle \in \mathbb{Z}_q$, which is the inner product of their associated vectors. We emphasize that this definition of IPE is different from the notion of inner product predicate encryption from [KSW08, SSW09, OT09, OT10, OT12]. In an inner product predicate encryption scheme, a message m is encrypted with a tag $\mathbf{y} \in \mathbb{Z}_q^n$. Decryption keys are still associated with vectors $\mathbf{x} \in \mathbb{Z}_q^n$. When a secret key for $\mathbf{x}$ is used to decrypt a ciphertext with tag $\mathbf{y}$, the output is m if $\langle \mathbf{x}, \mathbf{y} \rangle = 0$ and $\perp$ otherwise. In other words, decryption recovers the message only if the vectors of the secret key and ciphertext are orthogonal. In contrast, decryption in our setting outputs the actual value of the inner product.

Function-hiding IPE. Functional encryption enables delegation of decryption capabilities by issuing different function keys to users. In many applications, however, we require the additional property that the function keys themselves also hide the underlying function. This problem was first considered by Shen, Shi, and Waters [SSW09] for inner-product predicate encryption in the secret-key setting, and subsequently by many others in both the secret-key setting [AAB+15, BS15] and the public-key setting [BRS13a, BRS13b]. Bishop, Jain, and Kowalczyk [BJK15] were the first to give a direct construction of secret-key function-hiding IPE under an indistinguishability-based definition from the Symmetric External Diffie-Hellman (SXDH) assumption in bilinear groups. However, their security model imposes a somewhat unrealistic admissibility constraint on the adversary's queries.[1] Subsequently, Datta, Dutta, and Mukhopadhyay [DDM16] showed how to construct a secret-key function-hiding IPE from the SXDH assumption that removes the need for that additional constraint on the adversary's queries. In their construction, secret keys and ciphertexts of n-dimensional vectors consist of $4n + 8$ group elements. This was further improved upon in a work by Tomida, Abe, and Okamoto [TAO16] who gave a construction of a secret-key function hiding IPE from the DLIN assumption where the secret keys and ciphertexts consist of $2n + 5$ group elements. Recently, Kim, Kim, and Seo [KKS17] also gave a construction of function-hiding IPE from the SXDH assumption where the secret keys and ciphertexts consist of $2n + 8$ group elements. Notably, the master secret key in their construction is much smaller and just contains $6n + 4$ field elements (in all previous constructions, including this one, the master secret key contains $O(n^2)$ elements).

[1] Lin and Vaikuntanathan [LV16] subsequently showed how to *generically* boost function-hiding IPE schemes that satisfy this weaker notion of security to one that satisfies the full notion of security. Their generic transformation introduces a factor of 2 overhead in the length of the secret keys and ciphertexts.

Additional related work. Recently, Ramanna [Ram16] proposed new constructions for inner product encryption from the SXDH assumption, with applications to identity-based broadcast encryption. However, this construction also requires the use of quasi-adaptive non-interactive zero knowledge proofs. In addition, Abdalla, Raykova, and Wee [ARW16] as well as Lee and Lee [LL18] study how to use concrete assumptions on bilinear maps to obtain multi-input FE for inner products, but in a non-function-hiding setting.

1.1 Our Contributions

In this work, we give a new construction of a function-hiding inner product encryption where secret keys and ciphertexts of n-dimensional vectors contain just $n+1$ group elements. This corresponds to a noticeable reduction (by a factor of 4, 2, and 2, respectively) in parameter sizes (specifically, in the size of the secret keys and the ciphertexts) compared to the existing schemes of Datta et al. [DDM16], Tomida et al. [TAO16], and Kim et al. [KKS17]. We prove the security of our construction under a *stronger* simulation-based notion of security in the generic group model. We then describe a number of new applications enabled by inner product encryption. We highlight two of these applications here and give the full description in Section 4:

- **Biometric authentication:** Biometric-based authentication has grown in popularity to both augment and replace password-based authentication. Unlike the passwords in password-based authentication, biometrics are inherently noisy, so requiring exact matches between a supplied biometric and a user's ground truth credential generally does not work. A more appropriate metric is the *closeness* of the supplied biometric to the ground truth. We show that inner product encryption can be used to compute Hamming distances between bitstrings. This gives a simple biometric-based authentication system.

- **Nearest-neighbor search on encrypted data:** Consider an encrypted database of documents D. Given a document d, the problem of k-nearest-neighbor search is that of finding the top-k documents in D that are most similar to the query document d (for some definition of document similarity). A commonly used metric for document similarity is ℓ_2 distance between a vectorial representation of documents. We show that inner product encryption provides an efficient method to perform nearest-neighbor search over an encrypted database. Our work contrasts with existing works on searchable symmetric encryption (SSE) [SWP00, Goh03, CGKO06, BBO07] in that our protocols focus on retrieving *similar* documents, while SSE primarily deals with retrieving documents based on exact or partial keyword matches.

In addition to the above applications, we also show how to build a fully-secure single-key, two-input functional encryption scheme in the "small-message" setting (i.e., for schemes over a polynomial-sized plaintext space) using function-hiding IPE. Compared to existing functional encryption schemes that do not rely on heavy machinery such as multilinear maps or indistinguishability obfuscation, our scheme achieves significantly shorter ciphertexts.

IPE to two-input functional encryption. Multi-input functional encryption (MIFE) [GGG+14] generalizes FE to the setting where decryption keys are associated with functions of several inputs. A special case of MIFE is two-input functional encryption where the decryption function takes a secret key sk_f for a binary function f and two encryptions ct_x and ct_y of messages x and

y, respectively, and outputs $f(x, y)$. Notably, two-input functional encryption (for just a single function[2] f) suffices to construct property-preserving encryption [PR12] for binary properties. Such a property-preserving encryption scheme is defined with respect to a Boolean predicate P on two inputs. Then, there exists a *publicly* computable function that takes encryptions of messages x and y and decides whether $P(x, y)$ is satisfied. A special case of property-preserving encryption (for the comparison predicate) that has been extensively studied in recent years is order-preserving encryption (OPE) [AKSX04, BCLO09, BCO11], and its generalization, order-revealing encryption (ORE) [BLR$^+$15, CLWW16]. Property-preserving encryption for binary properties can be easily constructed from a two-input functional encryption scheme by simply publishing a function key sk_P for the predicate P. Checking whether two ciphertexts satisfy the predicate simply corresponds to decryption in the underlying functional encryption scheme. In this work, we show that inner-product encryption can be used very naturally to build a single-key, two-input FE scheme in the secret key setting for polynomially-sized domains. This gives a property-preserving encryption scheme for arbitrary binary properties over small domains.

Currently, all alternative constructions of general-purpose MIFE rely on strong primitives such as indistinguishability obfuscation [GGG$^+$14] or multilinear maps [BLR$^+$15]. While recent results show how to transform any functional encryption scheme into a MIFE scheme [AJ15, BKS15], applying these transformations to single-input functional encryption schemes based on standard assumptions [GVW12, GKP$^+$13] yields schemes that are secure only if the adversary obtains an *a priori* bounded number of secret keys *and* ciphertexts.[3] This means that even in the single-key setting, the adversary is still restricted to making an *a priori* bounded number of message queries. Moreover, in these existing constructions, the length of the ciphertexts is at least $\Omega(Q^4)$ where Q is the bound on the number of message queries the adversary makes.

In this work, we give an efficient construction of a single-key, two-input functional encryption scheme for general functions in the secret-key setting where the message space is small.[4] Because the function f is a function of two inputs, there are two types of ciphertexts: "left" encryptions of messages for the first input to f and "right" encryptions of messages for the second input to f. The reduction to function-hiding inner product encryption is very simple and resembles the "brute-force" construction of functional encryption from [BSW11, §4.1]. Specifically, if the message space is the set $\{m_1, \ldots, m_n\}$, a "left" encryption of a message m_i is just an IPE function key $\mathsf{sk}_{\mathbf{e}_i}$ for the basis vector $\mathbf{e}_i$ ($\mathbf{e}_{ii} = 1$ and $\mathbf{e}_{ij} = 0$ for all $i \neq j$). A right encryption of a message m_j is an IPE ciphertext for the vector $\mathbf{f}_j$ of functional evaluations where $\mathbf{f}_{jk} = f(m_k, m_j)$. By construction, $\langle \mathbf{e}_i, \mathbf{f}_j \rangle = \mathbf{f}_{ji} = f(m_i, m_j)$. Security of our construction follows from the fact that the IPE scheme is function-hiding. In contrast to existing constructions of MIFE from standard assumptions, the size of the ciphertexts in our two-input functional encryption scheme is *independent* of the number of ciphertext queries the adversary makes.

[2]This setting where the functionality f is fixed in advance is referred to as the single-key setting.

[3]This limitation arises because the underlying FE scheme is only secure against "bounded collusions," i.e., secure if the adversary makes a bounded number of key generation queries. After applying the single-input-to-multi-input transformation, this translates into an *additional* restriction on the number of ciphertexts the adversary can request.

[4]Recently, Lewi and Wu [LW16] along with Joye and Passelègue [JP16] independently gave constructions of order-revealing encryption from one-way functions in the small-message-space setting. While the techniques of [LW16] can be further extended to work for any two-input functionalities, their construction necessarily reveals whether two ciphertexts encrypt the same underlying value. The construction we propose applies more generally to *arbitrary* two-input functionality without this limitation.

Our construction. The starting point for our function-hiding IPE scheme is the constructions of [BJK15, DDM16, TAO16] which all leverage the power of dual pairing vector spaces developed by Okamoto and Takashima [OT08]. The master secret key in their constructions [BJK15, DDM16] consists of a random basis for a dual pairing vector space. In our work, we scale this basis by a fixed value (dependent on the basis). We correspondingly scale the components of the secret key. Our final scheme achieves shorter secret keys and ciphertexts, with no loss in security. We give a formal security proof in the generic bilinear group model.

Although achieving security in the standard model is important, we are able to obtain a significantly more efficient inner product encryption scheme (with full security) compared to all previous constructions [BJK15, DDM16] by relying on idealized assumptions. A series of works [FST10, AFK+12, Cos12] in the last 15 years have focused on constructing and characterizing pairing-friendly elliptic curves where the complexity of all known non-generic attacks over these curves is extremely high. The heuristic evidence thus suggests that if we instantiate our construction using one of these pairing-friendly elliptic curves, the best attacks will be generic in nature.[5] Though a proof in the generic group model is generally less satisfying than one in the standard model, for most practical applications, using a scheme whose security analysis leverages an idealized model is not a severe limitation. In fact, by considering constructions whose security relies on the generic group model, we obtain a function-hiding IPE scheme whose secret keys and ciphertexts are much shorter than those of existing schemes, and hence, quite efficient.

Implementation. To assess the practicality of our inner product encryption scheme, we provide a complete and open-source implementation[6] of our scheme in Python. We also perform a series of micro-benchmarks on our inner product encryption scheme for a wide range of parameter settings. Our results show that our encryption scheme is practical for a wide variety of real-world scenarios. For example, encrypting vectors of length 50 completes in about a tenth of a second on a typical desktop. Ciphertexts in our scheme are just a few KB. We describe our implementation and the micro-benchmarks we perform in Section 5.

2 Preliminaries

2.1 Notation

For an integer n, we write $[n]$ to denote the set $\{1, \ldots, n\}$. For a finite set S, we write $x \xleftarrow{R} S$ to denote sampling x uniformly at random from S. We use bold lowercase letters (e.g., $\mathbf{v}, \mathbf{w}$) to denote vectors and bold uppercase letters (e.g., $\mathbf{B}$, $\mathbf{B}^*$) to denote matrices. For a matrix $\mathbf{B}$, we use $\mathbf{B}^\top$ to denote the transpose of $\mathbf{B}$ and $\det(\mathbf{B})$ to denote its determinant. We recall that $\mathbb{GL}_n(\mathbb{Z}_q)$ is the general linear group of $(n \times n)$ matrices over $\mathbb{Z}_q$ (i.e., invertible matrices over $\mathbb{Z}_q$). We write λ for the security parameter. We say a function $\varepsilon(\lambda)$ is negligible in λ, if $\varepsilon(\lambda) = o(1/\lambda^c)$ for every $c \in \mathbb{N}$, and we write $\mathsf{negl}(\lambda)$ to denote a negligible function in λ. We say that an event occurs with *negligible probability* if the probability of the event is $\mathsf{negl}(\lambda)$, and an event occurs with *overwhelming probability* if its complement occurs with negligible probability.

[5] On certain classes of pairing curves, there are indeed non-generic attacks [KB16]. Nonetheless, there still remains a large class of pairing curves where there are no known non-generic attacks that perform significantly better than the generic ones.

[6] Our open-source implementation is available at `https://github.com/kevinlewi/fhipe`.

2.2 Bilinear Groups

In this section, we recall some basic definitions on (asymmetric) bilinear groups [Jou00, BF01, Mil04]. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two distinct groups of prime order q, and let $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$ be generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a function that maps two elements from $\mathbb{G}_1$ and $\mathbb{G}_2$ onto a target group $\mathbb{G}_T$, also of prime order q. In this work, we write the group operation in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ multiplicatively and write 1 to denote their multiplicative identity. We say that the tuple $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q, e)$ is an asymmetric bilinear group if the following properties hold:

- The group operations in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$, as well as the map e, are efficiently computable.
- The map e is non-degenerate: $e(g_1, g_2) \neq 1$.
- The map e is bilinear: for all $x, y \in \mathbb{Z}_q$, we have that $e(g_1^x, g_2^y) = e(g_1, g_2)^{xy}$.

In this work, we will often work with vectors of group elements. Let $\mathbb{G}$ be a group of prime order q. Then, for any group element $g \in \mathbb{G}$, and row vector $\mathbf{v} = (v_1, \ldots, v_n) \in \mathbb{Z}_q^n$, where $n \in \mathbb{N}$, we write $g^{\mathbf{v}}$ to denote the vector of group elements $(g^{v_1}, \ldots, g^{v_n})$. Moreover, for any scalar $k \in \mathbb{Z}_q$ and vectors $\mathbf{v}, \mathbf{w} \in \mathbb{Z}_q^n$, we write

$$(g^{\mathbf{v}})^k = g^{(k\mathbf{v})} \qquad \text{and} \qquad g^{\mathbf{v}} \cdot g^{\mathbf{w}} = g^{\mathbf{v}+\mathbf{w}}.$$

The pairing operation over groups is naturally extended to vectors as follows:

$$e(g_1^{\mathbf{v}}, g_2^{\mathbf{w}}) = \prod_{i \in [n]} e(g_1^{v_i}, g_2^{w_i}) = e(g_1, g_2)^{\langle \mathbf{v}, \mathbf{w} \rangle}.$$

2.3 Function-Hiding Inner Product Encryption

A (secret-key) inner product encryption (IPE) scheme is a tuple of algorithms $\Pi_{\mathsf{ipe}} = (\mathsf{IPE.Setup}, \mathsf{IPE.KeyGen}, \mathsf{IPE.Encrypt}, \mathsf{IPE.Decrypt})$ defined over a message space $\mathbb{Z}_q^n$ with the following properties:

- $\mathsf{IPE.Setup}(1^\lambda, S) \to (\mathsf{pp}, \mathsf{msk})$: On input a security parameter λ and a set $S \subseteq \mathbb{Z}_q$, the setup algorithm $\mathsf{IPE.Setup}$ outputs the public parameters pp and the master secret key msk.
- $\mathsf{IPE.KeyGen}(\mathsf{msk}, \mathbf{x}) \to \mathsf{sk}_{\mathbf{x}}$: On input the master secret key msk and a vector $\mathbf{x} \in \mathbb{Z}_q^n$, the key generation algorithm $\mathsf{IPE.KeyGen}$ outputs a functional secret key $\mathsf{sk}_{\mathbf{x}}$.
- $\mathsf{IPE.Encrypt}(\mathsf{msk}, \mathbf{y}) \to \mathsf{ct}_{\mathbf{y}}$: On input the master secret key msk and a vector $\mathbf{y} \in \mathbb{Z}_q^n$, the encryption algorithm $\mathsf{IPE.Encrypt}$ outputs a ciphertext $\mathsf{ct}_{\mathbf{y}}$.
- $\mathsf{IPE.Decrypt}(\mathsf{pp}, \mathsf{sk}, \mathsf{ct}) \to z$: On input the public parameters pp, a functional secret key sk, and a ciphertext ct, the decryption algorithm $\mathsf{IPE.Decrypt}$ either outputs a message $z \in \mathbb{Z}_q$ or a special symbol $\perp$.

Correctness. An IPE scheme $\Pi_{\mathsf{ipe}} = (\mathsf{IPE.Setup}, \mathsf{IPE.KeyGen}, \mathsf{IPE.Encrypt}, \mathsf{IPE.Decrypt})$ defined over a message space $\mathbb{Z}_q^n$ is correct if for all sets S where $|S| = \mathrm{poly}(\lambda)$, and all non-zero vectors $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_q^n \setminus \{\mathbf{0}\}$, where $\langle \mathbf{x}, \mathbf{y} \rangle \in S$, the following conditions holds. Letting $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{IPE.Setup}(1^\lambda, S)$, $\mathsf{sk}_{\mathbf{x}} \leftarrow \mathsf{IPE.KeyGen}(\mathsf{msk}, \mathbf{x})$, $\mathsf{ct}_{\mathbf{y}} \leftarrow \mathsf{IPE.Encrypt}(\mathsf{msk}, \mathbf{y})$, then

$$\Pr\left[\mathsf{IPE.Decrypt}(\mathsf{pp}, \mathsf{sk}_{\mathbf{x}}, \mathsf{ct}_{\mathbf{y}}) = \langle \mathbf{x}, \mathbf{y} \rangle\right] = 1 - \mathsf{negl}(\lambda).$$

Indistinguishability-based security. Previous works [ABCP15, ALS15, BJK15, DDM16] on inner product encryption considered an indistinguishability notion of security. We review this definition here.

Let $\Pi_{\mathsf{ipe}} = (\mathsf{IPE.Setup}, \mathsf{IPE.KeyGen}, \mathsf{IPE.Encrypt}, \mathsf{IPE.Decrypt})$ be an inner product encryption scheme. We now define the following experiment between a challenger and an adversary $\mathcal{A}$ that can make key generation and encryption oracle queries. In the following, we let $\mathbb{Z}_q$ be our message space and $S \subseteq \mathbb{Z}_q$ be any polynomial-size subset of the message space.

Definition 2.1 (Experiment $\mathsf{Expt}_b^{\mathsf{ipe\text{-}ind}}$)**.** Let $b \in \{0,1\}$. The challenger computes $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{IPE.Setup}(1^\lambda, S)$, gives pp to the adversary $\mathcal{A}$, and then responds to each oracle query type made by $\mathcal{A}$ in the following manner.

- **Key generation oracle.** On input a pair of vectors $\mathbf{x}_0, \mathbf{x}_1 \in \mathbb{Z}_q^n \setminus \{\mathbf{0}\}$, the challenger computes and returns $\mathsf{sk} \leftarrow \mathsf{IPE.KeyGen}(\mathsf{msk}, \mathbf{x}_b)$.
- **Encryption oracle.** On input a pair of vectors $\mathbf{y}_0, \mathbf{y}_1 \in \mathbb{Z}_q^n \setminus \{\mathbf{0}\}$, the challenger computes and returns $\mathsf{ct} \leftarrow \mathsf{IPE.Encrypt}(\mathsf{msk}, \mathbf{y}_b)$.

Eventually, $\mathcal{A}$ outputs a bit b', which is also the output of the experiment, denoted by $\mathsf{Expt}_b^{\mathsf{ipe\text{-}ind}}(\mathcal{A})$.

Definition 2.2 (Admissibility)**.** For an adversary $\mathcal{A}$, let Q_1 and Q_2 be the total number of key generation and encryption oracle queries made by $\mathcal{A}$, respectively. For $b \in \{0,1\}$, let $\mathbf{x}_b^{(1)}, \ldots, \mathbf{x}_b^{(Q_1)} \in \mathbb{Z}_q^n \setminus \{\mathbf{0}\}$ and $\mathbf{y}_b^{(1)}, \ldots, \mathbf{y}_b^{(Q_2)} \in \mathbb{Z}_q^n \setminus \{\mathbf{0}\}$ be the corresponding vectors that $\mathcal{A}$ submits to the key generation and encryption oracles, respectively. We say that $\mathcal{A}$ is *admissible* if for all $i \in [Q_1]$ and $j \in [Q_2]$, we have that

$$\left\langle \mathbf{x}_0^{(i)}, \mathbf{y}_0^{(j)} \right\rangle = \left\langle \mathbf{x}_1^{(i)}, \mathbf{y}_1^{(j)} \right\rangle.$$

Definition 2.3 (IND-Security for IPE)**.** We define an inner product encryption scheme denoted as $\Pi_{\mathsf{ipe}} = (\mathsf{IPE.Setup}, \mathsf{IPE.KeyGen}, \mathsf{IPE.Encrypt}, \mathsf{IPE.Decrypt})$ as fully-secure if for all efficient and admissible adversaries $\mathcal{A}$,

$$\left| \mathsf{Pr}[\mathsf{Expt}_0^{\mathsf{ipe\text{-}ind}}(\mathcal{A}) = 1] - \mathsf{Pr}[\mathsf{Expt}_1^{\mathsf{ipe\text{-}ind}}(\mathcal{A}) = 1] \right| = \mathsf{negl}(\lambda).$$

Simulation-based security. Next, we strengthen the indistinguishability based notion of security by introducing a simulation-based definition.[7] In the simulation-based definition, we require that every efficient adversary that interacts with the real encryption and key generation oracles can be simulated given only oracle access to the inner products between each pair of vectors the adversary submits to the key generation and encryption oracles.

Definition 2.4 (SIM-Security for IPE)**.** Let $\Pi_{\mathsf{ipe}} = (\mathsf{IPE.Setup}, \mathsf{IPE.KeyGen}, \mathsf{IPE.Encrypt}, \mathsf{IPE.Decrypt})$ be an inner product encryption scheme over a message space $\mathbb{Z}_q^n$. Then Π_{ipe} is SIM-secure if for all efficient adversaries $\mathcal{A}$, there exists an efficient simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3)$ such that the outputs of the following experiments are computationally indistinguishable:

[7] There are many lower bounds [BSW11, AGVW13] for the types of functional encryption that can be achieved under a simulation-based definition in the standard model. These lower bounds do not apply in idealized models such as the generic group model. See Remark 2.6 for additional details.

$\mathsf{Real}_{\mathcal{A}}(1^\lambda)$:

1. $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{IPE.Setup}(1^\lambda)$
2. $b \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{KeyGen}}(\mathsf{msk},\cdot),\ \mathcal{O}_{\mathsf{Enc}}(\mathsf{msk},\cdot)}(1^\lambda)$
3. output b

$\mathsf{Ideal}_{\mathcal{A},\mathcal{S}}(1^\lambda)$:

1. initialize $\mathcal{X} \leftarrow \emptyset$ and $\mathcal{Y} \leftarrow \emptyset$
2. $(\mathsf{pp}, \mathsf{st}) \leftarrow \mathcal{S}_1(1^\lambda)$
3. $b \leftarrow \mathcal{A}^{\mathcal{O}'_{\mathsf{KeyGen}}(\cdot),\ \mathcal{O}'_{\mathsf{Enc}}(\cdot)}(1^\lambda, \mathsf{pp})$
4. output b

where the oracles $\mathcal{O}_{\mathsf{KeyGen}}(\mathsf{sk},\cdot)$, $\mathcal{O}_{\mathsf{Enc}}(\mathsf{sk},\cdot)$, $\mathcal{O}'_{\mathsf{KeyGen}}(\cdot)$, $\mathcal{O}'_{\mathsf{Enc}}(\cdot)$ are defined as follows:

- Oracles $\mathcal{O}_{\mathsf{KeyGen}}(\mathsf{sk},\cdot)$ and $\mathcal{O}_{\mathsf{Enc}}(\mathsf{sk},\cdot)$ represent the real encryption and key generation oracles, respectively. Specifically, $\mathcal{O}_{\mathsf{KeyGen}}(\mathsf{sk},\mathbf{x}) = \mathsf{IPE.KeyGen}(\mathsf{sk},\mathbf{x})$ and $\mathcal{O}_{\mathsf{Enc}}(\mathsf{sk},\mathbf{y}) = \mathsf{IPE.Encrypt}(\mathsf{sk},\mathbf{y})$.
- Oracles $\mathcal{O}'_{\mathsf{KeyGen}}(\cdot)$ and $\mathcal{O}'_{\mathsf{Enc}}(\cdot)$ represent the ideal encryption and key generation oracles, respectively. The two oracles are stateful and share counters i and j (initialized to 0 at the beginning of the experiment) a simulator state st (initialized to the state output by $\mathcal{S}_1$), and a collection of mappings

$$\mathcal{C}_{\mathsf{ip}} = \left\{(i',j') \mapsto \langle \mathbf{x}^{(i')}, \mathbf{y}^{(j')} \rangle : i' \in [i], j' \in [j]\right\},$$

  where $\mathbf{x}^{(i)} \in \mathbb{Z}_q^n$ and $\mathbf{y}^{(j)} \in \mathbb{Z}_q^n$ are the inputs for the i^{th} invocation of $\mathcal{O}'_{\mathsf{KeyGen}}(\cdot)$ and the j^{th} invocation of $\mathcal{O}'_{\mathsf{Enc}}(\cdot)$ by the adversary, respectively. At the beginning of the experiment, the set $\mathcal{C}_{\mathsf{ip}}$ is initialized to the empty set.

  - On the adversary's i^{th} invocation of $\mathcal{O}'_{\mathsf{KeyGen}}(\cdot)$ with input vector $\mathbf{x}^{(i)} \in \mathbb{Z}_q^n$, the oracle $\mathcal{O}'_{\mathsf{KeyGen}}$ sets $i \leftarrow i+1$, updates the collection of mappings $\mathcal{C}_{\mathsf{ip}}$, and invokes the simulator $\mathcal{S}_2$ on inputs $\mathcal{C}_{\mathsf{ip}}$ and st. The simulator responds with a tuple $(\mathsf{sk}_{\mathbf{x}}, \mathsf{st}') \leftarrow \mathcal{S}_2(\mathcal{C}_{\mathsf{ip}}, \mathsf{st})$. The oracle updates the state $\mathsf{st} \leftarrow \mathsf{st}'$ and replies to the adversary with the secret key $\mathsf{sk}_{\mathbf{x}}$.
  - Similarly, on the adversary's j^{th} invocation of $\mathcal{O}'_{\mathsf{Enc}}(\cdot)$ with input vector $\mathbf{y} \in \mathbb{Z}_q^n$, the oracle $\mathcal{O}'_{\mathsf{Enc}}$ sets $j \leftarrow j+1$, updates the collection of mappings $\mathcal{C}_{\mathsf{ip}}$, and invokes the simulator $\mathcal{S}_3$ on input $\mathcal{C}_{\mathsf{ip}}$ and st. The simulator responds with a tuple $(\mathsf{ct}_{\mathbf{y}}, \mathsf{st}') \leftarrow \mathcal{S}_3(\mathcal{C}_{\mathsf{ip}}, \mathsf{st})$. The oracle updates the state $\mathsf{st} \leftarrow \mathsf{st}'$ and replies to the adversary with the ciphertext $\mathsf{ct}_{\mathbf{y}}$.

Remark 2.5 ($\mathsf{SIM} \implies \mathsf{IND}$)**.** It is straightforward to see that an IPE scheme that is secure under the simulation-based definition (Definition 2.4) is also secure under the indistinguishability-based definition (Definition 2.3).

Remark 2.6 (SIM-Security Lower Bound)**.** While the simulation-based notion of security (Definition 2.4) is very natural and captures the security guarantees we desire from a function-hiding inner-product encryption scheme, simulation-security is impossible in the standard model. This lower bound follows from the same argument made to show impossibility of non-interactive non-committing encryption [Nie02] and of simulation-secure functional encryption in the public-key setting [BSW11, §5.1]. We note that this lower bound only applies to *function-hiding* inner-product encryption. These lower bounds do not hold in idealized models such as the random oracle or the generic group model.

2.4 The Generic Group Model

In this work, we prove the security of our construction in a generic model of bilinear groups [BBS04, BBG05], which is an extension of the generic group model [Nec94, Sho97]. In the generic group model, access to the group elements is replaced by "handles." An adversary in the generic group model is also given access to a stateful oracle which implements the group operation, and in the bilinear group setting, the pairing operation. The generic group oracle maintains internally a mapping from handles to group elements, which it uses in order to consistently answer the oracle queries. Thus, when a scheme is shown to satisfy some security property in the generic group model, it means that no efficient adversary that only applies the group operations as a black-box can break that security property. As noted in Section 1, there is strong heuristic evidence that suggests that the best known attacks on existing pairing-friendly elliptic curves will be generic in nature.

Definition 2.7 (Generic Bilinear Group Oracle)**.** A generic bilinear group oracle is a stateful oracle $\mathcal{G}$ that responds to queries as follows.

- On a query $\mathsf{BG.Setup}(1^\lambda)$, the generic bilinear group oracle will generate two fresh nonces $\mathsf{pp}, \mathsf{sp} \xleftarrow{\text{R}} \{0,1\}^\lambda$ and a prime q (as in the real setup procedure). It outputs $(\mathsf{pp}, \mathsf{sp}, q)$. It will store the values generated, initialize an empty table $T \leftarrow \{\}$, and set the internal state so subsequent invocations of $\mathsf{BG.Setup}$ fail.

- On a query $\mathsf{BG.Encode}(k, x, i)$ where $k \in \{0,1\}^\lambda$, $x \in \mathbb{Z}_q$, and $i \in \{1, 2, \text{T}\}$ (for the "left" group $\mathbb{G}_1$, the "right" group $\mathbb{G}_2$, and the target group $\mathbb{G}_T$), the oracle checks that $k = \mathsf{sp}$ (returning $\perp$ if the check fails). The oracle then generates a fresh nonce $h \xleftarrow{\text{R}} \{0,1\}^\lambda$, adds the entry $h \mapsto (x, i)$ to the table T, and replies with h.

- On a query $\mathsf{BG.Add}(k, h_1, h_2)$ where $k, h_1, h_2 \in \{0,1\}^\lambda$, the oracle checks that $k = \mathsf{pp}$, that the handles h_1, h_2 are present in its internal table T, and are mapped to values (x_1, i_1) and (x_2, i_2), respectively, and where $i_1 = i_2$ (returning $\perp$ otherwise). If the checks pass, the oracle generates a fresh handle $h \xleftarrow{\text{R}} \{0,1\}^\lambda$, computes $x = x_1 + x_2 \in \mathbb{Z}_q$, adds the entry $h \mapsto (x, i_1)$ to T, and replies with h.

- On a query $\mathsf{BG.Pair}(k, h_1, h_2)$ where $k, h_1, h_2 \in \{0,1\}^\lambda$, the oracle checks that $k = \mathsf{pp}$, that the handles h_1, h_2 are present in T, and are mapped to values $(x_1, 1)$ and $(x_2, 2)$, respectively (returning $\perp$ otherwise). If the checks pass, the oracle generates a fresh handle $h \xleftarrow{\text{R}} \{0,1\}^\lambda$, computes $x = x_1 x_2 \in \mathbb{Z}_q$, adds the entry $h \mapsto (x_1 x_2, \text{T})$ to T, and replies with h.

- On a query $\mathsf{BG.ZeroTest}(k, x)$ where $k, x \in \{0,1\}^\lambda$, the oracle checks that $k = \mathsf{pp}$, that the handle h is present in T, and that h maps to some value (x, i) (returning $\perp$ otherwise). If the checks pass, the oracle returns "zero" if $x = 0 \in \mathbb{Z}_q$ and "non-zero" otherwise.

When analyzing schemes in a generic group model, it is often conducive to view the oracle queries as operating over formal polynomials. We formalize this in the following remark adapted from [Zim15, Remark 2.11]

Remark 2.8 (Oracle Queries as Formal Polynomials)**.** Although the generic bilinear map oracle is defined formally in terms of "handles" (Definition 2.7), it is more conducive to regard each oracle query as referring to a formal query polynomial. The formal variables in this formal query polynomial are specified by the expressions supplied to the $\mathsf{BG.Encode}$ oracle (as determined by the

details of the construction), and the adversary can construct terms that refer to new polynomials by making oracle queries for the generic group operations $\mathsf{BG.Add}$ and $\mathsf{BG.Pair}$. Rather than operating on a "handle," each valid $\mathsf{BG.ZeroTest}$ query refers to a formal query polynomial, and the result of the query is "zero" if the polynomial evaluates to zero when its variables are instantiated with the joint distribution over their values in $\mathbb{Z}_q$ as generated in the real security game.

We will also require the Schwartz-Zippel lemma [Sch80, Zip79], stated as follows.

Lemma 2.9 (Schwartz-Zippel [Sch80, Zip79], adapted)**.** *Fix a prime p and let $f \in \mathbb{F}_p[x_1, \ldots, x_n]$ be an n-variate polynomial with degree at most d and which is not identically zero. Then,*

$$\Pr[x_1, \ldots, x_n \xleftarrow{\mathrm{R}} \mathbb{F}_p : f(x_1, \ldots, x_n) = 0] \leq d/p.$$

3 Construction

In this section, we give our construction of function-hiding inner-product encryption. We then show that the scheme is simulation-secure (Definition 2.4) in the generic group model. Fix a security parameter $\lambda \in \mathbb{N}$, and let n be a positive integer. Let S be a polynomial-sized subset of $\mathbb{Z}_q$. We construct a function-hiding IPE scheme $\Pi_{\mathsf{ipe}} = (\mathsf{IPE.Setup}, \mathsf{IPE.KeyGen}, \mathsf{IPE.Encrypt}, \mathsf{IPE.Decrypt})$ as follows.

- $\mathsf{IPE.Setup}(1^\lambda, S)$: On input the security parameter λ, the setup algorithm samples an asymmetric bilinear group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q, e)$ and chooses generators $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$. Then, it samples $\mathbf{B} \leftarrow \mathbb{GL}_n(\mathbb{Z}_q)$ and sets $\mathbf{B}^\star = \det(\mathbf{B}) \cdot (\mathbf{B}^{-1})^\top$. Finally, the setup algorithm outputs the public parameters $\mathsf{pp} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q, e, S)$ and the master secret key $\mathsf{msk} = (\mathsf{pp}, g_1, g_2, \mathbf{B}, \mathbf{B}^\star)$.

- $\mathsf{IPE.KeyGen}(\mathsf{msk}, \mathbf{x})$: On input the master secret key msk and a vector $\mathbf{x} \in \mathbb{Z}_q^n$, the key generation algorithm chooses a uniformly random element $\alpha \xleftarrow{\mathrm{R}} \mathbb{Z}_q$ and outputs the pair

$$\mathsf{sk} = (K_1, K_2) = \left(g_1^{\alpha \cdot \det(\mathbf{B})},\ g_1^{\alpha \cdot \mathbf{x} \cdot \mathbf{B}}\right).$$

  Note that the second component is a vector of group elements.

- $\mathsf{IPE.Encrypt}(\mathsf{msk}, \mathbf{y})$: On input the master secret key msk and a vector $\mathbf{y} \in \mathbb{Z}_q^n$, the encryption algorithm chooses a uniformly random element $\beta \xleftarrow{\mathrm{R}} \mathbb{Z}_q$ and outputs the pair

$$\mathsf{ct} = (C_1, C_2) = \left(g_2^{\beta}\ ,\ g_2^{\beta \cdot \mathbf{y} \cdot \mathbf{B}^\star}\right).$$

- $\mathsf{IPE.Decrypt}(\mathsf{pp}, \mathsf{sk}, \mathsf{ct})$: On input the public parameters pp, a secret key $\mathsf{sk} = (K_1, K_2)$ and a ciphertext $\mathsf{ct} = (C_1, C_2)$, the decryption algorithm computes

$$D_1 = e(K_1, C_1) \qquad \text{and} \qquad D_2 = e(K_2, C_2).$$

  Then, it checks whether there exists $z \in S$ such that $(D_1)^z = D_2$. If so, the decryption algorithm outputs z. Otherwise, it outputs $\perp$. Note that this algorithm is efficient since $|S| = \mathrm{poly}(\lambda)$.

Correctness. As in [BJK15, DDM16], correctness holds when the plaintext vectors $\mathbf{x}$ and $\mathbf{y}$ satisfy $\langle \mathbf{x}, \mathbf{y} \rangle \in S$, for a polynomially-sized S. The correctness of Π_{ipe} follows from the fact that for any secret key $\mathsf{sk}_{\mathbf{x}} = (K_1, K_2)$ corresponding to a vector $\mathbf{x}$ and any ciphertext $\mathsf{ct}_{\mathbf{y}} = (C_1, C_2)$ corresponding to a vector $\mathbf{y}$, we have that

$$D_1 = e(K_1, C_1) = e(g_1, g_2)^{\alpha\beta \cdot \det(\mathbf{B})}$$

and

$$D_2 = e(K_2, C_2) = e(g_1, g_2)^{\alpha\beta \cdot \mathbf{x}\mathbf{B}(\mathbf{B}^\star)^\top \mathbf{y}^\top} = e(g_1, g_2)^{\alpha\beta \cdot \det(\mathbf{B}) \cdot \langle \mathbf{x}, \mathbf{y} \rangle},$$

where the last equality holds by the relation $\mathbf{B}(\mathbf{B}^\star)^\top = \det(\mathbf{B}) \cdot \mathbf{I}$, where $\mathbf{I}$ is the identity matrix. Therefore, if $\langle \mathbf{x}, \mathbf{y} \rangle \in S$, the decryption algorithm will correctly output $\langle \mathbf{x}, \mathbf{y} \rangle$.

Security. To prove security of Π_{ipe} in the generic group model, we construct a simulator which, given only the inner products of the vectors corresponding to the key generation and encryption queries, is able to correctly simulate the real distribution of the secret keys and ciphertexts.

Theorem 3.1. *The inner product encryption scheme Π_{ipe} is SIM-secure in the generic group model.*

Proof. In the following, we construct a generic bilinear group simulator $\mathcal{S}$ that interacts with the adversary $\mathcal{A}$ such that the distribution of responses in the real scheme is computationally indistinguishable from that in the ideal scheme. We begin by giving a high-level description of the simulator. We then proceed to give a formal specification of the formal variables we use in the proof, followed by a rigorous specification of the simulator. In the generic bilinear group model (Definition 2.7, the simulator $\mathcal{S}$ also simulates the responses of the generic bilinear group oracle.

Sketch of the simulator. The simulator must respond to the key generation and encryption queries (Definition 2.4) as well as the generic bilinear group operation queries (Definition 2.7). For each key generation and encryption query, the simulator responds with a fresh handle corresponding to each group element in the secret key and the ciphertext. Similarly, for each generic group oracle query, the simulator responds with a fresh handle for the resulting group element. The simulator maintains a table that maps handles to the formal polynomials the adversary forms via its queries (Remark 2.8). The major challenge in the simulation is in answering the zero-test queries. To consistently answer each zero-test query, the simulator first looks up the corresponding formal polynomial in its table and decomposes it into a "canonical" form, that is, as a sum of "honest" and "dishonest" components. The honest components correspond to a proper evaluation of the inner product while the dishonest components include any remaining terms after the valid inner product relations have been factored out. We argue, using properties of determinants, that if a query polynomial contains a dishonest component, then the resulting polynomial cannot be the identically zero polynomial over the formal variables corresponding to the randomly sampled elements in $\mathbf{B}$. Then, by the Schwartz-Zippel lemma, the simulator can correctly (with overwhelming probability) output "nonzero" in these cases. Finally, in the ideal experiment, the simulator is given the value of the inner product between each pair of vectors the adversary submits to the key generation and encryption oracles, so it can make the corresponding substitutions for the honest inner product relations and thus, correctly simulate the outputs of the zero-test oracle.

We now describe the formal variables in our construction and then give the full specification of the simulator. We conclude the proof by showing that our ideal-world simulator correctly simulates the real distribution.

Definition 3.2 (Formal Variables for Π_{ipe})**.** Let Q be the total number of queries made by the adversary $\mathcal{A}$ to each of the oracle query types. We define two (non-disjoint) sets of formal variables:

$$\mathcal{R} = \left\{\hat{\mathsf{d}}\right\} \cup \left\{\hat{\alpha}^{(i)}, \hat{\beta}^{(i)}\right\}_{i\in[Q]} \cup \left\{\hat{s}_\ell^{(i)}, \hat{t}_\ell^{(i)}\right\}_{i\in[Q],\ell\in[n]}$$

and

$$\mathcal{T} = \left\{\hat{\alpha}^{(i)}, \hat{\beta}^{(i)}\right\}_{i\in[Q]} \cup \left\{\hat{x}_k^{(i)}, \hat{y}_k^{(i)}\right\}_{i\in[Q],k\in[n]} \cup \left\{\hat{b}_{k,\ell}, \hat{b}^\star_{k,\ell}\right\}_{k,\ell\in[n]}.$$

The universe $\mathcal{U}$ of formal variables is defined to be the union $\mathcal{U} = \mathcal{R} \cup \mathcal{T}$.

Modeling the formal variables. In our security proof, the handles given to the adversary by the key generation and encryption oracles represent polynomials entirely expressible in the formal variables in $\mathcal{R}$. In other words, the formal polynomials the adversary submits to the zero-test oracle are all formal polynomials over the variables in $\mathcal{R}$. To answer the zero-test queries, the simulator performs a series of substitutions to re-express the adversary's query polynomial as a polynomial over the formal variables in $\mathcal{T}$. We now describe more explicitly how we model the formal variables in our scheme.

First, recall from our construction that $\mathbf{B}$ and $\mathbf{B}^\star$ are chosen such that $\mathbf{B}(\mathbf{B}^\star)^\top = \det(\mathbf{B}) \cdot \mathbf{I}$. In particular, this means that

$$b^\star_{k,\ell} = (-1)^{k+\ell} \det(\mathbf{B}_{k,\ell}),$$

where we write $b^\star_{k,\ell}$ to denote the $(k,\ell)^{\text{th}}$ entry in $\mathbf{B}^\star$ and $\mathbf{B}_{k,\ell}$ to denotes the (k,ℓ) minor of $\mathbf{B}$ (i.e., the matrix formed by taking $\mathbf{B}$ and removing it k^{th} row and ℓ^{th} column). Expanding the determinant as a sum over permutations, we can express $b^\star_{k,\ell}$ as a polynomial in the components of $\mathbf{B}$. First, for an index $k \in [n]$, let $S_{\neg k}$ denote the set $[n] \setminus [k]$. Then, we have

$$b^\star_{k,\ell} = (-1)^{k+\ell} \det(\mathbf{B}_{k,\ell}) = \sum_{\sigma: S_{\neg k} \to S_{\neg \ell}} \mathsf{sgn}(\sigma, k, \ell) \prod_{r \in S_{\neg k}} b_{r,\sigma(r)}, \tag{3.1}$$

where $\sigma : S_{\neg k} \to S_{\neg \ell}$ ranges over the set of bijections from $S_{\neg k}$ to $S_{\neg \ell}$, sgn is a sign function that maps its inputs to the set $\{-1, 1\}$, and $b_{k,\ell}$ is the $(k,\ell)^{\text{th}}$ component of $\mathbf{B}$. We collapse the sign of the monomials into the sgn function because our argument will not require explicit modeling of the signs.

For each $k, \ell \in [n]$, the simulator models the elements $\hat{b}_{k,\ell}, \hat{b}^\star_{k,\ell}$ to be formal symbols representing the variables $b_{k,\ell}$ and $b^\star_{k,\ell}$ from the construction, subject to the relation in Equation (3.1). In other words, the simulator models $\hat{b}^\star_{k,\ell}$ as a formal polynomial in the $\hat{b}_{i,j}$'s:

$$\hat{b}^\star_{k,\ell} = \sum_{\sigma: S_{\neg k} \to S_{\neg \ell}} \mathsf{sgn}(\sigma, k, \ell) \prod_{r \in S_{\neg k}} \hat{b}_{r,\sigma(r)}.$$

Also, for each $i \in [Q]$ and $k \in [n]$, the simulator models the elements $\hat{\alpha}^{(i)}, \hat{\beta}^{(i)}, \hat{x}_k^{(i)}, \hat{y}_k^{(i)}$ as formal symbols representing the variables $\alpha^{(i)}, \beta^{(i)}, x_k^{(i)}, y_k^{(i)}$ from the real scheme. Furthermore, for each $i \in [Q]$ and $\ell \in [n]$, the elements $\hat{s}_\ell^{(i)}, \hat{t}_\ell^{(i)}$ will be modeled as formal polynomials of the form

$$\hat{s}_\ell^{(i)} = \sum_{k\in[n]} \hat{x}_k^{(i)} \cdot \hat{b}_{k,\ell} \qquad \text{and} \qquad \hat{t}_\ell^{(i)} = \sum_{k\in[n]} \hat{y}_k^{(i)} \cdot \hat{b}^\star_{k,\ell}. \tag{3.2}$$

We will also use the formal symbol $\hat{\mathbf{B}}$ as shorthand to represent the following matrix of formal variables:

$$\hat{\mathbf{B}} = \begin{bmatrix} \hat{b}_{1,1} & \cdots & \hat{b}_{1,n} \\ \vdots & \ddots & \vdots \\ \hat{b}_{n,1} & \cdots & \hat{b}_{n,n} \end{bmatrix}.$$

Finally, the simulator models the formal symbol $\hat{\mathsf{d}}$ to be $\det(\mathbf{B})$. In particular, $\hat{\mathsf{d}} = \det(\hat{\mathbf{B}})$. Note that this means that $\hat{\mathsf{d}}$ is a formal polynomial of degree n over the formal variables in the matrix $\mathbf{B}$. In our description of the simulator, it will be the case that the only polynomials the adversary can form are those in the collection $\mathcal{R}$ (Definition 3.2).

Specification of the simulator. Fix an efficient adversary $\mathcal{A}$ that makes at most $Q = \mathrm{poly}(\lambda)$ queries to the key generation, encryption, and group operation oracles. At the beginning of the security game, the simulator initializes an empty collection of mappings $\mathcal{C}_{\mathsf{ip}}$ along with three empty tables $T_1, T_2, T_{\mathrm{T}} \leftarrow \{\}$ which each map handles in $\{0,1\}^\lambda$ to formal polynomials over the variables of $\mathcal{R}$. The simulator's state consists of the mappings $\mathcal{C}_{\mathsf{ip}}$ and the tables T_1, T_2, T_{T}. In the following description, we will implicitly assume that the simulator updates its internal state in response to each query. For $i \in [Q]$, on the i^{th} oracle query made by the adversary $\mathcal{A}$, the simulator $\mathcal{S}$ responds as follows.

- **Key generation queries.** On input a vector $\mathbf{x}^{(i)} \in \mathbb{Z}_q^n$ to $\mathcal{O}'_{\mathsf{KeyGen}}(\cdot)$, the simulator $\mathcal{S}$ receives as input a new collection $\mathcal{C}'_{\mathsf{ip}}$ of inner products and updates $\mathcal{C}_{\mathsf{ip}} \leftarrow \mathcal{C}'_{\mathsf{ip}}$. Then, $\mathcal{S}$ generates a fresh handle $h \xleftarrow{\mathrm{R}} \{0,1\}^\lambda$ and adds the mapping $h \mapsto \hat{\alpha}^{(i)} \cdot \hat{\mathsf{d}}$ to T_1. Next, for each $\ell \in [n]$, $\mathcal{S}$ generates a fresh handle $h_\ell \xleftarrow{\mathrm{R}} \{0,1\}^\lambda$ and adds the mapping $h_\ell \mapsto \hat{\alpha}^{(i)} \cdot \hat{s}_\ell^{(i)}$ to T_1. Finally, $\mathcal{S}$ sets $K_1 = h$, $K_2 = (h_1, \ldots, h_n)$, and responds with a secret key $\mathsf{sk} = (K_1, K_2)$.

- **Encryption queries.** On input a vector $\mathbf{y}^{(i)} \in \mathbb{Z}_p^n$ to $\mathcal{O}'_{\mathsf{Enc}}(\cdot)$, the simulator $\mathcal{S}$ receives as input a new collection $\mathcal{C}'_{\mathsf{ip}}$ of inner products and updates $\mathcal{C}_{\mathsf{ip}} \leftarrow \mathcal{C}'_{\mathsf{ip}}$. Then, $\mathcal{S}$ generates a fresh handle $h \xleftarrow{\mathrm{R}} \{0,1\}^\lambda$ and adds the mapping $h \mapsto \hat{\beta}^{(i)}$ to T_2. Next, for each $\ell \in [n]$, $\mathcal{S}$ generates a fresh handle $h_\ell \xleftarrow{\mathrm{R}} \{0,1\}^\lambda$ and adds the mapping $h_\ell \mapsto \hat{\beta}^{(i)} \cdot \hat{t}_\ell^{(i)}$ to T_2. Finally, $\mathcal{S}$ sets $C_1 = h$ and $C_2 = (h_1, \ldots, h_n)$, and responds with $\mathsf{ct} = (C_1, C_2)$.

- **Addition oracle queries.** Given two handles $h_1, h_2 \in \{0,1\}^\lambda$, $\mathcal{S}$ checks to see if there exist formal polynomials p_1 and p_2 for which the mappings $h_1 \mapsto p_1$ and $h_2 \mapsto p_2$ are present within the same table T_τ for $\tau \in \{1, 2, \mathrm{T}\}$ (returning $\perp$ if this is not the case). Then, it generates a fresh handle $h \leftarrow \{0,1\}^\lambda$, adds the mapping $h \mapsto (p_1 + p_2)$ to T_τ, and returns h.

- **Pairing oracle queries.** Given two handles $h_1, h_2 \in \{0,1\}^\lambda$, $\mathcal{S}$ checks to see if there exist formal polynomials p_1 and p_2 for which the mappings $h_1 \mapsto p_1$ is in T_1 and $h_2 \mapsto p_2$ is in T_2 (returning $\perp$ if this is not the case). Then, it generates a fresh handle $h \leftarrow \{0,1\}^\lambda$, adds the mapping $h \mapsto (p_1 \cdot p_2)$ to T_{T}, and returns h.

- **Zero-test oracle queries.** Given a handle $h \in \{0,1\}^\lambda$, $\mathcal{S}$ checks that there is a formal polynomial p and a tag $\tau \in \{1, 2, \mathrm{T}\}$ for which the mapping $h \mapsto p$ is present in T_τ (returning $\perp$ if this is not the case). The simulator then proceeds as follows.

1. The simulator "canonicalizes" the formal polynomial p by expressing it as a sum of products of formal variables in $\mathcal{T}$ with a $\mathrm{poly}(\lambda)$ number of terms. If p is identically zero, then the simulator outputs "zero".
2. If $\tau \neq \mathrm{T}$, the simulator outputs "non-zero".
3. Otherwise, the simulator decomposes p by grouping its terms into the form

$$p = \sum_{i,j \in [Q]} \hat{\alpha}^{(i)} \hat{\beta}^{(j)} \cdot \left(p_{i,j} \left(\hat{\mathsf{d}}, \left\{ \hat{s}_\ell^{(i)}, \hat{t}_\ell^{(j)} \right\}_{\ell \in [n]} \right) + f_{i,j} \left(\hat{\mathsf{d}}, \left\{ \hat{s}_\ell^{(i)}, \hat{t}_\ell^{(j)} \right\}_{\ell \in [n]} \right) \right), \tag{3.3}$$

   where for each $i, j \in [Q]$, the formal polynomial $p_{i,j}$ is defined as

$$p_{i,j} = c_{i,j} \cdot \left(\sum_{\ell=1}^{n} \hat{s}_\ell^{(i)} \hat{t}_\ell^{(j)} - z_{i,j} \, \hat{\mathsf{d}} \right), \tag{3.4}$$

   where $z_{i,j} \in \mathbb{Z}_q$ is the scalar mapped by the pair (i, j) in $\mathcal{C}_{\mathsf{ip}}$, the scalar $c_{i,j} \in \mathbb{Z}_q$ is the coefficient (which could be 0) of the term $\hat{s}_1^{(i)} \hat{t}_1^{(j)}$, and $f_{i,j}$ consists of all remaining terms (if any).
4. For each $i, j \in [Q]$, if $f_{i,j}$ consists of any terms, then the simulator outputs "non-zero".
5. Otherwise, the simulator outputs "zero".

Correctness of the simulator. We first note that the simulator's responses to the key generation, encryption, and group oracle queries made by the adversary $\mathcal{A}$ are distributed identically as in the real experiment $\mathsf{Real}_{\mathcal{A}}(1^\lambda)$. Hence, it remains to show that $\mathcal{S}$ correctly simulates the responses to the zero-test queries. To do so, we give a step-by-step analysis of the correctness of the simulation for responses to the zero-test queries made by the adversary $\mathcal{A}$.

1. We first show that the canonicalization step at the beginning of the adversary's response procedure is efficient. First, the adversary can only obtain handles to new monomials by querying the key generation and encryption oracles. In both cases, the formal variables in the monomials the adversary obtains are contained in the set $\mathcal{R}$ (Definition 3.2). Thus, the only formal polynomials the adversary is able to construct via its queries to the generic group oracle are formal polynomials over the formal variables in $\mathcal{R}$. Next, since the adversary can make at most Q queries, the polynomial p it submits to the zero-test oracle have at most $\mathrm{poly}(Q)$ terms and degree at most 2.

   By applying the relations in Equation (3.2) and expanding the determinant $\hat{\mathsf{d}}$ as a formal polynomial (of degree n) in the $\hat{b}_{k,\ell}$'s, we can express the formal polynomial p as a formal polynomial over the formal variables in $\mathcal{T}$. Since p has degree at most 2 over the variables in $\mathcal{R}$, we conclude that p can be expanded as a sum of at most $\mathrm{poly}(Q, n)$ monomials over the formal variables in $\mathcal{T}$ and has degree at most $d = \mathrm{poly}(n)$. Since both the input polynomial and the expanded canonicalized polynomial are polynomially-sized, this process is efficient. Finally, if the canonicalized polynomial is the identically zero polynomial, then the simulator correctly outputs "zero".

2. Consider the case where $\tau = 1$. By construction, the only monomials the adversary obtains where $\tau = 1$ are the monomials it receives in response to key generation queries. Then, we can write the formal polynomial p as follows:

$$p = \sum_{i \in [Q]} \hat{\alpha}^{(i)} \left(c_0^{(i)} \hat{\mathsf{d}} + \sum_{\ell \in [n]} c_\ell^{(i)} \hat{s}_\ell^{(i)} \right) = \sum_{i \in [Q]} \hat{\alpha}^{(i)} \left(c_0^{(i)} \hat{\mathsf{d}} + \sum_{\ell \in [n]} c_\ell^{(i)} \sum_{k \in [n]} \hat{x}_k^{(i)} \hat{b}_{k,\ell} \right), \tag{3.5}$$

where $c_0^{(i)}, \ldots, c_n^{(i)} \in \mathbb{Z}_q$ are scalars. By admissibility, each of the adversary's queries $\mathbf{x}^{(i)}$ to the key generation oracle is non-zero. In other words, for all $i \in [Q]$, there exists some $k \in [n]$ such that $\hat{x}_k^{(i)} \neq 0$. Thus, for all $i \in [Q]$, the sum $\sum_{k \in [n]} \hat{x}_k^{(i)} \hat{b}_{k,\ell}$ is not the identically zero polynomial over the formal variables $\{\hat{b}_{k,\ell}\}_{k,\ell \in [n]}$. We conclude from Equation (3.5) that if p is not the identically zero polynomial, then it cannot be the identically zero polynomial over the formal variables $\{\hat{\alpha}^{(i)}\}_{i \in [Q]}$ and $\{\hat{b}_{k,\ell}\}_{k,\ell \in [n]}$ (after expanding $\hat{\mathsf{d}}$ as a formal polynomial of degree n in the $\hat{b}_{k,\ell}$'s). Importantly, this holds irrespective of the adversary's choice of queries $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(Q)}$ (provided they are all non-zero). Since the variables $\left\{\hat{\alpha}^{(i)}\right\}_{i \in [Q]}$ and $\{\hat{b}_{k,\ell}\}_{k,\ell \in [n]}$ are distributed uniformly and independently in the real game, and moreover, the polynomial p has degree $\mathrm{poly}(n) = \mathrm{poly}(\lambda)$ in those variables, we conclude by Schwartz-Zippel (Lemma 2.9) that p evaluates to non-zero in the real distribution with overwhelming probability. Correctness of the simulation then follows with overwhelming probability.

   The case for $\tau = 2$ follows analogously. In this case, we express p as a formal polynomial (of $\mathrm{poly}(n)$ degree) over the formal variables $\{\hat{\beta}^{(i)}\}_{i \in [Q]}$, $\{\hat{b}^{\star}_{k,\ell}\}_{k,\ell \in [n]}$, and $\{\hat{y}_k^{(i)}\}_{i \in [Q], k \in [n]}$. By the same argument as above, if $\mathbf{y}^{(i)} \neq \mathbf{0}$ for all $i \in [Q]$, then irrespective of the actual values of $\mathbf{y}^{(i)}$, the polynomial p is not the identically zero polynomial over the remaining formal variables $\{\hat{\beta}^{(i)}\}_{i \in [Q]}$, $\{\hat{b}^{\star}_{k,\ell}\}_{k,\ell \in [n]}$. In isolation, each of the variables $b^{\star}_{k,\ell}$ is distributed uniformly and independently in the real scheme, and so, correctness of the simulation again follows by Schwartz-Zippel.

3. We show that the decomposition performed by the simulator preserves the modeling between the formal symbols and the elements of $\mathbb{Z}_q$ they represent in the real distribution. First, note that we can always decompose the polynomial p into a sum of monomials. By construction, every monomial the adversary obtains from the key generation oracle include the variable $\hat{\alpha}^{(i)}$ for some $i \in [Q]$. Similarly, every monomial the adversary obtains from the encryption oracle includes the variable $\hat{\beta}^{(i)}$ for some $i \in [Q]$. Since polynomials in the target group (where $\tau = \mathrm{T}$) can only be formed by multiplying together two polynomials in each of the two base groups, it follows that each monomial in the target group must contain $\hat{\alpha}^{(i)}$ and $\hat{\beta}^{(j)}$ for some $i, j \in [Q]$. Hence, the simulator in this step simply groups like terms for each $i, j \in [Q]$ to obtain the decomposition in Equation (3.3). Note that this step is efficient since the canonicalized polynomial p only has $\mathrm{poly}(Q, n)$ monomials.

4. Fix a pair of indices $i, j \in [Q]$, and suppose that $f_{i,j}$ contains at least a single term. We argue that the polynomial $f_{i,j}$ cannot be the identically zero polynomial when expressed as a polynomial over the formal variables $\hat{b}_{k,\ell}$ (for $k, \ell \in [n]$). More importantly, this must hold irrespective of the adversary's choice of (admissible) queries $\left\{\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\right\}_{i \in [Q]}$. The claim then follows by the Schwartz-Zippel lemma. To do this, we first give a characterization lemma that classifies the polynomial $f_{i,j}$ into one of three possibilities:

Lemma 3.3. *The polynomial $f_{i,j}$ from Step 3 formed by the adversary $\mathcal{A}$ must either:*

- *contain a "cross-term" of the form $c \cdot \hat{s}_{\ell_1}^{(i)} \hat{t}_{\ell_2}^{(j)}$, where $c \in \mathbb{Z}_q$ is non-zero and $\ell_1 \neq \ell_2$,*
- *contain no cross-terms, but instead contain a "consistent" term of the form $c \cdot \hat{s}_{\ell}^{(i)} \hat{t}_{\ell}^{(j)}$ for some non-zero scalar $c \in \mathbb{Z}_q$ and $\ell \in [n]$, or*
- *is a polynomial in the terms $\hat{\mathsf{d}}$, $\hat{s}_{\ell}^{(i)}$, and $\hat{\mathsf{d}} \cdot \hat{t}_{\ell}^{(j)}$ for $\ell \in [n]$.*

In particular, $f_{i,j}$ cannot contain any terms of the form $c_1 \cdot \hat{t}_{\ell}^{(j)}$ or $c_2 \cdot \hat{\mathsf{d}} \cdot \hat{s}_{\ell}^{(i)}$, for any coefficients $c_1, c_2 \in \mathbb{Z}_q$ and $\ell \in [n]$.

This lemma follows immediately by inspection of the structure of $f_{i,j}$. We now give a high-level survey of our basic argument, covering each possible case for the polynomial $f_{i,j}$.

- First, we show that if $f_{i,j}$ contains any "cross-terms" of the form $c \cdot \hat{s}_{\ell_1}^{(i)} \hat{t}_{\ell_2}^{(j)}$, where $c \in \mathbb{Z}_q$ is non-zero and $\ell_1 \neq \ell_2$, then $f_{i,j}$ cannot be identically zero over the $\hat{b}_{k,\ell}$ variables, irrespective of the adversary's choice of (admissible) queries $\left\{\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\right\}_{i \in [Q]}$.
- Next, we consider the case where $f_{i,j}$ does not contain any cross-terms, but instead, contains a "consistent" term of the form $c \cdot \hat{s}_{\ell}^{(i)} \hat{t}_{\ell}^{(j)}$ for some non-zero scalar $c \in \mathbb{Z}_q$ and $\ell \in [n]$. We assume, towards a contradiction, that $f_{i,j}$ is the identically zero polynomial when instantiated with the adversary's queries $\left\{\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\right\}_{i \in [Q]}$ and making the substitutions from Equation (3.2). We show then that it must be the case that $f_{i,j}$ actually contains a linear combination of all of the $\hat{s}_{\ell}^{(i)} \hat{t}_{\ell}^{(j)}$ terms for each $\ell \in [n]$, together with the $\hat{\mathsf{d}}$ term. However, by construction of the simulator, this linear combination has already been factored out of $f_{i,j}$ in the previous step (Step 3). This yields the desired contradiction, and we conclude that $f_{i,j}$ cannot be the identically zero polynomial.
- For the final case, if $f_{i,j}$ contains neither a cross-term nor a consistent term, then it must not contain any terms of the form $\hat{s}_{\ell_1}^{(i)} \hat{t}_{\ell_2}^{(j)}$. In particular, it must be a non-zero polynomial in $\hat{\mathsf{d}}$, $\hat{\mathsf{d}} \cdot \hat{t}_{\ell}^{(j)}$, and $\hat{s}_{\ell}^{(i)}$ for $\ell \in [n]$. We then argue, by a similar argument used in Step 2, that by expanding each term into the underlying formal variables $\hat{b}_{k,\ell}$ for $k, \ell \in [n]$, each of the terms that make up $f_{i,j}$ are linearly independent. This allows us to conclude that $f_{i,j}$ cannot be identically zero.

We now describe the argument in detail. First, we rewrite $f_{i,j}$ as a polynomial over the formal variables $\hat{x}_k^{(i)}, \hat{y}_k^{(j)}, \hat{b}_{k,\ell}$ for $k, \ell \in [n]$, and then reason about the structure of the terms in $f_{i,j}$ to obtain the desired conclusion. We begin by expressing the quantities $\hat{s}_{\ell_1}^{(i)} \hat{t}_{\ell_2}^{(j)}$ as a polynomial in the $\hat{b}_{k,\ell}$'s by making the substitutions in Equations (3.1) and (3.2). For ease of notation, we absorb the sign of each term in the cofactor expansion of Equation (3.1) into an (unspecified[8]) function sgn with output space $\{-1, 1\}$, as they are not important in our argument. Then, we obtain

[8]Specifically, our argument only considers the types of monomials that appear in the cofactor expansion. Since every monomial appears at most once in the cofactor expansion, no cancellations occur in the signed expansion. Thus, disregarding the signs does not affect the correctness of the argument.

$$\begin{aligned}
\hat{s}_{\ell_1}^{(i)}\hat{t}_{\ell_2}^{(j)} &= \left(\sum_{k_1\in[n]} \hat{x}_{k_1}^{(i)} \cdot \hat{b}_{k_1,\ell_1}\right)\left(\sum_{k_2\in[n]} \hat{y}_{k_2}^{(j)} \cdot \hat{b}^{\star}_{k_2,\ell_2}\right) \\
&= \sum_{k_1,k_2\in[n]} \hat{x}_{k_1}^{(i)} \cdot \hat{y}_{k_2}^{(j)} \cdot \hat{b}_{k_1,\ell_1} \cdot \hat{b}^{\star}_{k_2,\ell_2} \\
&= \sum_{k_1,k_2\in[n]} \hat{x}_{k_1}^{(i)} \cdot \hat{y}_{k_2}^{(j)} \cdot \hat{b}_{k_1,\ell_1} \cdot \left(\sum_{\sigma:S_{\neg k_2}\to S_{\neg \ell_2}} \mathsf{sgn}(\sigma,k_1,k_2) \prod_{r\in S_{\neg k_2}} \hat{b}_{r,\sigma(r)}\right)
\end{aligned}$$

$$\hat{s}_{\ell_1}^{(i)}\hat{t}_{\ell_2}^{(j)} = \sum_{k_1,k_2\in[n]} \sum_{\sigma:S_{\neg k_2}\to S_{\neg \ell_2}} \hat{x}_{k_1}^{(i)} \cdot \hat{y}_{k_2}^{(j)} \cdot \hat{b}_{k_1,\ell_1} \cdot \mathsf{sgn}(\sigma,k_1,k_2) \prod_{r\in S_{\neg k_2}} \hat{b}_{r,\sigma(r)}. \tag{3.6}$$

Similarly, we can express $\hat{\mathsf{d}}$ as

$$\hat{\mathsf{d}} = \sum_{\sigma:[n]\to[n]} \prod_{r\in[n]} \hat{b}_{r,\sigma(r)} \cdot \mathsf{sgn}(\sigma),$$

where again, we absorb the signs into the sgn function.

To complete the proof, we perform a case-by-case analysis over the possible monomials in $f_{i,j}$. Our case-by-case analysis will proceed by examination of the structure of the determinant $\hat{\mathsf{d}}$. First, recall that $\hat{\mathbf{B}}$ is the following matrix of formal variables:

$$\hat{\mathbf{B}} = \begin{bmatrix} \hat{b}_{1,1} & \cdots & \hat{b}_{1,n} \\ \vdots & \ddots & \vdots \\ \hat{b}_{n,1} & \cdots & \hat{b}_{n,n} \end{bmatrix},$$

and that $\hat{\mathsf{d}} = \det(\hat{\mathbf{B}})$.

- **Case 1:** Suppose $f_{i,j}$ contains a scalar multiple of a cross-term, which is a term of the form $\hat{s}_{\ell_1}^{(i)}\hat{t}_{\ell_2}^{(j)}$, where $\ell_1 \neq \ell_2$. Consider its expansion into the sum of monomials given in Equation (3.6). By inspection, we see that each monomial in the expansion contains the product of $\hat{b}_{k_1,\ell_1}$ and $\hat{b}_{r,\ell_1}$ for some $r \neq k_2$. Moreover, none of the monomials in the expansion contains the variable $\hat{b}_{r,\ell_2}$ for all $r \in [n]$. In other words, each monomial in the expansion of $\hat{s}_{\ell_1}^{(i)}\hat{t}_{\ell_2}^{(j)}$ contains the product of exactly two (not necessarily distinct) variables in column ℓ_1 of $\hat{\mathbf{B}}$ but no variables in column ℓ_2. Furthermore, observe that in the expansion of all consistent terms, (that is, terms of the form $\hat{s}_{\ell}^{(i)}\hat{t}_{\ell}^{(j)}$ for some $\ell \in [n]$, and the term $\hat{\mathsf{d}}$), each monomial contains exactly one variable from each column of $\hat{\mathbf{B}}$. This means that none of the products in the expansion of $\hat{s}_{\ell_1}^{(i)}\hat{t}_{\ell_2}^{(j)}$ can be canceled by monomials appearing in in the expansion the other terms in $f_{i,j}$. Thus, $f_{i,j}$ cannot be the identically zero polynomial in the formal variables in $\mathcal{T}$.

- **Case 2:** We can now assume without loss of generality that $f_{i,j}$ is a linear combination of the consistent terms $\hat{s}_\ell^{(i)}\hat{t}_\ell^{(j)}$ for some $\ell \in [n]$, along with the term $\hat{\mathsf{d}}$. Recall, from the definition of $p_{i,j}$ in Equation (3.4), that $f_{i,j}$ cannot contain a non-zero multiple of $\hat{s}_1^{(i)}\hat{t}_1^{(j)}$. Thus, in our analysis below, $\ell \neq 1$. We consider two subcases based on the vectors $\mathbf{x}^{(i)}$ and $\mathbf{y}^{(j)}$, neither of which can be the all-zeroes vector.
  (a) Suppose $\mathbf{x}^{(i)}$ and $\mathbf{y}^{(j)}$ are non-zero in exactly one coordinate k—that is, $x_k^{(i)} \neq 0$ and $y_k^{(j)} \neq 0$. Then, note that for each $\ell \in [n]$, the expansion from Equation (3.6) of each consistent term $\hat{s}_\ell^{(i)}\hat{t}_\ell^{(j)}$ is a sum of monomials which each contain $\hat{b}_{k,\ell}$, but *do not* contain $\hat{b}_{k,\ell^*}$ for any $\ell^* \neq \ell$. Hence, if $\hat{\mathsf{d}}$ is not present in $f_{i,j}$, then these monomials can never be cancelled out. Thus, $f_{i,j}$ cannot be identically zero. Alternatively, if $\hat{\mathsf{d}}$ is present in $f_{i,j}$, then note that the expansion of $\hat{\mathsf{d}}$ consists of a product containing the variable $\hat{b}_{k,1}$, but since the term $\hat{s}_1^{(i)}\hat{t}_1^{(j)}$ is absent in $f_{i,j}$, we again have that $f_{i,j}$ cannot be identically zero in this case.
  (b) On the other hand, suppose there exist $k_1, k_2 \in [n]$ with $k_1 \neq k_2$ and for which $x_{k_1}^{(i)} \neq 0$ and $y_{k_2}^{(j)} \neq 0$. Then, the expansion from Equation (3.6) of the term $\hat{s}_\ell^{(i)}\hat{t}_\ell^{(j)}$ contains a monomial that contains the product of both $\hat{b}_{k_1,\ell}$ and $\hat{b}_{k_1,1}$ (here, we have used the fact that $\ell \neq 1$), but *does not* contains a monomial that contains the product of the variables $\hat{b}_{k_1,\ell^*}$ and $\hat{b}_{k_1,1}$ for any $\ell^* \neq \ell$. Furthermore, the term $\hat{\mathsf{d}}$ expands into a sum of monomials that each contain exactly one variable from each row of $\hat{\mathbf{B}}$, and so $\hat{\mathsf{d}}$ cannot cancel any of these monomials. Hence, these monomials cannot be cancelled in the expansion of $f_{i,j}$, and therefore $f_{i,j}$ cannot be identically zero. Note that in the honest evaluation, the monomials that contain the product $\hat{b}_{k_1,\ell}\hat{b}_{k_1,1}$ are cancelled out by monomials appearing in the expansion of $\hat{s}_1^{(i)}\hat{t}_1^{(j)}$. However, these monomials are not present in the expansion of $f_{i,j}$.
- **Case 3:** Finally, we have that case where $f_{i,j}$ contains neither a cross-term nor a consistent term. Then, it must not contain any terms of the form $\hat{s}_{\ell_1}^{(i)}\hat{t}_{\ell_2}^{(j)}$, and instead, must be a non-zero polynomial in $\hat{\mathsf{d}}$, $\hat{s}_\ell^{(i)}$, and $\hat{\mathsf{d}} \cdot \hat{t}_\ell^{(j)}$ for $\ell \in [n]$. Explicitly, the polynomial $f_{i,j}$ can be written as

$$f_{i,j} = c_1 \cdot \hat{\mathsf{d}} + \sum_{\ell \in [n]} c_{2,\ell} \cdot \hat{s}_\ell^{(i)} + \sum_{\ell \in [n]} c_{3,\ell} \cdot \hat{\mathsf{d}} \cdot \hat{t}_\ell^{(j)},$$

  for scalars $c_1, c_{2,1}, \ldots, c_{2,n}, c_{3,1}, \ldots, c_{3,n} \in \mathbb{Z}_q$, where at least one of these coefficients must be non-zero. Applying the substitutions from Equation (3.2) and expanding the determinant as a polynomial over $\hat{b}_{k,\ell}$ for $k, \ell \in [n]$, we conclude that $\hat{\mathsf{d}}$ is a polynomial of degree n, the terms $\hat{s}_\ell^{(i)}$ are polynomials of degree 1, and the terms $\hat{\mathsf{d}} \cdot \hat{t}_\ell^{(j)}$ or polynomials of degree $(2n-1)$ over the formal variables $\hat{b}_{k,\ell}$. This step relies on the fact that the adversary's queries $\mathbf{x}, \mathbf{y}$ are non-zero, and thus, irrespective of the actual values of $\mathbf{x}, \mathbf{y}$, the expansion of $\hat{s}_\ell^{(i)}$ and $\hat{t}_\ell^{(j)}$ as a function of $\hat{b}_{k,\ell}$ yields a polynomial that is not identically zero. Finally, all of the monomials in the summation are linearly independent over the polynomial ring $\mathbb{Z}_q[\hat{b}_{1,1}, \ldots, \hat{b}_{n,n}]$, and so, since at least one of the coefficients is non-zero, we conclude that $f_{i,j}$ is not identically zero.

We have shown in all cases that $f_{i,j}$ cannot be identically zero over the formal variables $\hat{b}_{k,\ell}$ for $k, \ell \in [n]$ irrespective of the adversary's queries (provided they are non-zero). Moreover, $f_{i,j}$ can always be expressed as a polynomial of degree $\mathrm{poly}(Q, n)$, so the simulator correctly outputs "non-zero" with overwhelming probability.

5. Finally, if the simulator has reached this step, then this means that for all $i, j \in [Q]$, the polynomial $f_{i,j}$ is identically zero. Note that we can express $p_{i,j}$ as

$$\begin{aligned} p_{i,j} &= c_{i,j} \cdot \left(\sum_{\ell \in [n]} \hat{s}_\ell^{(i)} \hat{t}_\ell^{(j)} - z\, \hat{\mathsf{d}} \right) \\ &= c_{i,j} \cdot \left(\sum_{\ell \in [n]} \sum_{k_1, k_2 \in [n]} \hat{x}_{k_1}^{(i)} \cdot \hat{y}_{k_2}^{(j)} \cdot \hat{b}_{k_1,\ell_1} \cdot \hat{b}^\star_{k_2,\ell_2} - z\, \hat{\mathsf{d}} \right) \\ &= c_{i,j} \cdot \left(\sum_{k_1, k_2 \in [n]} \hat{x}_{k_1}^{(i)} \cdot \hat{y}_{k_2}^{(j)} \cdot \left(\sum_{\ell \in [n]} \hat{b}_{k_1,\ell_1} \cdot \hat{b}^\star_{k_2,\ell_2} \right) - z\, \hat{\mathsf{d}} \right) \end{aligned}$$

Now, instantiating the formal variables with the values they represent in the real distribution, and using the fact that $\mathbf{B}(\mathbf{B}^\star)^\top = \det(\mathbf{B}) \cdot \mathbf{I}$, we have

$$p_{i,j} = c_{i,j} \cdot \left(\sum_{k \in [n]} x_k^{(i)} \cdot y_k^{(i)} \cdot \det(\mathbf{B}) - z_{i,j} \cdot \det(\mathbf{B}) \right).$$

Since $z_{i,j} \in \mathbb{Z}_q$ is the scalar mapped by the pair (i, j) in the collection of inner product mappings $\mathcal{C}_{\mathsf{ip}}$, this means that $z_{i,j} = \langle \mathbf{x}^{(i)}, \mathbf{y}^{(j)} \rangle$, and so

$$p_{i,j} = c_{i,j} \cdot \det(\mathbf{B}) \cdot 0 = 0.$$

Thus, p can be rewritten as

$$p = \sum_{i,j \in [Q]} \hat{\alpha}^{(i)} \hat{\beta}^{(j)} \cdot (0 + 0),$$

and so p is identically zero. The simulated and the real distribution are identical in this case.

We have shown that the distribution of the simulated responses and the real responses are statistically indistinguishable, which concludes the proof. □

4 Applications

In this section, we describe several applications of function-hiding IPE to biometric authentication and performing nearest-neighbor searches on an encrypted database. Then, in Section 4.2, we show how to construct two-input functional encryption for small domains from any function-hiding inner product encryption scheme.

4.1 Direct Applications of Function-Hiding IPE

We begin with several direct applications of function-hiding IPE.

Biometric authentication. Suppose an organization wants to deploy a biometric-based authentication system (e.g., fingerprint readers, iris scanners) to restrict access to certain areas within a complex. The biometric scanner is interfaced to an external authentication server that enforces the authorization policies. By offloading the authentication to a central server, it is no longer necessary for *every* biometric scanner to store the list of employee biometric signatures or their authorization policies. However, as with password-based authentication, it is a security risk to store each employee's biometric information in the clear on the server. In password-based authentication, the server typically stores a salted hash of each user's password, which allows it to check whether or not a user has provided the correct password without needing to store the user's password in the clear. In contrast to passwords, biometrics are noisy by nature. In the biometrics setting, authentication should succeed when the provided biometric is "close" to a user's stored credential. Consequently, hash-based methods are inappropriate in our setting. A better approach computes a Hamming distance between the biometric and a user's stored credential, where authentication passes only if this Hamming distance is small.

Inner product encryption provides an efficient way to compute Hamming distances between pairs of secret vectors. Given two binary vectors $\mathbf{x}, \mathbf{y} \in \{0,1\}^n$, let $\mathbf{x}', \mathbf{y}' \in \{-1,1\}^n$ be the vectors where each 0-entry in $\mathbf{x}$ and $\mathbf{y}$ is mapped to -1 in $\mathbf{x}'$ and $\mathbf{y}'$, and each 1-entry of $\mathbf{x}$ and $\mathbf{y}$ is mapped to 1 in $\mathbf{x}'$ and $\mathbf{y}'$, respectively. Then, by construction, $\langle \mathbf{x}', \mathbf{y}' \rangle = n - 2 \cdot d(\mathbf{x}, \mathbf{y})$, where $d(\mathbf{x}, \mathbf{y})$ is the Hamming distance between $\mathbf{x}$ and $\mathbf{y}$. Thus, given only the encryptions of $\mathbf{x}$ and $\mathbf{y}$, a decryptor can compute their Hamming distance using only the public parameters and without learning anything else about $\mathbf{x}$ and $\mathbf{y}$.

In our biometric authentication example, each biometric scanner is given the master secret key for a function-hiding IPE scheme. The authentication server stores an encryption of each user's biometric under the master secret key (but does not store the master secret key itself). When an employee tries to authenticate using a biometric scanner, the scanner reads the employee's biometric, encrypts it using the secret key, and sends it to the authentication server. The server computes the Hamming distance of the encrypted biometric with the stored biometric for the employee. Authentication succeeds if the resulting Hamming distance is small. Since the authentication server only stores encrypted credentials, a compromise of the authentication server does not result in a compromise of any employees' biometric information.

Nearest-neighbor search on encrypted data. Another application of inner product encryption is in performing nearest-neighbor search over an encrypted database. A simple way of measuring document similarity is to first embed the documents into an Euclidean space and then measure the ℓ_2-distance between the vectors corresponding to the documents. Suppose an organization has an encrypted set of documents and wants to allow employees to search for similar documents. With each document, the server stores an encryption of the vector representation of the document. Then, each employee who is authorized to search for documents in the database is given the master secret key for the IPE scheme. When an employee wants to find the set of documents that most closely matches her query, she first projects her query into the feature space, encrypts the resulting query vector, and sends the encrypted query vector to the database. The database then computes the ℓ_2-distance between the query vector and each document, and returns the set of documents with the smallest ℓ_2-distance (i.e., the nearest neighbors).

Using an IPE scheme, it is straightforward to construct an encryption scheme that allows a decryptor to compute the ℓ_2-distance between two encrypted vectors. Specifically, given two vectors

$\mathbf{x}, \mathbf{y} \in \mathbb{Z}_q^n$, their ℓ_2-distance is given by $\|\mathbf{x}-\mathbf{y}\|_2 = \|\mathbf{x}\|_2 - 2\langle \mathbf{x}, \mathbf{y}\rangle + \|\mathbf{y}\|_2$. Now, define the vectors $\mathbf{x}' = (\|\mathbf{x}\|_2, -2x_1, \ldots - 2x_n, 1) \in \mathbb{Z}_q^{n+2}$ and $\mathbf{y}' = (1, y_1, \ldots y_n, \|\mathbf{y}\|_2) \in \mathbb{Z}_q^{n+2}$. By construction, we have that $\langle \mathbf{x}', \mathbf{y}'\rangle = \|\mathbf{x}-\mathbf{y}\|_2$. Thus, an inner product encryption scheme yields a scheme that supports computing the ℓ_2-distance between encrypted vectors, which yields a solution for nearest-neighbor search over encrypted documents.

Secure linear regression. Linear regression is an indispensable tool in statistical analysis. Given a sequence of points $(x_1, y_1), \ldots, (x_n, y_n)$, the goal is to fit a line $y = \mathsf{m}x + \mathsf{c}$ for which the errors $e_i = y_i - (\mathsf{m}x_i + \mathsf{c})$ for $i \in [n]$ are minimized. In the least-squares approach, setting the parameters as

$$\mathsf{m} = \frac{1}{n\sigma^2}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y}) \qquad \text{and} \qquad \mathsf{c} = \bar{y} - \mathsf{m}\bar{x}$$

minimizes the sum of squared errors $\sum_{i=1}^{n} {e_i}^2$, where $\bar{x}$ and $\bar{y}$ denote the means of $\{x_1, \ldots, x_n\}$ and $\{y_1, \ldots, y_n\}$, respectively, and σ is the standard deviation of $\{x_1, \ldots, x_n\}$.

Suppose there are two clients Alice and Bob who each hold a list of secret values $\mathcal{X} = (x_1, \ldots, x_n)$ and $\mathcal{Y} = (y_1, \ldots, y_n)$, respectively. They want to allow a data-mining server S to compute various aggregate statistics such as the mean, the standard deviation, and the linear regression coefficients on their joint data while preserving the privacy of individual data values, and without having to store or manipulate the data itself. We describe how to efficiently compute the regression parameters m and c using a function-hiding inner product encryption scheme Π_{ipe} as follows. First, a trusted third party generates the public parameters pp and master secret key msk by running the setup algorithm, sending msk to the clients A and B, and pp to the server S. Alice then constructs a vector $\mathbf{x} = (x_1 - \bar{x}, \ldots, x_n - \bar{x})$, and sends the mean of her values $\bar{x}$, the standard deviation σ, the total number of values n, and $\mathsf{sk}_{\mathbf{x}} \leftarrow \mathsf{IPE.KeyGen}(\mathsf{msk}, \mathbf{x})$ to S. Similarly, Bob constructs a vector $\mathbf{y} = (y_1 - \bar{y}, \ldots, y_n - \bar{y})$, and sends the mean of his values $\bar{y}$ and $\mathsf{ct}_{\mathbf{y}} \leftarrow \mathsf{IPE.Encrypt}(\mathsf{msk}, \mathbf{y})$ to S. Using the public parameters pp, the server S simply computes

$$\mathsf{m} \leftarrow \mathsf{IPE.Decrypt}(\mathsf{pp}, \mathsf{sk}_{\mathbf{x}}, \mathsf{ct}_{\mathbf{y}})/(n\sigma^2) \qquad \text{and} \qquad \mathsf{c} \leftarrow \bar{y} - \mathsf{m}\bar{x}.$$

The function-hiding property of the underlying inner product encryption scheme ensures that the server learns nothing more about the secret values held by Alice and Bob, other than their means $\bar{x}, \bar{y}$, and the standard deviation σ.

One important caveat to note in this procedure is that the decryption operation $\mathsf{IPE.Decrypt}$ from Section 3 requires solving the discrete logarithm problem in $\mathbb{G}_T$. Thus, S must have an *a priori* estimate of the range of possible values for m. In many real-world applications, the server S has prior knowledge about the distributions of $\mathcal{X}$ and $\mathcal{Y}$, and thus, can approximately bound the value of the regression coefficients between them. Alternatively, Alice and Bob can also quantize their input values to an appropriate precision so that decryption is always efficient for S.

4.2 General Two-Input Functional Encryption

Multi-input functional encryption (MIFE) was introduced by Goldwasser et al. [GGG+14] who gave constructions based on indistinguishability obfuscation [BGI+12, GGH+13]. However, given that the current state of candidate constructions for indistinguishability obfuscation are prohibitively inefficient, multi-input functional encryption is still considered a theoretical concept. Recently, Boneh et al. [BLR+15] showed how to construct multi-input functional encryption using multilinear

maps, resulting in a more efficient and "implementable" scheme. Unfortunately, numerous recent attacks on multilinear maps in the public-key setting [CHL+15, BWZ14, CGH+15, HJ15, CLR15, MF15, Cor15, CJL16, MSZ16] have raised some doubts over the security of constructions relying on these primitives. In this section, we show how a function-hiding inner production encryption scheme can be leveraged to build a single-key two-input functional encryption scheme. As described in Section 1, single-key two-input functional encryption can be used to build order-revealing, and more generally, property-preserving encryption.

The bounded-message setting. Recently, Ananth and Jain [AJ15] as well as Brakerski, Komargodski, and Segev [BKS15] describe an "arity-amplification" transformation for constructing multi-input functional encryption starting from any single-input functional encryption scheme. In the secret-key setting, they show that starting from the Gorbunov et al. [GVW12] functional encryption scheme, it is possible to obtain two-input (and more generally, multi-input) functional encryption from standard assumptions (the existence of one-way functions and low-depth PRGs.) in the *bounded-message* setting.[9] In the bounded-message setting, the adversary can only make an *a priori* bounded number Q of ciphertext queries. In the case of the Gorbunov et al. scheme, the size of the scheme parameters and ciphertexts both grow with $\Omega(Q^4D^2)$, where D is the depth of the circuit family needed to evaluate the function and apply the arity-amplification transformation. Subsequently, Goldwasser et al. [GKP+13] showed how to construct succinct *single-key* functional encryption in the bounded collusion setting from the Learning with Errors (LWE) assumption. To extend their construction to multiple key queries (needed for the arity-amplification transformation for multi-input functional encryption), they apply the generic conversion from [GVW12], thus resulting in ciphertext sizes that still grow with $\Omega(Q^4)$, but independent of the depth D.

The small-domain setting. We consider a new setting for secret-key multi-input functional encryption, which we call the *small-domain* setting.[10] In the (single-key) small-domain setting, the security experiment places no restrictions on the number of ciphertexts the adversary can request,[11] but instead requires that the size of the plaintext space be polynomial in the security parameter λ. In contrast, the bounded-message setting allows large plaintext spaces, but there is an *a priori* bound on the number of ciphertexts that can be given out. In our new small-domain setting, the message-space must be polynomial-sized, but in exchange, we can support an *a priori* unbounded number of encryptions under an indistinguishability-based notion of security and in the single-key setting. This is advantageous when the same plaintext is encrypted multiple times under a function that does not reveal whether two ciphertexts encrypt the same plaintext.

In this section, we show how to construct a single-key two-input functional encryption (TIFE) scheme in the secret-key setting from function-hiding inner product encryption. Our construction

[9] Originally, Gorbunov et al. [GVW13] showed that their scheme is secure in the *bounded-collusion* setting—that is, secure against adversaries that only make an *a priori* bounded number of *key generation queries*. However, after applying the arity-amplification transformation of [AJ15, BKS15] to obtain a two-input functional encryption scheme, each encryption query requires making a key generation query to the underlying scheme. As a result, there is a limit on *both* the number of key generation and the number of encryption queries the resulting scheme can support.

[10] This setting has also been studied in the specific context of order-revealing encryption [LW16, JP16].

[11] Boneh et al. [BSW11] showed that simulation-based notions of security are impossible in the setting where the adversary can make adaptive key generation queries and an *a priori* unbounded number of ciphertext queries. However, their lower bound does not extend to indistinguishability-based notions of security or to the setting where the adversary cannot make adaptive key generation queries.

is a direct application of inner product encryption, and as a result, our ciphertext sizes are linear in the size of the domain, and *independent* of both the complexity of the function (so long as the function is efficiently computable) and the number of ciphertext queries the adversary makes. We now define the syntax and correctness requirements on a single-key two-input functional encryption scheme in the secret-key setting.

Definition 4.1 (Single-Key Two-Input Functional Encryption in the Secret-Key Settting)**.** For a security parameter λ and an efficiently computable function $f : [N] \times [N] \to \mathbb{Z}_q$, a *secret-key two-input functional encryption scheme for f* is a tuple of algorithms $\Pi = (\mathsf{TIFE.Setup}, \mathsf{TIFE.Encrypt_L}, \mathsf{TIFE.Encrypt_R}, \mathsf{TIFE.Decrypt})$ with the following properties.

- $\mathsf{TIFE.Setup}(1^\lambda) \to (\mathsf{pp}, \mathsf{sk})$. On input the security parameter λ, the setup algorithm outputs the public parameters pp and the secret key sk.
- $\mathsf{TIFE.Encrypt_L}(\mathsf{sk}, x) \to \mathsf{ct_L}$. On input the secret key sk and a message $x \in [N]$, the left encryption algorithm outputs a ciphertext $\mathsf{ct_L}$.
- $\mathsf{TIFE.Encrypt_R}(\mathsf{sk}, y) \to \mathsf{ct_R}$. On input the secret key sk and a message $y \in [N]$, the right encryption algorithm outputs a ciphertext $\mathsf{ct_R}$.
- $\mathsf{TIFE.Decrypt}(\mathsf{pp}, \mathsf{ct_L}, \mathsf{ct_R}) \to z$. On input the public parameters pp and two ciphertexts $\mathsf{ct_L}$ and $\mathsf{ct_R}$, the decrypt algorithm outputs an element $z \in \mathbb{Z}_q$.

Definition 4.2 (Correctness)**.** A two-input functional encryption scheme is *correct* if for a security parameter $\lambda \in \mathbb{N}$, all messages $x, y \in [N]$, $(\mathsf{pp}, \mathsf{sk}) \leftarrow \mathsf{TIFE.Setup}(1^\lambda)$, $\mathsf{ct_L} \leftarrow \mathsf{TIFE.Encrypt_L}(\mathsf{sk}, x)$, and $\mathsf{ct_R} \leftarrow \mathsf{TIFE.Encrypt_R}(\mathsf{sk}, y)$, we have that

$$\Pr\left[\mathsf{TIFE.Decrypt}(\mathsf{pp}, \mathsf{ct_L}, \mathsf{ct_R}) = f(x, y)\right] = 1 - \mathsf{negl}(\lambda).$$

Two-input functional encryption security. We define security for two-input functional encryption in the context of the following experiment between a challenger and an adversary $\mathcal{A}$ that can make left encryption and right encryption oracle queries.

Definition 4.3 (Experiment $\mathsf{Expt}_b^{\mathsf{tife}}$)**.** For $b \in \{0, 1\}$, we have that the challenger samples the public parameters and secret key $(\mathsf{pp}, \mathsf{sk}) \leftarrow \mathsf{TIFE.Setup}(1^\lambda)$, and sends pp to the adversary $\mathcal{A}$. It then responds to oracle query type by $\mathcal{A}$ as follows:

- **Left encryption oracle.** On input two messages $x_0, x_1 \in [N]$, the challenger computes and outputs $\mathsf{ct_L} \leftarrow \mathsf{TIFE.Encrypt_L}(\mathsf{sk}, x_b)$.
- **Right encryption oracle.** On input two messages $y_0, y_1 \in [N]$, the challenger computes and outputs $\mathsf{ct_R} \leftarrow \mathsf{TIFE.Encrypt_R}(\mathsf{sk}, y_b)$.

Eventually, $\mathcal{A}$ outputs a bit b', which is also the output of the experiment. We denote this $\mathsf{Expt}_b^{\mathsf{tife}}(\mathcal{A})$.

Definition 4.4 (Admissibility)**.** For an adversary $\mathcal{A}$, let Q_L and Q_R be the total number of left encryption and right encryption oracle queries made by $\mathcal{A}$, respectively. For $b \in \{0, 1\}$, let $x_b^{(1)}, \ldots, x_b^{(Q_\mathsf{L})} \in [N]$ and $y_b^{(1)}, \ldots, y_b^{(Q_\mathsf{R})} \in [N]$ be the corresponding messages that $\mathcal{A}$ submits to the oracles. We say that $\mathcal{A}$ is *admissible* if for all $i \in [Q_\mathsf{L}]$ and $j \in [Q_\mathsf{R}]$, we have that

$$f\left(x_0^{(i)}, y_0^{(j)}\right) = f\left(x_1^{(i)}, y_1^{(j)}\right).$$

Definition 4.5 (Two-Input Functional Encryption Security)**.** We say that a two-input functional encryption scheme satisfies *indistinguishability under a chosen plaintext attack* (IND-CPA) security if for all efficient and admissible adversaries $\mathcal{A}$,

$$\left|\mathsf{Pr}[\mathsf{Expt}_0^{\mathsf{tife}}(\mathcal{A}) = 1] - \mathsf{Pr}[\mathsf{Expt}_1^{\mathsf{tife}}(\mathcal{A}) = 1]\right| = \mathsf{negl}(\lambda).$$

Construction. Fix a security parameter λ, a positive integer N, an efficiently computable two-input function $f : [N]\times[N] \to \mathbb{Z}_q$. We define $\Pi_{\mathsf{ipe}} = (\mathsf{IPE.Setup}, \mathsf{IPE.KeyGen}, \mathsf{IPE.Encrypt}, \mathsf{IPE.Decrypt})$ to be a function-hiding IPE scheme defined over the message space $\mathbb{Z}_q^{N+1}$. We construct a two-input functional encryption scheme $\Pi_{\mathsf{tife}} = (\mathsf{TIFE.Setup}, \mathsf{TIFE.Encrypt_L}, \mathsf{TIFE.Encrypt_R}, \mathsf{TIFE.Decrypt})$ as follows.

- $\mathsf{TIFE.Setup}(1^\lambda)$. The setup algorithm computes and outputs $(\mathsf{pp}, \mathsf{sk}) \leftarrow \mathsf{IPE.Setup}(1^\lambda)$.
- $\mathsf{TIFE.Encrypt_L}(\mathsf{sk}, x)$. For $x \in [N]$, let $\mathbf{e}_x \in \mathbb{Z}_q^{N+1}$ be the basis vector where $(\mathbf{e}_x)_i = 0$ for all $i \neq x$ and $(\mathbf{e}_x)_x = 1$. The left encrypt algorithm outputs $\mathsf{ct_L} \leftarrow \mathsf{IPE.KeyGen}(\mathsf{sk}, \mathbf{e}_x)$.
- $\mathsf{TIFE.Encrypt_R}(\mathsf{sk}, y)$. For $y \in [N]$, let $\mathbf{f}_y \in \mathbb{Z}_q^{N+1}$ be the vector such that for all $x \in [N]$, $\mathbf{f}_{y,x} = f(x, y)$ and $\mathbf{f}_{y,N+1} = 1$. The right encrypt algorithm outputs $\mathsf{ct_R} \leftarrow \mathsf{IPE.Encrypt}(\mathsf{sk}, \mathbf{f}_y)$.
- $\mathsf{TIFE.Decrypt}(\mathsf{pp}, \mathsf{ct_L}, \mathsf{ct_R})$. The decryption algorithm outputs $z \leftarrow \mathsf{IPE.Decrypt}(\mathsf{pp}, \mathsf{ct_L}, \mathsf{ct_R})$.

Correctness. Let $(\mathsf{pp}, \mathsf{sk})$ be the public parameters and secret key output by $\mathsf{TIFE.Setup}(1^\lambda)$, which corresponds to the public parameters and secret key, respectively, of the underlying IPE scheme Π_{ipe}. For any two inputs $x, y \in [N]$, let $\mathsf{ct_L} = \mathsf{TIFE.Encrypt_L}(\mathsf{sk}, x)$ and $\mathsf{ct_R} = \mathsf{TIFE.Encrypt_R}(\mathsf{sk}, y)$. By correctness of Π_{ipe}, the following holds with overwhelming probability:

$$\mathsf{TIFE.Decrypt}(\mathsf{pp}, \mathsf{ct_L}, \mathsf{ct_R}) = \langle \mathbf{e}_x, \mathbf{f}_y \rangle = \mathbf{f}_{y,x} = f(x, y).$$

Security. The security of Π_{tife} follows fairly straightforwardly from the function-hiding properties of Π_{ipe}.

Theorem 4.6. *If Π_{ipe} is fully-secure under an indistinguishability based notion of security, (Definition 2.3), then the two-input functional encryption scheme Π_{tife} is IND-CPA secure (Definition 4.5).*

Proof. Suppose $\mathcal{A}$ is an efficient algorithm that can distinguish between experiments $\mathsf{Expt}_0^{\mathsf{tife}}$ and $\mathsf{Expt}_1^{\mathsf{tife}}$ with some non-negligible probability. We use $\mathcal{A}$ to construct an algorithm $\mathcal{B}$ that distinguishes between $\mathsf{Expt}_0^{\mathsf{ipe\text{-}ind}}$ and $\mathsf{Expt}_1^{\mathsf{ipe\text{-}ind}}$ with the same non-negligible probability.

Fix some $b \in \{0, 1\}$. Algorithm $\mathcal{B}$ proceeds as follows. At the beginning of the game, $\mathcal{B}$ receives the public parameters pp from the challenger in $\mathsf{Expt}_b^{\mathsf{ipe\text{-}ind}}$ and sends pp to $\mathcal{A}$. Whenever $\mathcal{A}$ queries the left encryption or the right encryption oracle, algorithm $\mathcal{B}$ simulates the response by making an appropriate query to its key generation or encryption oracle, as follows:

- **Left encryption oracle.** On input a pair of messages $x_0, x_1 \in [N]$, algorithm $\mathcal{B}$ constructs the vectors $\mathbf{u}_0, \mathbf{u}_1 \in \mathbb{Z}_p^{N+1} \setminus \{\mathbf{0}\}$. For $b \in \{0, 1\}$, $\mathbf{u}_b$ is the canonical basis vector that is 0 everywhere except at position x_b, where it is 1. Algorithm $\mathcal{B}$ submits the pair of vectors $\mathbf{u}_0, \mathbf{u}_1$ to the key generation oracle in $\mathsf{Expt}_b^{\mathsf{ipe\text{-}ind}}$ and obtains a secret key $\mathsf{ct_L}$, which it forwards to $\mathcal{A}$.

- **Right encryption oracle.** On input a pair of messages $y_0, y_1 \in [N]$, algorithm $\mathcal{B}$ constructs two vectors $\mathbf{v}_b \in \mathbb{Z}_p^{N+1} \setminus \{\mathbf{0}\}$, where $\mathbf{v}_{b,i} = f(i, y_b)$ for $b \in \{0,1\}$ and $i \in [N]$ and $\mathbf{v}_{b,N+1} = 1$ for $b \in \{0,1\}$. Algorithm $\mathcal{B}$ submits the pair of vectors $\mathbf{v}_0, \mathbf{v}_1$ to the encryption oracle in $\mathsf{Expt}_b^{\mathsf{ipe\text{-}ind}}$ and obtains a ciphertext ct_R, which it forwards to $\mathcal{A}$.

Eventually, $\mathcal{A}$ outputs a bit b', which $\mathcal{B}$ outputs. This concludes the description of algorithm $\mathcal{B}$.

We claim that $\mathcal{B}$ is an admissible adversary for $\mathsf{Expt}_b^{\mathsf{ipe\text{-}ind}}$. By construction, all the queries algorithm $\mathcal{B}$ makes to the key generation and encryption oracles are non-zero. Next, let Q_1 be the number of key generation oracle queries and Q_2 be the number of encryption oracle queries that algorithm $\mathcal{B}$ makes. By construction of algorithm $\mathcal{B}$, Q_1 and Q_2 are the number of left and encryption queries, respectively, submitted by $\mathcal{A}$. For $i \in [Q_1]$ and $j \in [Q_2]$, let $x_0^{(i)}, x_1^{(i)} \in [N]$ and $y_0^{(j)}, y_1^{(j)} \in [N]$ be the inputs submitted by $\mathcal{A}$ on its i^{th} left encryption query and j^{th} right encryption query, respectively, and let $\mathbf{u}_0^{(i)}, \mathbf{u}_1^{(i)}$ and $\mathbf{v}_0^{(j)}, \mathbf{v}_1^{(j)}$ be the corresponding inputs $\mathcal{B}$ submits on its i^{th} key generation oracle query and j^{th} encryption oracle query, respectively.

By correctness, $f(x_b^{(i)}, y_b^{(j)}) = \langle \mathbf{u}_b^{(i)}, \mathbf{v}_b^{(j)} \rangle$. Since $\mathcal{A}$ is admissible, we have that $f(x_0^{(i)}, y_0^{(j)}) = f(x_1^{(i)}, y_1^{(j)})$. Thus, $\langle \mathbf{u}_0^{(i)}, \mathbf{v}_0^{(j)} \rangle = \langle \mathbf{u}_1^{(i)}, \mathbf{v}_1^{(j)} \rangle$, and $\mathcal{B}$ is admissible. We conclude that $\mathcal{B}$ has perfectly simulated the challenger of $\mathsf{Expt}_b^{\mathsf{tife}}$, and so $\mathsf{Pr}[\mathsf{Expt}_b^{\mathsf{ipe\text{-}ind}}(\mathcal{B}) = 1] = \mathsf{Pr}[\mathsf{Expt}_b^{\mathsf{tife}}(\mathcal{A}) = 1]$. □

5 Implementation and Evaluation

To evaluate the practicality of our main construction, we implemented our function-hiding IPE as well as our two-input functional encryption scheme. Our library is publicly available under a standard open-source license. Our implementation uses the Charm [AGM+13] library to implement the pairing group operations (backed by PBC [Lyn06]), and FLINT [HJP13] for the finite field arithmetic in $\mathbb{Z}_q$. In our benchmarks, we measure the time needed to encrypt, issue keys for, and compute the inner product for N-dimensional binary vectors for several different values of N. We run all of our benchmarks on a Linux desktop with an 8-core Intel Core i7-4790K 4.00GHz processor and 16GB of RAM.

In our implementation, the running time of the setup algorithm is dominated by the inversion of a random $n \times n$ matrix in $\mathbb{Z}_q$, where q is either a 160-bit or 224-bit prime, corresponding to 80 and 112 bits of security, respectively. The inverse computation was done naïvely in $O(n^3)$ time in C. Although this procedure is quite computationally expensive, we note that it only needs to be performed once, and can be done offline on a more powerful machine. As a point of reference, at the 80-bit security level, the setup algorithm completes in about 5 minutes on the desktop for vectors of dimension $N = 100$. Since all of the other IPE operations are agnostic to the actual values in the matrices $\mathbf{B}$ and $\mathbf{B}^\star$, for the benchmarks with higher-dimensional vectors (that is, $N > 100$), we measure the performance with respect to matrices $\mathbf{B}$ and $\mathbf{B}^\star$ that are sampled uniformly at random (rather than setting $\mathbf{B}^\star$ to be a scaled inverse of $\mathbf{B}^\top$ as in the normal setup algorithm). Using simulated rather than real matrices has no effect on the micro-benchmarks of the underlying IPE operations.

Recall from Section 3 that the decryption routine in our IPE scheme requires computing a discrete logarithm. We implemented the baby-step giant-step algorithm [Sha71] for computing discrete logs. In our benchmarks, we measured the runtime of each of the elementary IPE operations as well

	MNT159 ($\lambda = 80$)				MNT224 ($\lambda = 112$)			
N	Keygen	Encrypt	Decrypt	|ct|	Keygen	Encrypt	Decrypt	|ct|
5	0.8 ms	2.6 ms	9.9 ms	791 B	1.4 ms	3.9 ms	20.2 ms	990 B
10	1.2 ms	4.5 ms	24.1 ms	1.4 KB	1.9 ms	7.5 ms	40.1 ms	1.9 KB
30	2.4 ms	12.5 ms	67.1 ms	4.0 KB	4.1 ms	21.1 ms	112.3 ms	5.4 KB
50	4.0 ms	20.7 ms	110.2 ms	6.6 KB	6.6 ms	34.9 ms	184.4 ms	9.0 KB
100	9.8 ms	43.2 ms	217.8 ms	13.0 KB	14.5 ms	71.4 ms	366.4 ms	17.7 KB
250	40.9 ms	124.4 ms	540.9 ms	32.3 KB	52.2 ms	194.6 ms	907.0 ms	44.1 KB
500	140.6 ms	310.5 ms	1.1 s	64.6 KB	163.0 ms	447.7 ms	1.8 s	88.0 KB
750	303.7 ms	555.9 ms	1.6 s	96.8 KB	333.3 ms	753.3 ms	2.7 s	132.0 KB

Table 1: Micro-benchmarks for our inner product encryption scheme over two different pairing curves: MNT159 (for 80 bits of security) and MNT224 (for 112 bits of security). For $N > 100$, we used a simulated setup procedure where the matrices $\mathbf{B}$ and $\mathbf{B}^\star$ that would normally be generated by the setup procedure are instead sampled uniformly at random. For the run-time measurements of the basic IPE operations (Keygen, Encrypt, and Decrypt), we average the performance over 10 runs. We also measure the size |ct| of the IPE ciphertexts.

as the size of the IPE ciphertexts for vectors of varying dimension N. The concrete performance numbers are summarized in Table 1 and Figure 1.

From Table 1, we see that key generation and encryption operations complete in just a few hundred milliseconds, even for high-dimensional vectors. Decryption is slightly slower, requiring on the order of a few seconds for vectors containing 500 components. The difference in run-times is due to the fact that decryption require N pairing operations, while key generation and encryption only require group exponentiation. On the desktop, a single group exponentiation takes about 0.6 ms, while a pairing takes about 2 ms. It is also worth noting that while the only essential difference between key generation and encryption in our IPE scheme is that key generation operates over $\mathbb{G}_1$ while encryption operates over $\mathbb{G}_2$, there is a fairly substantial difference in the run-times of the two operations (generally speaking, at least a factor of 2x). This is an artifact of using an asymmetric pairing group. Group operations in $\mathbb{G}_1$ are faster than those in $\mathbb{G}_2$, so as a result, key generation is much faster than encryption in our IPE scheme.

5.1 Applications

In this section, we revisit some of our example applications from Section 4, and show how our function-hiding inner production encryption can be efficiently applied in those scenarios.

Biometric authentication. In addition to fingerprint scanning, ocular biometrics such as iris scanning and retina scanning are a popular form of biometrics. In recent work, Hämmerle-Uhl et al. [HPPU15] show how a typical iris recognition code consisting of 640 bits is sufficiently descriptive to uniquely identify individuals with considerable accuracy. Thus, in the biometric

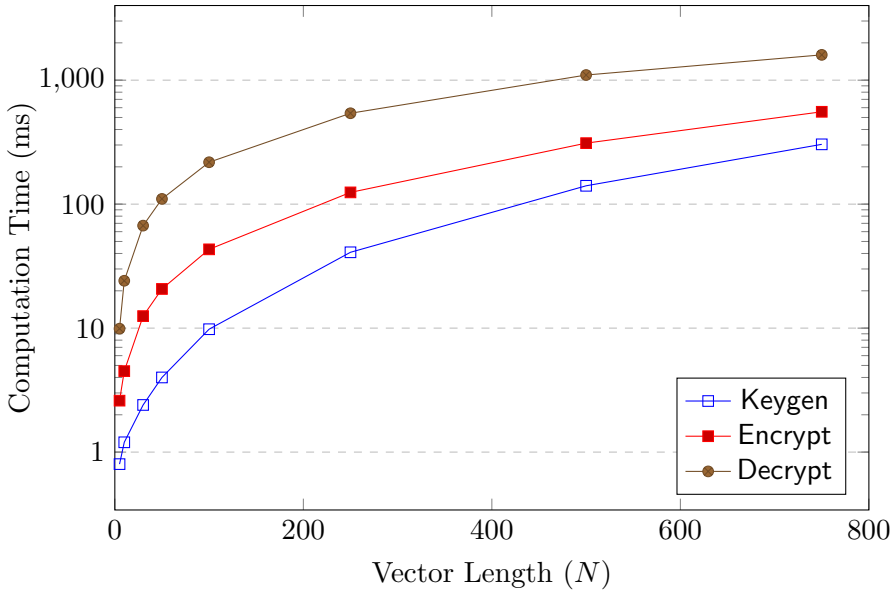

Figure 1: Micro-benchmarks of each of the elementary operations of our function-hiding IPE scheme over the MNT159 curve (provides $\lambda = 80$ bits of security).

authentication example described in Section 4, the authentication server would store an *encryption* of the 640-bit descriptor of each user under a function-hiding inner product encryption scheme. Each iris scanner is given the secret key for the IPE scheme. To authenticate a user, the scanner scans the user's iris, computes the 640-bit descriptor for the resulting scan, extracts a secret key corresponding to the descriptor, and sends the secret key to the server. The authentication server computes the Hamming distance between the user's iris profile and the database's ground truth and accepts if the Hamming distance is sufficiently small.

To leverage this solution, we require a function-hiding inner product encryption scheme that supports vectors of length 640. As our benchmarks show (Table 1), for both 80 and 112 bits of security, the time needed to encrypt the 640-bit description of an iris scan completes within a second, and the time needed to measure the Hamming distance with respect to a ground truth (i.e., decryption) requires at most two seconds. We conclude that using our scheme, biometric authentication based on iris scans requires just 2-3 seconds of computation. This is quite reasonable for environments where iris scans are used to control physical access to restricted areas.

Nearest-neighbor search on encrypted data. To support nearest-neighbor search (based on ℓ_2 distance) on encrypted documents, we first represent documents as a vector of words. One such representation is a binary vector where each component corresponds to a word in the dictionary and the vector has a 1 in each position that corresponds to a word that appears in the document. While the resulting representation can be high-dimensional, applying standard dimensionality-reduction techniques will yield a more compact representation that is more suitable for our methods. For example, Wei and Croft [WC06] describe a topic-modeling based approach that clusters documents based on topic. They show that a set of only a few hundred vectors is sufficient to describe a database containing hundreds of thousands of documents. Thus, using a topic-based embedding of documents, each document can be represented as a vector with dimension under 1000. Nearest-neighbor search

based on ℓ_2 distance can then be performed using our function-hiding inner production encryption scheme in a second or two.

Order-revealing encryption. As noted in Section 1, single-key two-input functional encryption suffices to build property-preserving encryption for binary properties over a small message space. An important special case of two-input property-preserving encryption is order-revealing encryption. Our inner-product encryption scheme gives a direct solution to order-revealing encryption that achieves the best-possible notion of security introduced by Boldyreva et al. [BCLO09].[12] Previous constructions of order-revealing encryption either do not achieve this notion of security [BCLO09, BCO11, CLWW16], or rely on extremely strong cryptographic primitives and thus, are very impractical [BLR$^+$15, GGG$^+$14]. Our inner product encryption scheme provides an alternative solution to this problem in the setting where we have a bounded message space. As a concrete example of a scenario with a bounded message space, suppose an analyst seeks to encrypt users' ages or salary ranges in a database. By construction, the message space in this setting only contains only a small number of possible values (e.g., at most 150). In this case, encryption requires less than a tenth of a second, and decryption requires a quarter of a second. We note, though, that in this bounded-message setting, ORE can also be efficiently constructed directly from one-way functions using the construction described in [LW16]. However, this type of construction does not support two-input functionalities that do not reveal equality relations. In contrast, our IPE construction gives single-key two-input functional encryption for all two-input functionalities over a small domain.

6 Conclusions

In this work, we constructed a fully-secure, secret-key, function-hiding inner product encryption scheme with secret keys and ciphertexts that are considerably shorter than those in all existing constructions of function-hiding IPE. We proved the security of our construction in a generic model of bilinear maps. We also highlighted several applications of function-hiding inner product encryption and showed how this primitive directly gives a construction of single-key, two-input functional encryption for arbitrary functions over polynomial-sized plaintext spaces. Finally, to assess the practicality of our construction, we implemented and benchmarked our inner product encryption scheme on several parameter settings. We conclude with several interesting open problems for further study:

1. Can we construct function-hiding inner product encryption with equally short secret keys and ciphertexts from a concrete assumption in bilinear groups (such as the k-linear assumption)?

2. Can we obtain two-input functional encryption for general functions from function-hiding inner product encryption where the encrypted vectors are *sublinear* in the size of the plaintext space?

[12] This "best-possible" notion of semantic security essentially states that ciphertexts in the ORE scheme reveal no additional information other than the ordering between their underlying plaintext values.

Acknowledgments

This work was supported by NSF, DARPA, the Simons foundation, a grant from ONR, and an NSF Graduate Research Fellowship. Opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA.

References

[AAB+15] Shashank Agrawal, Shweta Agrawal, Saikrishna Badrinarayanan, Abishek Kumarasubramanian, Manoj Prabhakaran, and Amit Sahai. On the practical security of inner product functional encryption. In *PKC*, 2015.

[ABCP15] Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. Simple functional encryption schemes for inner products. In *PKC*, 2015.

[AFK+12] Diego F. Aranha, Laura Fuentes-Castañeda, Edward Knapp, Alfred Menezes, and Francisco Rodríguez-Henríquez. Implementing pairings at the 192-bit security level. In *Pairing-Based Cryptography*, 2012.

[AGM+13] Joseph A. Akinyele, Christina Garman, Ian Miers, Matthew W. Pagano, Michael Rushanan, Matthew Green, and Aviel D. Rubin. Charm: a framework for rapidly prototyping cryptosystems. *Journal of Cryptographic Engineering*, 3(2):111–128, 2013.

[AGVW13] Shweta Agrawal, Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption: New perspectives and lower bounds. In *CRYPTO*, 2013.

[AJ15] Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In *CRYPTO*, 2015.

[AKSX04] Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu. Order-preserving encryption for numeric data. In *ACM SIGMOD*, 2004.

[ALS15] Shweta Agrawal, Benoît Libert, and Damien Stehlé. Fully secure functional encryption for inner products, from standard assumptions. *IACR Cryptology ePrint Archive*, 2015, 2015.

[ARW16] Michel Abdalla, Mariana Raykova, and Hoeteck Wee. Multi-input inner-product functional encryption from pairings. Cryptology ePrint Archive, Report 2016/425, 2016. `http://eprint.iacr.org/`.

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In *EUROCRYPT*, 2005.

[BBO07] Mihir Bellare, Alexandra Boldyreva, and Adam O'Neill. Deterministic and efficiently searchable encryption. In *CRYPTO*, 2007.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *CRYPTO*, 2004.

[BCLO09] Alexandra Boldyreva, Nathan Chenette, Younho Lee, and Adam O'Neill. Order-preserving symmetric encryption. In *EUROCRYPT*, 2009.

[BCO11] Alexandra Boldyreva, Nathan Chenette, and Adam O'Neill. Order-preserving encryption revisited: Improved security analysis and alternative solutions. In *CRYPTO*, 2011.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In *CRYPTO*, 2001.

[BGI+12] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2), 2012.

[BJK15] Allison Bishop, Abhishek Jain, and Lucas Kowalczyk. Function-hiding inner product encryption. In *ASIACRYPT*, 2015.

[BKS15] Zvika Brakerski, Ilan Komargodski, and Gil Segev. From single-input to multi-input functional encryption in the private-key setting. *IACR Cryptology ePrint Archive*, 2015, 2015.

[BLR+15] Dan Boneh, Kevin Lewi, Mariana Raykova, Amit Sahai, Mark Zhandry, and Joe Zimmerman. Semantically secure order-revealing encryption: Multi-input functional encryption without obfuscation. In *EUROCRYPT*, 2015.

[BRS13a] Dan Boneh, Ananth Raghunathan, and Gil Segev. Function-private identity-based encryption: Hiding the function in functional encryption. In *CRYPTO*, 2013.

[BRS13b] Dan Boneh, Ananth Raghunathan, and Gil Segev. Function-private subspace-membership encryption and its applications. In *ASIACRYPT*, 2013.

[BS15] Zvika Brakerski and Gil Segev. Function-private functional encryption in the private-key setting. In *TCC*, 2015.

[BSW07] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *IEEE S&P*, 2007.

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *TCC*, 2011.

[BWZ14] Dan Boneh, David J. Wu, and Joe Zimmerman. Immunizing multilinear maps against zeroizing attacks. *IACR Cryptology ePrint Archive*, 2014, 2014.

[CGH+15] Jean-Sébastien Coron, Craig Gentry, Shai Halevi, Tancrède Lepoint, Hemanta K. Maji, Eric Miles, Mariana Raykova, Amit Sahai, and Mehdi Tibouchi. Zeroizing without low-level zeroes: New MMAP attacks and their limitations. In *CRYPTO*, 2015.

[CGKO06] Reza Curtmola, Juan A. Garay, Seny Kamara, and Rafail Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. In *ACM CCS*, 2006.

[CHL+15] Jung Hee Cheon, Kyoohyung Han, Changmin Lee, Hansol Ryu, and Damien Stehlé. Cryptanalysis of the multilinear map over the integers. In *EUROCRYPT*, 2015.

[CJL16] Jung Hee Cheon, Jinhyuck Jeong, and Changmin Lee. An algorithm for NTRU problems and cryptanalysis of the GGH multilinear map without an encoding of zero. *IACR Cryptology ePrint Archive*, 2016, 2016.

[CLR15] Jung Hee Cheon, Changmin Lee, and Hansol Ryu. Cryptanalysis of the new CLT multilinear maps. *IACR Cryptology ePrint Archive*, 2015, 2015.

[CLWW16] Nathan Chenette, Kevin Lewi, Stephen A. Weis, and David J. Wu. Practical order-revealing encryption with limited leakage. In *FSE*, 2016.

[Coc01] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In *Cryptography and Coding*, 2001.

[Cor15] Jean-Sébastien Coron. Cryptanalysis of GGH15 multilinear maps, 2015.

[Cos12] Craig Costello. Particularly friendly members of family trees. *IACR Cryptology ePrint Archive*, 2012, 2012.

[DDM16] Pratish Datta, Ratna Dutta, and Sourav Mukhopadhyay. Functional encryption for inner product with full function privacy. In *PKC*, 2016.

[FST10] David Freeman, Michael Scott, and Edlyn Teske. A taxonomy of pairing-friendly elliptic curves. *J. Cryptology*, 23(2), 2010.

[GGG+14] Shafi Goldwasser, S. Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. In *EUROCRYPT*, 2014.

[GGH+13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, 2013.

[GGHZ16] Sanjam Garg, Craig Gentry, Shai Halevi, and Mark Zhandry. Functional encryption without obfuscation. In *TCC*, 2016.

[GKP+13] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In *STOC*, 2013.

[Goh03] Eu-Jin Goh. Secure indexes. *IACR Cryptology ePrint Archive*, 2003, 2003.

[GVW12] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In *CRYPTO*, 2012.

[GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In *STOC*, 2013.

[GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits from LWE. In *CRYPTO*, 2015.

[HJ15] Yupu Hu and Huiwen Jia. Cryptanalysis of GGH map. *IACR Cryptology ePrint Archive*, 2015, 2015.

[HJP13] W. Hart, F. Johansson, and S. Pancratz. FLINT: Fast Library for Number Theory, 2013. Version 2.4.0, `http://flintlib.org`.

[HPPU15] Jutta Hämmerle-Uhl, Georg Penn, Gerhard Pötzelsberger, and Andreas Uhl. Size-reduction strategies for iris codes. *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, 9(1), 2015.

[Jou00] Antoine Joux. A one round protocol for tripartite diffie-hellman. In *ANTS*, 2000.

[JP16] Marc Joye and Alain Passelègue. Practical trade-offs for multi-input functional encryption. *IACR Cryptology ePrint Archive*, 2016, 2016.

[KB16] Taechan Kim and Razvan Barbulescu. Extended tower number field sieve: A new complexity for the medium prime case. In *CRYPTO*, pages 543–571, 2016.

[KKS17] Sungwook Kim, Jinsu Kim, and Jae Hong Seo. A new approach for practical function-private inner product encryption. *IACR Cryptology ePrint Archive*, 2017:4, 2017.

[KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *EUROCRYPT*, 2008.

[LL18] Kwangsu Lee and Dong Hoon Lee. Two-input functional encryption for inner products from bilinear maps. *IEICE Transactions*, 101-A(6), 2018.

[LV16] Huijia Lin and Vinod Vaikuntanathan. Indistinguishability obfuscation from ddh-like assumptions on constant-degree graded encodings. In *FOCS*, pages 11–20, 2016.

[LW16] Kevin Lewi and David J. Wu. Order-revealing encryption: New constructions, applications, and lower bounds. In *ACM CCS*, 2016. To appear.

[Lyn06] Ben Lynn. The pairing-based cryptography library. *Internet: crypto. stanford. edu/pbc/[Mar. 27, 2013]*, 2006.

[MF15] Brice Minaud and Pierre-Alain Fouque. Cryptanalysis of the new multilinear map over the integers. *IACR Cryptology ePrint Archive*, 2015, 2015.

[Mil04] Victor S. Miller. The weil pairing, and its efficient calculation. *J. Cryptology*, 17(4), 2004.

[MSZ16] Eric Miles, Amit Sahai, and Mark Zhandry. Annihilation attacks for multilinear maps: Cryptanalysis of indistinguishability obfuscation over GGH13. *IACR Cryptology ePrint Archive*, 2016, 2016.

[Nec94] V.I. Nechaev. Complexity of a determinate algorithm for the discrete logarithm. *MATHEMATICAL NOTES*, 55, 1994.

[Nie02] Jesper Buus Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In *CRYPTO*, 2002.

[O'N10] Adam O'Neill. Definitional issues in functional encryption. *IACR Cryptology ePrint Archive*, 2010, 2010.

[OT08] Tatsuaki Okamoto and Katsuyuki Takashima. Homomorphic encryption and signatures from vector decomposition. In *Pairing*, 2008.

[OT09] Tatsuaki Okamoto and Katsuyuki Takashima. Hierarchical predicate encryption for inner-products. In *ASIACRYPT*, 2009.

[OT10] Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In *CRYPTO*, 2010.

[OT12] Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure unbounded inner-product and attribute-based encryption. In *ASIACRYPT*, 2012.

[PR12] Omkant Pandey and Yannis Rouselakis. Property preserving symmetric encryption. In *EUROCRYPT*, 2012.

[Ram16] Somindu C. Ramanna. More efficient constructions for inner-product encryption. *IACR Cryptology ePrint Archive*, 2016, 2016.

[Sch80] Jacob T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4), 1980.

[Sha71] Daniel Shanks. Class number, a theory of factorization, and genera. In *Proc. Sympos. Pure Math.*, 1971.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In *EUROCRYPT*, 1997.

[SS10] Amit Sahai and Hakan Seyalioglu. Worry-free encryption: functional encryption with public keys. In *ACM CCS*, 2010.

[SSW09] Emily Shen, Elaine Shi, and Brent Waters. Predicate privacy in encryption systems. In *TCC*, 2009.

[SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, 2005.

[SWP00] Dawn Xiaodong Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In *IEEE S&P*, 2000.

[TAO16] Junichi Tomida, Masayuki Abe, and Tatsuaki Okamoto. Efficient functional encryption for inner-product values with full-hiding security. In *ISC*, 2016.

[Wat15] Brent Waters. A punctured programming approach to adaptively secure functional encryption. In *CRYPTO*, 2015.

[WC06] Xing Wei and W. Bruce Croft. Lda-based document models for ad-hoc retrieval. In *SIGIR*, 2006.

[Zim15] Joe Zimmerman. How to obfuscate programs directly. In *EUROCRYPT*, 2015.

[Zip79] Richard Zippel. Probabilistic algorithms for sparse polynomials. In *EUROSAM*, 1979.