Projective Arithmetic Functional Encryption and Indistinguishability Obfuscation from Degree-5 Multilinear Maps

Prabhanjan Ananth(✉) and Amit Sahai

Center for Encrypted Functionalities and Department of Computer Science,
UCLA, Los Angeles, USA
{prabhanjan,sahai}@cs.ucla.edu

Abstract. In this work, we propose a variant of functional encryption called *projective arithmetic functional encryption* (PAFE). Roughly speaking, our notion is like functional encryption for arithmetic circuits, but where secret keys only yield partially decrypted values. These partially decrypted values can be linearly combined with known coefficients and the result can be tested to see if it is a small value.

We give a *degree-preserving* construction of PAFE from multilinear maps. That is, we show how to achieve PAFE for arithmetic circuits of degree d using only degree-d multilinear maps. Our construction is based on an assumption over such multilinear maps, that we justify in a generic model. We then turn to applying our notion of PAFE to one of the most pressing open problems in the foundations of cryptography: building secure indistinguishability obfuscation (i$\mathcal{O}$) from simpler building blocks.

i$\mathcal{O}$ from degree-5 multilinear maps. Recently, the works of Lin [Eurocrypt 2016] and Lin-Vaikuntanathan [FOCS 2016] showed how to build i$\mathcal{O}$ from constant-degree multilinear maps. However, no explicit constant

The full version of this paper is available in [AS16].

P. Ananth—Work done in part while visiting the Simons Institute for Theoretical Computer Science, supported by the Simons Foundation and by the DIMACS/Simons Collaboration in Cryptography through NSF grant #CNS-1523467. This work was partially supported by grant #360584 from the Simons Foundation and the grants listed under Amit Sahai.

A. Sahai—Work done in part while visiting the Simons Institute for Theoretical Computer Science, supported by the Simons Foundation and by the DIMACS/Simons Collaboration in Cryptography through NSF grant #CNS-1523467. Research supported in part from a DARPA/ARL SAFEWARE award, NSF Frontier Award 1413955, NSF grants 1619348, 1228984, 1136174, and 1065276, BSF grant 2012378, a Xerox Faculty Research Award, a Google Faculty Research Award, an equipment grant from Intel, and an Okawa Foundation Research Grant. This material is based upon work supported by the Defense Advanced Research Projects Agency through the ARL under Contract W911NF-15-C-0205. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense, the National Science Foundation, or the U.S. Government.

J.-S. Coron and J.B. Nielsen (Eds.): EUROCRYPT 2017, Part I, LNCS 10210, pp. 152–181, 2017.
DOI: 10.1007/978-3-319-56620-7_6

was given in these works, and an analysis of these published works shows that the degree requirement would be in excess of 30. The ultimate "dream" goal of this line of work would be to reduce the degree requirement all the way to 2, allowing for the use of well-studied bilinear maps, or barring that, to a low constant that may be supportable by alternative secure low-degree multilinear map candidates. We make substantial progress toward this goal by showing how to leverage PAFE for degree-5 arithmetic circuits to achieve $\mathsf{i}\mathcal{O}$, thus yielding the first $\mathsf{i}\mathcal{O}$ construction from degree-5 multilinear maps.

1 Introduction

Functional encryption (FE), introduced by Sahai and Waters [SW05,SW08], allows for the creation of secret keys sk_f corresponding to functions f, such that when such a secret key sk_f is applied to an encryption of x, decryption yields $f(x)$ but, intuitively speaking, nothing more is revealed about x. In this work, we will focus on the secret-key variant of FE where knowledge of the master secret key is needed to perform encryption. Functional encryption has proven to be remarkably versatile: it captures as special cases efficient applications like attribute-based encryption for formulas [GPSW06,BSW07] and predicate encryption for inner products [KSW08] from bilinear maps. At the same time, the general notion of functional encryption implies remarkably powerful primitives, including most notably indistinguishability obfuscation ($\mathsf{i}\mathcal{O}$) [AJ15,BV15,AJS15,BNPW16].

In this work, we continue the study of functional encryption notions, constructions, and implications. As a byproduct of our study, we tackle the one of the most pressing open problems in theoretical cryptography: building secure $\mathsf{i}\mathcal{O}$ from simpler building blocks. In particular, we give the first construction of $\mathsf{i}\mathcal{O}$ using only degree-5 multilinear maps.

FE in the Arithmetic Context. For a number of cryptographic objects that deal with general computations, *arithmetic* circuits have been considered in addition to boolean circuits. The primary motivation for this arises when we wish to apply these objects to cryptographic computations, since many cryptographic computations can be better expressed as arithmetic circuits rather than boolean circuits. For example, zero-knowledge proofs [GMR89] for arithmetic circuits (e.g. [GS08] in the bilinear setting) have been influential because they allow for the construction of zero-knowledge protocols whose structure and complexity more closely match the structure and complexity of algebraic cryptographic algorithms.

In a similar spirit, we study general FE in the context where secret keys should correspond to arithmetic circuits. Notably however, our motivation will not (primarily) be efficiency, but rather achieving new feasibility results, as we will elaborate below.

Previous work has studied FE for arithmetic circuits in two special cases: The work of Boneh et al. [BNS13,BGG+14] studied attribute-based encryption for arithmetic circuits from the LWE assumption. (Our work will diverge technically from this.) Another line of work started with the work of Katz, Sahai, and Waters [KSW08], studying FE where secret keys corresponded to arithmetic inner product computations, using bilinear groups as the underlying cryptographic tool. There has been several followup papers on FE for inner products [ABCP15, AAB+15, BJK15, ABCP16, DDM16, LV16] with various security notions and correctness properties. An issue that will be important to us, and that arises already in the context of inner products, concerns the *correctness* property of the FE scheme. Ideally, a secret key for an arithmetic circuit C, when applied to an encryption of x, should allow the decryptor to learn $C(x)$. However, FE constructions typically store values "in the exponent," and thus the difficulty of discrete logarithms in bilinear groups implies that if $C(x)$ is superpolynomial, it will be difficult to recover. This issue has been dealt with in the past either by requiring that decryption only reveals whether $C(x) = 0$, as in [KSW08], or by requiring that decryption only reveals $C(x)$ if $C(x)$ is polynomially bounded, such as in the works of Abdalla et al. and others [ABCP15, BJK15, ABCP16, DDM16]. We will diverge from past work when dealing with this issue, in order to provide greater flexibility, and in so doing, we introduce our notion of *projective*[1] *arithmetic FE.*

1.1 Our Contributions

Projective Arithmetic FE (PAFE). In projective arithmetic FE, like in FE, encrypting a value x yields a ciphertext c. Also like in (arithmetic) FE, in PAFE each secret key sk_C is associated with an *arithmetic* circuit[2] C. However, unlike in FE, in PAFE when the secret key sk_C is applied to the ciphertext c, it does not directly yield the decrypted value $C(x)$, but rather this yields a partial decryption p_C. We call this process *projective decryption.* We envision a party holding a collection of secret keys $\{sk_C\}_C$ would apply projective decryption using these secret keys to the ciphertext c to obtain a collection of partial decryptions $\{p_C\}_C$. Finally, this party can choose any collection of small coefficients $\{\alpha_C\}_C$ arbitrarily, and then call a different efficient *recovery* algorithm which is given all the partial decryptions $\{p_C\}_C$ and coefficients $\{\alpha_C\}_C$. The recovery algorithm then outputs a bit that indicates whether $\sum_C \alpha_C C(x) = 0$

[1] We call our notion *projective* FE because, roughly speaking, a user holding a collection of keys $\{sk_C\}_C$ for several arithmetic circuits C can only learn information about various *linear projections* $\sum_C \alpha_C C(x)$ for known small coefficients $\{\alpha_C\}_C$. We discuss this in more detail below. Our name is also loosely inspired by the notion of projective hash functions, introduced by Cramer and Shoup [CS02], where keys (called projective keys) only allow one to evaluate the hash function on inputs x in some NP language, but not on all strings. In our setting, as well, our keys are similarly only "partially functional" in that they only allow the user to learn information about various linear projections, and they do not in general reveal the full information that should be learned by obtaining all $C(x)$ values. However, to the best of our knowledge, only this loose relationship exists between projective hash functions and our notion of projective FE.

[2] We only are interested in arithmetic circuits of fan-in 2.

or not. (More generally, we can allow the user to recover the value of $\sum_C \alpha_C C(x)$ as long as it is bounded by a polynomial.)

Thus, projective arithmetic FE can be seen as relaxing the correctness guarantee that would be provided by the standard notion of FE when applied to arithmetic circuits over fields of superpolynomial size (which is not known to be achievable). Of course, if decryption actually allowed a user to learn $\{C(x)\}_C$ for several arithmetic circuits C, then the user would be able to compute $\sum_C \alpha_C C(x)$ for any set of small coefficients $\{\alpha_C\}_C$ of her choice. Note that our notion is more permissive than *only* revealing whether $C(x) = 0$, as in the original work for FE for inner products [KSW08], or only revealing $C(x)$ if it is polynomially bounded, such as in other works on FE for inner products [ABCP15,BJK15,ABCP16,DDM16]. With regard to security, our notion will, intuitively speaking, only require indistinguishability of encryptions of x from encryptions of y, if $C(x) = C(y)$ for all secret keys sk_C obtained by the adversary. However, for our application of PAFE to iO, we require a stronger notion of security that we call *semi-functional security*. We give an intuitive explanation of this notion in the technical overview.

Degree-Preserving Construction of PAFE from Multilinear Maps. The first main technical contribution of our work is a construction of (secret-key) PAFE for degree-d arithmetic circuits, from degree-d *asymmetric* multilinear maps[3]. Furthermore, it suffices that the groups over which the multilinear maps are defined are prime order. Our construction is based on an explicit pair of assumptions over such multilinear maps, that we can justify in the standard generic multilinear model.

Theorem 1 (Informal). *There exists secret-key PAFE for degree-d arithmetic circuits from degree-d prime order asymmetric multilinear maps under Assumptions #1 and #2 (see Sect. 4.1).*

Our assumptions do not require any low-level encodings of 0 to be given to the adversary, and we thus believe them to be instantiable using existing candidate multilinear maps. Indeed, because of some pseudorandomness properties of our construction and generic proof of security, we believe that our assumptions can be proven secure in the Weak MMap model considered in the works of Miles et al. and Garg et al. [MSZ16,GMM+16], which would give further evidence of its instantiability. Because we want to posit instantiable assumptions, we do not formulate a succinct version of our assumption together with a reduction of security as was done in the works of Gentry et al. or Lin and Vaikuntanathan [GLSW15,LV16], because unfortunately no existing candidate multilinear map construction is known to securely support such reductions, and indeed the assumptions of [GLSW15,LV16] are broken when instantiated with existing candidates. We stress that, like in the recent work of [Lin16,LV16], if

[3] Roughly speaking, asymmetric multilinear maps disallows pairing of elements from the same group structure.

the degree d is constant, then our pair of assumptions would only involve a constant-degree multilinear map.

Our construction can be seen as a generalizing FE for inner products (degree 2 functions) from bilinear maps, to higher degrees in a degree preserving manner. Thus, our construction can be applied to cryptographic computations that are naturally represented as arithmetic functions of low degree, but not as inner products. In more detail, we introduce the notion of slotted encodings that has the same flavor of multilinear maps defined over composite order groups. We then show how to emulate slotted encodings using prime-order multilinear maps. However, this emulation strategy only works in the case of constant degree. We hope that this technique will be useful to transform constructions based on constant degree composite order multilinear maps (for example [Lin16]) to constructions based on constant degree prime order multilinear maps.

$\mathsf{i}\mathcal{O}$ from Degree-5 Multilinear Maps. Our motivation for building PAFE for arithmetic circuits in a degree-preserving manner is to achieve new feasibility results for $\mathsf{i}\mathcal{O}$ from low-degree multilinear maps. The concept of $\mathsf{i}\mathcal{O}$ was first defined by Barak et al. [BGI+01]. Informally speaking, $\mathsf{i}\mathcal{O}$ converts a program (represented by a boolean circuit) into a "pseudo-canonical form." That is, for any two equivalent programs P_0, P_1 of the same size, we require that $\mathsf{i}\mathcal{O}(P_0)$ is computationally indistinguishable from $\mathsf{i}\mathcal{O}(P_1)$. The first candidate construction of $\mathsf{i}\mathcal{O}$ was given by Garg et al. [GGH+13b], and especially since the introduction of punctured programming techniques of Sahai and Waters [SW14], $\mathsf{i}\mathcal{O}$ has found numerous applications, with numerous papers published since 2013 that use $\mathsf{i}\mathcal{O}$ to accomplish cryptographic tasks that were not known to be feasible before (see, e.g., [GGH+13b,SW14,GGHR14,HSW14,GGG+14,BPR15,BP15, CHN+16,BGJ+16]). However, it is still not known how to build $\mathsf{i}\mathcal{O}$ from standard cryptographic assumptions. Given the enormous applicability of $\mathsf{i}\mathcal{O}$ to a wide variety of cryptographic problems, one of the most pressing open problems in the foundations of cryptography is to find ways to construct $\mathsf{i}\mathcal{O}$ from simpler building blocks. Indeed, while there have been dozens of papers published showing how to use $\mathsf{i}\mathcal{O}$ to accomplish amazing things, only a handful of papers have explored simpler building blocks that suffice for constructing $\mathsf{i}\mathcal{O}$.

One line of work toward this objective is by Lin [Lin16] and Lin and Vaikuntanathan [LV16], who showed how to build $\mathsf{i}\mathcal{O}$ from constant-degree multilinear maps. Unfortunately, no explicit constant was given in these works, and an analysis of these published works shows that the degree requirement would be in excess of 100. The ultimate "dream" goal of this line of work would be to reduce the degree requirement all the way to 2, allowing for the use of well-studied bilinear maps, or barring that, to a low constant that may be supportable by alternative secure low-degree multilinear map candidates.

We make substantial progress toward this goal by showing how to achieve $\mathsf{i}\mathcal{O}$ starting from PAFE. Specifically, we first construct ε-sublinear secret key functional encryption for NC^1 circuits, with constant $\varepsilon < 1$, starting from PAFE[4] for

[4] We additionally require that PAFE has encryption complexity to be multiplicative overhead in the message size. Our construction of PAFE satisfies this property.

degree-d arithmetic circuits and a specific type of degree d-randomizing polynomials [IK00, AIK06][5]. We require that the randomizing polynomials satisfy some additional properties such as the encoding polynomials should be homogenous, the randomness complexity[6] is ε-sub-linear in the circuit size and the decoding algorithm should be executed as a sequence of linear functions. We call a scheme that satisfies these additional properties as homogenous randomizing polynomials with ε-sub-linear randomness complexity. As we will see later, we can achieve ε-sub-linear randomness complexity property for free by employing an appropriate pseudorandom generator of $\frac{1}{\varepsilon'}$-stretch, where constant $\varepsilon' > 1$ is related to ε. Hence, we only care about constructing homogenous randomizing polynomials (without sublinear property) and we provide an information theoretic construction achieving the same.

Once we construct ε-sublinear secret key functional encryption, we can then invoke the result of [BNPW16] and additionally assume learning with errors to obtain iO. For this transformation, we are required to assume that the underlying FE scheme and learning with errors is sub-exponentially secure. Thus,

Theorem 2 (Informal). *We construct an indistinguishability obfuscation scheme for P/poly assuming the following: for some constant d,*

1. *Sub-exponentially secure PAFE scheme for degree d arithmetic circuits with multiplicative overhead in encryption complexity. From Theorem 1, this can be based on sub-exponentially secure Assumptions #1 and #2 (Sect. 4.1).*
2. *Sub-exponentially secure degree d homogenous randomizing polynomials with ε-sub-linear randomness complexity. This can be based on sub-exponentially secure pseudorandom generators of stretch $\frac{1}{\varepsilon'}$, where constant $\varepsilon' > 1$ is related to ε.*
3. *Sub-exponentially secure learning with errors.*

Instantiation: We show how to leverage PAFE for degree-5 arithmetic circuits to achieve i$\mathcal{O}$, thus yielding the first i$\mathcal{O}$ construction from degree-5 multilinear maps. The crucial step in this transformation is to first construct homogenous randomizing polynomials with sub-linear randomness complexity of degree 15. We first identify that the work of [AIK06] satisfies the required properties of a degree-3 homogenous randomizing polynomials scheme. To achieve sublinear randomness complexity, we assume an explicit degree-2 pseudo-random generator (PRGs) achieving super-linear stretch in the boolean setting, and a related explicit degree-3 PRG achieving super-quadratic stretch in the arithmetic setting. In particular we use a boolean PRG of stretch 1.49 and an algebraic PRG of stretch 2.49 [OW14] (see also [AL16]). We then observe that for a special class of circuits $\mathcal{C}$, the degree of the above polynomials can be reduced to 5 if we additionally allow for pre-processing of randomness. Also, we show how to

[5] The degree of a randomizing polynomial is defined to be the maximum degree of the polynomials computing the encoding function.

[6] Randomness complexity in this context refers to the size of the random string used in the encoding algorithm.

remove the algebraic PRG part in the construction of randomizing polynomials for $\mathcal{C}$.

As alluded to above, the fact that our PAFE can directly deal with an arithmetic PRG in a degree-preserving manner is critical to allowing us to achieve $\mathsf{i}\mathcal{O}$ with just degree-5 mutlilinear maps.

Theorem 3 (Informal). *We construct an indistinguishability obfuscation scheme for P/poly assuming the following: for some constant d,*

1. *Sub-exponentially secure PAFE scheme for degree 5 arithmetic circuits with multiplicative overhead in encryption complexity. From Theorem 1, this can be based on sub-exponentially secure Assumptions #1 and #2 (Sect. 4.1).*
2. *Sub-exponentially secure degree 5 homogenous randomizing polynomials for $\mathcal{C}$ with ε-sub-linear randomness complexity. This can be based on sub-exponentially secure boolean PRG of stretch 1.01.*
3. *Sub-exponentially secure learning with errors.*

Concurrent Work(s). In a concurrent work, Lin obtains a new IO construction with a security reduction to (1) L-linear maps with the subexponential symmetric external Diffie-Hellman (SXDH) assumption, (2) subexponentially secure locality-L PRG, and (3) subexponential LWE. When using a locality 5 PRG, 5-linear maps with the SXDH assumption suffice. The L-linear maps consist of L source groups $G_1, \cdots, G_L$, whose elements $g_1^{a_1}, \cdots, g_L^{a_L}$ can be "paired" together to yield an element in a target group $g_T^{a_1 \cdots a_L}$. The SXDH assumption on such multilinear maps is a natural generalization of the SXDH assumption on bilinear maps: It postulates that the DDH assumption holds in every source group G_d, that is, elements g_d^a, g_d^b, g_d^{ab} are indistinguishable from g_d^a, g_d^b, g_d^r, for random a, b and r.

To obtain IO, she first constructs collusion-resistant FE schemes for computing degree-L polynomials from L-linear maps, and then bootstraps such FE schemes to IO for P, assuming subexponentially secure locality-L PRG and LWE.

A corollary of our degree-preserving PAFE construction is a construction of FE for degree-2 polynomials from bilinear maps. Concurrently, two works [BCF16, Gay16] achieved the same result based on concrete assumptions on bilinear maps.

We now give a technical overview of our approach.

1.2 Technical Overview

We give an informal description of the algorithms of projective arithmetic functional encryption (PAFE). We focus on secret-key setting in this work.

- **Setup**: It outputs secret key MSK.
- **Key Generation**: On input an arithmetic circuit C and master secret key, it produces a functional key sk_C.
- **Encryption**: On input message x, it outputs a ciphertext CT.

- **Projective Decryption**: On input a functional key sk_C and ciphertext CT, it produces a partial decrypted value ι.
- **Recover**: On input many partial decrypted values $\{\iota_i\}$ and a linear function (specified as co-efficients), it outputs the result of applying the linear function on the values contained in $\{\iota_i\}$.

We first show how to achieve iO starting from secret-key PAFE. Later, we show how to obtain PAFE for degree $\mathbf{D}$ polynomials starting from degree $\mathbf{D}$ multilinear maps.

iO from Secret-Key PAFE: We start with the goal of constructing a sub-linear secret-key FE scheme for NC^1 (from which we can obtain iO [BNPW16]) starting from PAFE for constant degree arithmetic circuits. Our goal is to minimize the degree of arithmetic circuits that suffices us to achieve sub-linear FE.

We start with the standard tool of randomizing polynomials to implement NC^1 using a constant degree arithmetic circuit. We use randomizing polynomials with a special decoder: the decoder is a sequence of linear functions chosen adaptively[7]. At a high level the construction proceeds as follows: let the randomizing polynomial of circuit C, input x and randomness r be of the form $p_1(x;r), \ldots, p_N(x;r)$. The sub-linear FE functional key corresponding to a circuit C are a collection of PAFE keys for $p_1, \ldots, p_N$. The encryption of x w.r.t sublinear FE scheme is a PAFE encryption of (x, r). To obtain $C(x)$, first execute the projective decryption algorithm on key of p_i and ciphertext of (x, r) to obtain partial decrypted values corresponding to $p_i(x, r)$. Now, execute the recover algorithm on input a linear function and the above partial decrypted values, where the linear function is chosen by the decoder of the randomizing polynomials scheme. Depending on the output of the recover algorithm, the decoder picks a new linear function. This process is repeated until we finally recover the output of the circuit C.

Before we justify why this scheme is secure, we remark as to why this scheme satisfies the sub-linear efficiency property. In order to achieve sub-linear efficiency, we require that $|r| = |C|^{1-\varepsilon}$ for some $\varepsilon > 0$. Thus, we require randomizing polynomials with sub-linear randomness complexity. We remark later how to achieve this.

The next goal is to argue security: prior works either employ function privacy properties [BS15] or Trojan techniques [CIJ+13, ABSV15] to make the above approach work. However, going through these routes is going to increase the degree of arithmetic circuits required to achieve sub-linear FE. Instead, we start with a PAFE scheme with a stronger security guarantee called *semi-functional* security. This notion is inspired by the dual system methodology introduced by Waters [Wat09] in different context and later employed by several other works

[7] That is, choice of every linear function could depend on the output of the previously chosen linear functions on the encoding of computation.

(see for example, [LOS+10, GGHZ14]). Associated with this notion, there are two types of objects:

- *Semi-Functional Keys:* A semi-functional key is associated with an arithmetic circuit C and a hardwired value v.
- *Semi-Functional Ciphertexts:* A semi-functional ciphertext is generated just using the master secret key.

We define how honestly generated keys, honestly generated ciphertexts and semi-functional keys, semi-functional ciphertexts are required to behave with each other in Table 1. Honestly generated key or ciphertext refers to generation of key or ciphertext according to the description of the scheme.

Table 1. We consider four possibilities of decryption: (a) honestly generated keys correctly decrypts honestly generated ciphertexts (from correctness property), (b) semi-functional keys also correctly decrypts honestly generated ciphertexts, (c) there is no correctness guarantee on the decryption of honestly generated keys on semi-functional ciphertexts, (d) Finally, the decryption of semi-functional keys on semi-functional ciphertexts yields the hardwired value associated with the key.

	Honestly generated keys	Semi-functional keys
Honestly generated ciphertexts	Honest decryption	Honest decryption
Semi-functional ciphertexts	Not defined	Output hardwired value

A PAFE scheme is said to satisfy semi-functional security if both the following definitions are satisfied:

- *Indistinguishability of Semi-functional keys*: It should be hard to distinguish an honestly generated functional key of C from a semi-functional key of C associated with any hardwired value v.
- *Indistinguishability of Semi-functional Ciphertexts*: It should be hard to distinguish an honestly generated ciphertext of x from a semi-functional ciphertext if every functional key of C issued is a semi-functional key associated with hardwired value $C(x)$.

Once we have a secret key PAFE scheme that satisfies semi-functional security then we can prove the security as follows: we consider a simple case when the adversary only submits one message query (x_0, x_1).

- We first turn the functional key associated with an arithmetic circuit C into a semi-functional key with the hardwired value $C(x_0)$.
- Once all the functional keys are semi-functional, we can now switch the ciphertext of x_0 to semi-functional ciphertext.

- Since $C(x_0) = C(x_1)$, we can switch back the semi-functional keys to be honestly generated functional keys.
- Finally, we switch back the ciphertext from semi-functional to honestly generated ciphertext of x_1.

If the adversary requests multiple message queries, then the above process is to be repeated one message query at a time.

Choice of Randomizing Polynomials with Sub-linear Randomness: The next question is what randomizing polynomials do we choose to instantiate the above approach. As we will see later, if we choose randomizing polynomials with sub-linear randomness complexity of degree $\mathbf{D}$ then it suffices build PAFE from degree $\mathbf{D}$ multilinear maps. Also, we will require the polynomials to be homogenous.

Hence, our goal is to choose a homogenous randomizing polynomials with minimal degree and also satisfying (i) linear decodability and (ii) sub-linear randomness complexity properties. We achieve this in the following steps:

1. First, build randomizing polynomials with minimal degree. We start with [AIK06] for NC^1, where the polynomials are of degree 3. In spirit, this is essentially information theoretic Yao with the wire keys being elements over $\mathbb{F}_\mathbf{p}$ and every wire key is associated with a random mask (which is represented as a bit) that helps in figuring out which of the four entries to be decoded for the next gate.
2. The above scheme already satisfies linear decodability property. This is because the decryption of every garbled gate is a linear operation. The linear function chosen to decrypt one garbled gate now depends on the linear functions chosen to decrypt its children gates.
3. Next, we tackle sub-linear randomness complexity: we generate the wire keys and the random masks as the output of a PRG. The total length of all the wire keys is roughly square the size of the NC^1 circuit. This is because, the size of the wire keys at the bottom most (input) layer are proportional to the size of the circuit. We use an algebraic PRG of stretch $(2 + \varepsilon)$ to generate the wire keys and we use a boolean PRG to generate the random masks. The degree of the algebraic PRG over $\mathbb{F}_\mathbf{p}$ is 3 while the degree of the boolean PRG represented over $\mathbb{F}_\mathbf{p}$ is 5. When the above PRGs are plugged into the randomizing polynomials construction from the above step, we get the degree of the polynomials to be 15.
4. Finally, we show how to make the above randomizing polynomials homogenous. This is done using a standard homogenization argument: add dummy variables to the polynomials such that the degree of all the terms in the polynomials are the same. While evaluating these polynomials, set all these dummy variables to 1. This retains the functionality and at the same time ensures homogeneity.

We can now use the above randomizing polynomials scheme to instantiate the above approach. After partial decryption, we get partial decrypted values associated with $\{p_i(x;r)\}$. Now, since the decoding is composed of many linear functions, we can execute the Recover algorithm (multiple times) to recover the output.

Reducing the Degree: We can apply some preprocessing to reduce the degree of the above polynomials further. We remark how to reduce the degree to 5. Later, in the technical sections, we explore alternate ways of reducing the degree, as well.

Suppose we intend to construct sublinear FE for a specific class of circuits $\mathcal{C}$. In this case, we are required to construct randomizing polynomials only for $\mathcal{C} \in NC^1$.

We define $\mathcal{C}$ as follows: every circuit $C \in \mathcal{C}$ of output length $\mathbf{N}$ is of the form $C = (C_1, \ldots, C_{\mathbf{N}})$, where (i) C_i outputs the i^{th} output bit of C, (ii) $|C_i| = \text{poly}(\lambda)$ for a fixed polynomial poly, (iii) Depth of C_i is $c \cdot \log(\lambda)$, where c is a constant independent of $|C|$ and, (iv) C_i for every $i \in [\mathbf{N}]$ has the same topology–what is different, however, are the constants associated with the wires. We show later that it suffices to build sublinear FE for $\mathcal{C}$ to obtain iO. We now focus on obtain randomizing polynomials for $\mathcal{C}$.

We start with the randomizing polynomials scheme that we described above. Recall that it involved generating a garbled table for every gate in the circuit C. Moreover, the randomness to generate this garbled table is derived from an algebraic and a boolean PRG. We make the following useful changes: let $C = (C_1, \ldots, C_{\mathbf{N}})$ such that C_i outputs the i^{th} output bit of C. Let $w_1^i, \ldots, w_{\mathsf{nw}}^i$ be the set of wires in C_i and $G_1^i, \ldots, G_{\mathsf{ng}}^i$ be the set of gates in C_i.

- We invoke nw number of instantiations of boolean PRGs $\mathsf{bPRG}_1^w, \ldots, \mathsf{bPRG}_{\mathsf{nw}}^w$ and $\mathsf{bPRG}_1^r, \ldots, \mathsf{bPRG}_{\mathsf{nw}}^r$. All these PRGs have the same structure (i.e., same predicates is used) and have degree 5 over arbitrary field (with slightly superlinear stretch $1{+}\varepsilon$). Pseudorandom generator bPRG_j^w is used to generate wire keys for wires $w_j^1, \ldots, w_j^{\mathbf{N}}$. Recall that earlier we were using an algebraic PRG of quadratic stretch. This is because the size of wire keys was proportional to exponential in depth, which could potentially be linear in the size of the circuit. However, since we are considering the specific circuit class $\mathcal{C}$, the depth of every circuit is $c \log(\lambda)$. And thus the size of the wire keys is independent of the security parameter. This is turn allows us to use just a PRG of superlinear stretch $1{+}\varepsilon$. Finally, bPRG_j^r is used to generate random masks for the wires $w_j^1, \ldots, w_j^{\mathbf{N}}$.
- We now consider the [AIK06] randomizing polynomials associated with circuit C. As before, we substitute the variables associated with wire keys and random masks with the polynomials associated with the appropriate PRGs. The formal variables in the PRG polynomials are associated with the seed.
- The result of the above process is the encoding of C consisting of polynomials $p_1, \ldots, p_N$ with variables associated with the seeds of PRGs. Note that the degree of these polynomials is still 15.

- We then observe that there are polynomials $q_1, \ldots, q_T$ in seed variables such that $p_1, \ldots, p_N$ can be rewritten in terms of $q_1, \ldots, q_T$ and moreover, the degree of p_i in the new variables $\{q_i\}$ is 5. The advantage of doing this is that the polynomials $\{q_i\}$ can be evaluated during the encryption phase[8]. The only thing we need to be wary of is the fact that T could be as big as $|C|$. If this is the case then the encryption complexity would be at least linear in $|C|$, which violate the sublinearity of the FE scheme. We show how to carefully pick $q_1, \ldots, q_T$ such that T is sub-linear in $|C|$ and the above properties hold. We refer the reader to the technical sections for more details.

The only missing piece here is to show that sublinear FE for this special class of circuits $\mathcal{C}$ with sub-exponential security loss implies iO. To show this, it suffices to show that sublinear FE for $\mathcal{C}$ implies sublinear FE for all circuits. Consider the transformation from FE for NC^1 to FE for all circuits by [ABSV15] – the same transformation also works for single-key sublinear secret key FE. We consider a variant of their transformation. In this transformation, a sublinear FE key for circuit C' is generated by constructing a circuit C that has hardwired into it C' and value v. Circuit C takes as input x, PRF key K and mode b. If $b = 0$ it outputs a Yao's garbled circuit of (C, x) computed w.r.t randomness derived from K. If $b = 1$ it outputs the value v. We can re-write C as being composed of sub-circuits $C_1, \ldots, C_{\mathbf{N}}$ such that each of C_i is in NC^1, $|C_i| = \text{poly}(\lambda)$ and depth of C_i is $c \cdot \log(\lambda)$ for a fixed polynomial poly and fixed constant c. Intuitively, C_i, has hardwired into it gate G_i of C' and i^{th} block of v. It computes a garbled table corresponding to G_i if $b = 0$, otherwise it outputs the i^{th} block of v.

Constructing PAFE: We now focus on building PAFE from multilinear maps. The first attempt to encrypt the input $x = (x_1, \ldots, x_{\ell_{\mathsf{inp}}})$ would be to just encode every x_i separately. Now, during evaluation of circuits $C_1, \ldots, C_N$ on these encodings will yield a top level encoding of $C_i(x)$. This homomorphic evaluation would correspond to projective decryption operation. The recover algorithm would just compute a linear function on all the top level encodings of $C_i(x)$ and using zero test parameters, recover the answer if the output of the linear function is 0.

However, we cannot allow the adversary to evaluate recover outputs for circuits C_i of his choice. We should ensure that he recovers outputs only for circuits corresponding to which he has been issued functional keys. The main challenge in designing a functional key for C is to guarantee authenticity – how do we ensure that if the adversary, given a functional key corresponding to C, can only evaluate C on these inputs? To ensure this, we introduce a parallel branch of computation: we instead encode (x_i, α_i) where $\{\alpha_i\}$ are random elements determined during the setup. Then as part of the functional key associated with C, we give out an encoding of $C(\{\alpha_i\})$ at the top level that will allow us to cancel the α_i part after computing C on encodings of $\{(x_i, \alpha_i)\}$ and in the end, just get an encoding of $C(x)$. However, to implement this, we need to make sure that

[8] This idea is similar in spirit to the recent work of Bitansky et al. [BLP16], who introduced degree reduction techniques in a different context.

the computation of C on $\{x_i\}$ and $\{\alpha_i\}$ are done separately even though x_i and α_i are encoded together.

The work of [Zim15,AB15] used the above idea in the context of designing iO. As we will discuss below, we extend their techniques in several ways, to deal with the problem of mixing ciphertext components and achieving the semi-functional security properties we need from our PAFE scheme. However, before we discuss these difficulties, we note that the work of [Zim15,AB15] implement parallel branches by using composite order multilinear maps. Composite order multilinear maps allow for jointly encoding for a vector of elements such that addition and multiplication operations can be homomorphically performed on every component of the vector separately.

However, one of the primary motivations for this line of work on building constructions for $\mathsf{i}\mathcal{O}$ from low-degree multilinear maps is to enable the use of future candidate low-degree multilinear maps, where achieving composite order may not be possible. Indeed, current instantiations of composite order multlinear maps [CLT13] have poorly understood security properties, and have been subject to efficient cryptanalytic attacks in some settings (see, e.g., [CHL+15,CGH+15]). Thus, instead of relying on composite order multilinear maps, we do the following: we introduce a primitive called a slotted encoding scheme, that allows for the same functionality as offered by composite order multilinear maps. This then helps us in implementing the idea of [Zim15,AB15] using a slotted encoding scheme. We later show how to realize a constant degree slotted encoding scheme using prime order multilinear maps. We define slotted encodings next.

Slotted Encoding: A slotted encoding scheme, parameterized by L (number of slots), has the following algorithms: (i) Setup: this generates the secret parameters, (ii) Encode: it takes as input $(a_1,\ldots,a_L)$ and outputs an encoding of it, (iii) Arithmetic operations: it takes two encodings of $(a_1,\ldots,a_L)$ and $(b_1,\ldots,b_L)$ and performs arithmetic operations on every component separately. For instance, addition of encoding of $(a_1,\ldots,a_L)$ and $(b_1,\ldots,b_L)$ would lead to encoding of $(a_1+b_1,\ldots,a_L+b_L)$, (iv) Zero Testing: It outputs success if the encoding of $(a_1,\ldots,a_L)$ is such that $a_i=0$ for every i.

In this work, we will be interested in asymmetric slotted encodings, where the slotted encodings is associated with a tree T such that every encoding is associated with a node in T and two encodings can be paired only if their associated nodes are siblings. The degree of slotted encodings is defined to be the maximum degree of polynomials the scheme lets us evaluate.

Constant Degree Slotted Encoding from Prime Order MMaps: We start with the simple case when degree of slotted encodings is 2 (the bilinear case). The idea of dual vector spaces were introduced by [OT08] and further developed as relevant to us by [OT09,BJK15] to address this problem for bilinear maps. In this framework, there is an algorithm that generates $2n$ vectors $(\mu_1,\ldots,\mu_n),(\nu_1,\ldots,\nu_n)$ of dimension n such that: (i) inner product, $\langle\mu_i,\nu_i\rangle=1$ and, (ii) inner product, $\langle\mu_i,\nu_j\rangle=0$ when $i\neq j$. Using this, we can encode

$(a_1, \ldots, a_n)$ associated with some node u in the tree as follows: encode every element of the vector $a_1\mu_1 + \cdots + a_n\mu_n$. The encoding of $(b_1, \ldots, b_n)$ associated with a node v, which is a sibling of u, will be encodings of the vector $b_1\nu_1 + \cdots + b_n\nu_n$. Now, computing inner product of both these encodings will lead to an encoding of $a_1 \cdot b_1 + \cdots + a_n \cdot b_n$.

This idea doesn't suffice for degree 3. So our idea is to work modularly, and consider multiple layers of vectors. The encoding of $(a_1, \ldots, a_n)$ under node u will be encodings of the vector $(a_1\mu_1 \otimes \mu'_1 + \cdots + a_n\mu_n \otimes \mu'_n)$[9], where $\{\mu'_i\}$ is a basis of a vector space associated with the parent of node u. Now, when this is combined with encoding of $b_1\nu_1 + \cdots + b_n\nu_n$, computed under node v, we get encoding of $(a_1 b_1 \mu'_1 + \cdots a_n b_n \mu'_n)$. Using this we can then continue for one more level.

To generalize this for higher degrees we require tensoring of multiple vectors (potentially as many as the depth of the tree). This means that the size of the encodings at the lower levels is exponential in the depth and thus, we can only handle constant depth trees. Implementing our tensoring idea for multiple levels is fairly technical, and we refer the reader to the relevant technical section for more details.

PAFE from Slotted Encodings: Using slotted encodings, we make a next attempt in constructing PAFE:

- To encrypt $x = (x_1, \ldots, x_{\ell_{\mathsf{inp}}})$, we compute a slotted encoding of (x_i, α_i), where α_i are sampled uniformly at random during the setup phase.
- A functional key of C consists of a slotted encoding of $(0, C(\{\alpha_i\}))$ at the top level.

The partial decryption first homomorphically evaluates C on slotted encodings of (x_i, α_i) to get a slotted encoding of $(C(\{x_i\}), C(\{\alpha_i\}))$. The second slot can be 'canceled' using top level encoding of $(0, C(\{\alpha_i\}))$ to get an encoding of $(C(\{x_i\}), 0)$. The hope is that if the evaluator uses a different circuit C' then the second slot will not get canceled and hence, he would be unable to get a zero encoding.

However, choosing a different C' is not the only thing an adversary can do. He could also mix encodings from different ciphertexts and try to compute C on it – the above approach does not prevent such attacks. In order to handle this, we need to ensure that the evaluation of ciphertexts can never be mixed. In order to solve this problem, we use a mask γ that be independently sampled for every ciphertext. Every encoding will now be associated with this mask. Implementing this idea will crucially make use of the fact that the polynomial computed by the arithmetic circuit is a homogenous polynomial.

Yet another problem arises is in the security proof: for example, to design semi-functional keys, we need to hardwire a value in the functional key. In order to enable this, we introduce a third slot. With this new modification, we put

[9] Here, $\mu_i \otimes \mu_j$ denotes the tensoring of μ_i and μ_j.

forward a template of our construction. Our actual construction involves more details which we skip to keep this section informal.

- To encrypt $x = (x_1, \ldots, x_{\ell_{\mathsf{inp}}})$, we compute a slotted encoding of $(x_i, \alpha_i, 0)$, where α_i are sampled uniformly at random during the setup phase. Additionally, you give out encoding of $(0, S, 0)$ at one level lower than the top level, where S is also picked at random in the setup phase.
- A functional key of C consists of a slotted encoding of $(0, C(\{\alpha_i\}) \cdot S^{-1}, 0)$ at the top level.

The decryption proceeds as before, except that the encodings of $(0, C(\{\alpha_i\}) \cdot S^{-1}, 0)$ and $(0, S, 0)$ are paired together before we proceed.

Note that in both the ciphertext and the functional key, the third slot is not used at all. The third slot helps in the security proof. To see how we describe the semi-functional parameters at a high level as follows:

- *Semi-functional Ciphertexts*: To encrypt $x = (x_1, \ldots, x_{\ell_{\mathsf{inp}}})$, we compute a slotted encoding of $(0, \alpha_i, 0)$, where α_i is computed as before. Additionally, you give out encoding of $(0, S, 1)$ at one level lower than the top level, where S is also picked at random in the setup phase. Note that the third slot now contains 1 which signals that it is activated.
- *Semi-functional Keys*: A functional key of C consists of a slotted encoding of $(0, C(\{\alpha_i\}), v)$ at the one level lower than top level, where v is the hardwired value associated with the semi-functional key.

During the decryption of semi-functional key with honestly generated ciphertext, the third slot will not be used since it will be deactivated in the ciphertext. So the decryption proceeds normally. However, during the decryption of semi-functional key with semi-functional ciphertexts, the third slot is used since the third slot is activated in the ciphertext. We argue the security of our construction in the ideal multilinear map model.

Comparison with [LV16]. We now compare our work with the recent exciting work of [LV16], in order to illustrate some differences that allow us to achieve lower degree. The work of [LV16] first defines FE for NC^0 with a non-trivial efficiency property and give a new bootstrapping theorem[10] to achieve compact FE. They then show how to achieve FE for NC^0 from constant degree multilinear maps[11]. Interestingly, they use arithmetic randomizing polynomials within their construction of FE for NC^0 – this will be important as we note below.

[10] Their bootstrapping theorem also works if we start with FE for constant degree polynomials over $\mathbb{F}_2$.

[11] Note that, in particular, the security of their scheme reduces to a succinct assumption called the multilinear joint SXDH assumption. As we noted earlier, unfortunately this assumption is not known to be instantiable with existing multilinear map candidates. However, one can posit a different assumption that directly assumes their FE for NC^0 scheme to be secure, and we do not know of any attacks on that (non-succinct) assumption.

In contrast, we do not build FE for NC^0, but rather show how to proceed directly from projective arithmetic FE for degree-5 arithmetic circuits to iO (without additional use of multilinear maps). Furthermore, our construction of PAFE is *degree preserving*, so to achieve PAFE for degree-5 arithmetic circuits, we only need degree-5 multilinear maps. In contrast, in [LV16], to build FE for NC^0, their work has to "pay" in degree not only based on the depth of the NC^0 circuit that underlies each secret key, but also for the arithmetic randomizing polynomial that they apply to the NC^0 circuit. This adds a significant overhead in the constant degree their multilinear map must support. Our approach is simpler, as our randomizing polynomials are only used in the path from PAFE to iO, which does not use multilinear maps in any additional way. There are, of course, many other technical differences between our work and [LV16], as well. Another conceptual idea that we introduce, and that is different from [LV16], is the notion of slotted encodings, an abstraction of composite order multilinear maps, and our method for emulating slotted encodings using prime order multilinear maps without increasing the degree.

Organization. We define the notion of projective arithmetic functional encryption and present a degree-preserving construction of PAFE from slotted encodings. In the full version, we show how to combine PAFE and (a stronger notion of) randomizing polynomials to obtain secret key functional encryption that can then bootstrapped to obtain iO.

2 Projective Arithmetic Functional Encryption

Throughout this paper we will use standard cryptographic notation and concepts; for details, refer to the full version. In this section, we introduce the notion of projective arithmetic functional encryption scheme. There are two main differences from a (standard) functional encryption scheme:

- Functional keys are associated with arithmetic circuits.
- The projective decryption algorithm only outputs partial decrypted values. There is a recover algorithm that computes on the partial decrypted values and produces an output.

2.1 Definition

We can consider either a public key projective arithmetic FE scheme or a secret key projective arithmetic secret key FE scheme. In this work, we define and construct a secret key projective arithmetic FE scheme.

A secret-key projective arithmetic functional encryption (FE) scheme PAFE over field $\mathbb{F}_{\mathbf{p}}$ is associated with a message space $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda \in \mathbb{N}}$ and a arithmetic circuit class $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ over $\mathbb{F}_{\mathbf{p}}$. Here, $\mathcal{X}$ comprises of strings with every symbol in the string belongs to $\mathbb{F}_{\mathbf{p}}$.

PAFE comprises of a tuple $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{ProjectDec})$ of PPT algorithms with the following properties:

- $\mathsf{Setup}(1^\lambda)$: The setup algorithm takes as input the unary representation of the security parameter, and outputs a secret key MSK.
- $\mathsf{KeyGen}(\mathsf{MSK}, C)$: The key-generation algorithm takes as input the secret key MSK and a arithmetic circuit $C \in \mathcal{C}_\lambda$, over $\mathbb{F}_{\mathbf{p}}$, and outputs a functional key sk_C.
- $\mathsf{Enc}(\mathsf{MSK}, x)$: The encryption algorithm takes as input the secret key MSK and a message $x \in \mathcal{X}_\lambda$, and outputs a ciphertext CT.
- $\mathsf{ProjectDec}(sk_C, \mathsf{CT})$: The projective decryption algorithm takes as input a functional key sk_C and a ciphertext CT, and outputs a partial decrypted value ι.
- $\mathsf{Recover}(c_1, \iota_1, \ldots, c_{\ell_{\mathfrak{f}}}, \iota_{\ell_{\mathfrak{f}}})$: The recover algorithm takes as input co-efficients $c_1, \ldots, c_{\ell_{\mathfrak{f}}} \in \mathbb{F}_{\mathbf{p}}$, partial decrypted values $\iota_1, \ldots, \iota_{\ell_{\mathfrak{f}}}$ and outputs out.

We first define the correctness property and later, define the security property.

B-Correctness. The correctness is parameterized by a set $B \subseteq \mathbb{F}_{\mathbf{p}}$. We emphasize that B is a set of polynomial size, i.e., $|B| = \text{poly}(\lambda)$. Consider an honestly generated ciphertext CT of input x. Consider honestly generated keys $sk_{C_1}, \ldots, sk_{C_{\ell_{\mathfrak{f}}}}$. Denote the corresponding decrypted values to be $\iota_1, \ldots, \iota_{\ell_{\mathfrak{f}}}$. If it holds that $\sum_{i=1}^{\ell_{\mathfrak{f}}} c_i \cdot C_i(x) = \mathsf{out}^* \in B$ then we require that $\mathsf{Recover}(c_1, \iota_1, \ldots, c_{\ell_{\mathfrak{f}}}, \iota_{\ell_{\mathfrak{f}}})$, where $c_i \in \mathbb{F}_{\mathbf{p}}$, always outputs out^*.

Remark 1. Our construction only supports the case when $B = \{0\}$ when implemented by multilinear maps that only allows for zero testing at the final level. However, if encodings of 1 are given out at the top level, then B can be defined to be the set $\{0, \ldots, \text{poly}(\lambda)\}$, where poly is a fixed polynomial.

Remark 2 ((B,B′)-Correctness). We can also consider a property that we call (B, B')-correctness. It is the same as B-correctness except that the co-efficients c_i input to the above evaluation algorithm has to be in the set $B' \subseteq \mathbb{F}_{\mathbf{p}}$.

Remark 3 (Alternate Notation of Evaluation). Instead of feeding coefficients to the evaluation algorithm, we can directly feed in the description of the linear function. That is, if $\mathsf{out}^* \leftarrow \mathsf{Recover}(\mathfrak{f}, (\iota_1, \ldots, \iota_{\ell_{\ell_{\mathfrak{f}}}}))$ with $\mathfrak{f}$ being a linear function then we require that $\mathfrak{f}(C_1(x), \ldots, C_{\ell_{\ell_{\mathfrak{f}}}}) = \mathsf{out}^*$, where ι_i is obtained by decrypting a functional key of C_i with x.

2.2 Semi-Functional Security

We introduce a notion of semi-functional security associated with projective arithmetic FE. We refer the reader to the technical overview for an informal intuition behind the notion of semi-functional security.

We define the following two auxiliary algorithms.

Semi-Functional Key Generation, $\mathsf{sfKG}(\mathsf{MSK}, C, \theta)$: On input master secret key MSK, arithmetic circuit C, value θ, it outputs a semi-functional key sk_C.

Semi-Functional Encryption, $\mathsf{sfEnc}(\mathsf{MSK}, 1^{\ell_{\mathsf{inp}}})$: On input master secret key MSK and ℓ_{inp}, it outputs a semi-functional ciphertext CT.

We now introduce two security properties. We start with the first property, namely indistinguishability of semi-functional keys.

This property states that it should be hard for an efficient adversary to distinguish a semi-functional key associated with circuit C and value v from an honestly generated key associated with C. Additionally, the adversary can request for other semi-functional keys or honestly generated keys. The ciphertexts will be honestly generated.

Definition 1 (Indistinguishability of Semi-Functional Keys). *Consider a projective arithmetic functional encryption scheme* $\mathsf{PAFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{ProjectDec}, \mathsf{Recover})$*. We say that* PAFE *satisfies* **indistinguishability of semi-functional keys** *with respect to* sfKG *if for any PPT adversary* $\mathcal{A}$ *there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that*

$$\mathsf{Advtge}_{\mathcal{A}}^{\mathsf{PAFE}}(\lambda) = \Big| \mathsf{Pr}[\mathsf{Expt}_{\mathcal{A}}^{\mathsf{PAFE}}(\lambda, 0) = 1] - \mathsf{Pr}[\mathsf{Expt}_{\mathcal{A}}^{\mathsf{PAFE}}(\lambda, 1) = 1] \Big| \leq \mathsf{negl}(\lambda),$$

for all sufficiently large $\lambda \in \mathbb{N}$*, where for each* $b \in \{0,1\}$ *and* $\lambda \in \mathbb{N}$ *the experiment* $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{PAFE}}(1^\lambda, b)$*, modeled as a game between the adversary* $\mathcal{A}$ *and a challenger, is defined as follows:*

1. **Setup phase:** *The challenger samples* $\mathsf{MSK} \leftarrow \mathsf{Setup}(1^\lambda)$.
2. **Message queries:** *On input* 1^λ *the adversary submits* $(x_1, \ldots, x_{\ell_{\mathbf{x}}})$ *for some polynomial* $\ell_{\mathbf{x}} = \ell_{\mathbf{x}}(\lambda)$.
3. **Function queries:** *The adversary also submits arithmetic circuit queries to the challenger. There are three tuples the adversary submits:*
   - *This comprises of circuits and values associated with every circuit;* $(C_1^0, \theta_1, \ldots, C_{\ell_{\mathbf{f}}}^0, \theta_{\ell_{\mathbf{f}}})$*. Here,* $\theta_j \in \mathbb{F}_{\mathbf{p}}$.
   - *This comprises of just circuits;* $(C_1^1, \ldots, C_{\ell'_{\mathbf{f}}}^1)$.
   - *This corresponds to a challenge circuit pair query* (C^*, θ^*)
4. **Challenger's response:** *The challenger replies with* $(\mathsf{CT}_1, \ldots, \mathsf{CT}_{\ell_{\mathbf{x}}})$*, where* $\mathsf{CT}_i \leftarrow \mathsf{Enc}(\mathsf{MSK}, x_i)$ *for every* $i \in [\ell_{\mathbf{x}}]$*. It also sends the following functional keys: for every* $j \in [\ell_{\mathbf{f}}]$,
   - $sk_{C_j^0} \leftarrow \mathsf{sfKG}(\mathsf{MSK}, C_j^0, \theta_j)$.
   - $sk_{C_j^1} \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, C_j^1)$.
   - *If* $b = 0$*, generate* $sk_{C^*} \leftarrow \mathsf{sfKG}(\mathsf{MSK}, C^*, \theta^*)$*. Otherwise generate* $sk_{C^*} \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, C^*)$.
5. **Output phase:** *The adversary outputs a bit* b' *which is defined as the output of the experiment.*

The second property is indistinguishability of semi-functional ciphertexts. This property states that it should be hard for an efficient adversary to distinguish honestly generated ciphertext of x from a semi-functional ciphertext. In this experiment, it is required that the adversary only gets semi-functional keys associated with circuits C_i and value v_i such that $v_i = C_i(x)$.

Definition 2 (Indistinguishability of Semi-Functional Ciphertexts). *Consider a projective arithmetic functional encryption scheme* $\mathsf{PAFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{ProjectDec}, \mathsf{Recover})$. *We say that* PAFE *satisfies* **indistinguishability of semi-functional ciphertexts** *with respect to* sfEnc *if for any PPT adversary $\mathcal{A}$ there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that*

$$\mathsf{Advtge}_{\mathcal{A}}^{\mathsf{PAFE}}(\lambda) = \left| \Pr[\mathsf{Expt}_{\mathcal{A}}^{\mathsf{PAFE}}(\lambda, 0) = 1] - \Pr[\mathsf{Expt}_{\mathcal{A}}^{\mathsf{PAFE}}(\lambda, 1) = 1] \right| \leq \mathsf{negl}(\lambda),$$

for all sufficiently large $\lambda \in \mathbb{N}$, where for each $b \in \{0,1\}$ and $\lambda \in \mathbb{N}$ the experiment $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{PAFE}}(1^\lambda, b)$, modeled as a game between the adversary $\mathcal{A}$ and a challenger, is defined as follows:

1. **Setup phase:** *The challenger samples* $\mathsf{MSK} \leftarrow \mathsf{Setup}(1^\lambda)$.
2. **Message queries:** *On input 1^λ the adversary submits $(x_1, \ldots, x_{\ell_{\mathbf{x}}})$ for some polynomial $\ell_{\mathbf{x}} = \ell_{\mathbf{x}}(\lambda)$ and it also sends the challenge query x^*.*
3. **Function queries:** *The adversary also submits arithmetic circuit queries to the challenger. The query is of the form $(C_1, \theta_1, \ldots, C_{\ell_{\mathbf{f}}}, \theta_{\ell_{\mathbf{f}}})$. It should hold that $\theta_j = C_j(x^*)$ for every $j \in [\ell_{\mathbf{f}}]$. If it does not hold, the experiment is aborted.*
4. **Challenger's response:** *The challenger replies with $(\mathsf{CT}_1, \ldots, \mathsf{CT}_{\ell_{\mathbf{x}}})$, where $\mathsf{CT}_i \leftarrow \mathsf{Enc}(\mathsf{MSK}, x_i)$ for every $i \in [\ell_{\mathbf{x}}]$. It sends $\mathsf{CT}^* \leftarrow \mathsf{Enc}(\mathsf{MSK}, x^*)$ only if $b = 0$, otherwise it sends $\mathsf{CT}^* \leftarrow \mathsf{sfEnc}\left(\mathsf{MSK}, 1^{|x^*|}\right)$. Finally, it sends the following functional keys: for every $j \in [\ell_{\mathbf{f}}]$, compute $sk_{C_j} \leftarrow \mathsf{sfKG}(\mathsf{MSK}, C_j, \theta_j)$.*
5. **Output phase:** *The adversary outputs a bit b' which is defined as the output of the experiment.*

Remark 4. One can also define a stronger property where instead of submitting one challenge message x^*, the challenger submits a challenge message pair (x_0^*, x_1^*) and the requirement that for every circuit C_j query, $C_j(x_0^*) = C_j(x_1^*)$. The reduction, in response, encrypts x_b^* where b is the challenge bit. It can be seen that this stronger security property is implied by the above property.

We now define semi-functional security property.

Definition 3. *We say that a projective arithmetic FE scheme, over $\mathbb{F}_{\mathbf{p}}$, is said to be* **semi-functionally secure** *if it satisfies both (i) indistinguishability of semi-functional keys property and, (ii) indistinguishability of semi-functional ciphertexts property.*

2.3 Other Notions

We also consider the following two notions of projective arithmetic FE.

Constant Degree Projective Arithmetic FE. In this work, we are interested in projective arithmetic FE for circuits that compute constant degree arithmetic circuits. In particular, we consider constant degree arithmetic circuits over arbitrary field $\mathbb{F}_{\mathbf{p}}$.

Multiplicative Overhead in Encryption Complexity. We say that a projective arithmetic FE scheme, over field $\mathbb{F}_{\mathbf{p}}$, satisfies multiplicative overhead in encryption complexity property if the complexity of encrypting x is $|x| \cdot \text{poly}(\lambda, \log(\mathbf{p}))$. That is,

Definition 4 (Multiplicative Overhead in Encryption Complexity). *Consider a projective arithmetic FE scheme* $\mathsf{PAFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{ProjectDec})$*, over field* $\mathbb{F}_{\mathbf{p}}$*. We say that* PAFE *satisfies multiplicative overhead in encryption complexity if* $|\mathsf{Enc}(\mathsf{MSK}, x)| = |x| \cdot \text{poly}(\lambda, \log(\mathbf{p}))$*, where* MSK *is the secret key generated during setup.*

Circuits versus Polynomials. Often in this manuscript, we interchangeably use arithmetic circuits over $\mathbb{F}_{\mathbf{p}}$ with polynomials computed over $\mathbb{F}_{\mathbf{p}}$. If there is a polynomial p over $\mathbb{F}_{\mathbf{p}}$ having $\text{poly}(\lambda)$ number of terms then there is a $\text{poly}'(\lambda)$-sized arithmetic circuit over $\mathbb{F}_{\mathbf{p}}$, where poly and poly$'$ are polynomials. However, the reverse in general need not be true: if there is a $\text{poly}'(\lambda)$-sized arithmetic circuit over $\mathbb{F}_{\mathbf{p}}$ then the associated polynomial could have exponentially many terms. For example: $(x_1 + x_2) \cdots (x_{2n-1} + x_{2n})$ has a succinct circuit representation but when expanded as a polynomial has exponential number of terms.

In this work, we are only interested in arithmetic circuits which can be expressed as polynomials efficiently. In particular, we consider arithmetic circuits of constant fan-in and constant depth.

3 Slotted Encodings

We define the notion of slotted encodings: this concept can be thought of as abstraction of composite order multilinear maps. It allows for jointly encoding a vector of elements. Given the encodings of two vectors, using the addition and multiplication operations it is possible to either homomorphically add the vectors component-wise or multiply them component-wise.

To define this primitive, we first define the notion of structured asymmetric multilinear maps in Sect. 3.1. We show in Sect. 3.2 how to instantiate this form of structured asymmetric multilinear maps using current known instantiations of multilinear maps. Once we have armed ourselves with the definition of structured multilinear maps, we define the notion of slotted encodings (a special type of structured multilinear maps) in Sect. 3.3. In the full version, we show how to realize slotted encodings using structured asymmetric multilinear maps for the constant degree[12] case.

3.1 Structured (Asymmetric) Multilinear Maps

We define the notion of structured asymmetric multilinear maps. It is associated with a binary tree T. Every node is associated with a group structure and

[12] As we see later, this corresponds to the scenario where the structured multilinear maps is associated with constant number of bilinear maps.

additionally, every non leaf node is associated with a *noisy* bilinear map. Every element in this group structure has multiple noisy representations as in the case of recent multilinear map candidates [GGH13a, CLT13, GGH15].

Suppose nodes u and v are children of node w in tree T. And let the respective associated groups be $\mathbf{G}_u, \mathbf{G}_v$ and $\mathbf{G}_w$ respectively. Let e_{uv} be the bilinear map associated with node w. Then $e_{uv} : \mathbf{G}_u \times \mathbf{G}_v \rightarrow \mathbf{G}_w$.

Before we define structured multilinear maps we first put forward some notation about trees and also define some structural properties that will be useful later.

Notation About Trees: Consider a tree $T = (V, E)$, where V denotes the set of vertices and E denotes the set of edges. We are only interested in binary trees (every node has only two children) in this work.

1. We define the function $\mathbf{lc} : [V] \rightarrow \{0, 1\}$ such that $\mathbf{lc}(u) = 0$ if u is the left child of its parent, else $\mathbf{lc}(u) = 1$ if u is the right child of its parent.
2. We define $\mathbf{par} : [V] \rightarrow [V]$ such that $\mathbf{par}(u) = v$ if v is the parent of u.
3. $\mathbf{rt}(T) = w$ if the root of T is w.

Definition of Structured Multilinear Maps. A structured multilinear maps is defined by the tuple $\mathsf{SMMap} = (T = (V, E),\ \{\mathbf{G}_u\}_{u \in V})$ and associated with ring R, where:

- $T = (V, E)$ is a tree.
- $\mathbf{G}_u$ is a group structure associated with node $u \in V$. The order of the group is N.

The encoding of elements and operations performed on them are specified by the following algorithms:

- **Secret Key Generation,** $\mathsf{Gen}(1^\lambda)$: It outputs secret key $\mathbf{sk}$ and zero test parameters ztpp.
- **Encoding,** $\mathsf{Encode}(\mathbf{sk}, a, u \in V)$: In addition to secret key $\mathbf{sk}$, it takes as input $a \in R$ and a node $u \in V$. It outputs an encoding $[a]_{\mathbf{u}}$.
- **Add,** $[a]_{\mathbf{u}} + [b]_{\mathbf{u}} = [a + b]_{\mathbf{u}}$. Note that only elements corresponding to the same node in the tree can be added.
- **Multiply,** $[a]_{\mathbf{u}} \circ [b]_{\mathbf{v}} = [a \cdot b]_{\mathbf{w}}$. Here, w is the parent of u and v, i.e., $w = \mathbf{par}(u)$ and $w = \mathbf{par}(v)$.
- **Zero Test,** $\mathsf{ZeroTest}(\mathsf{ztpp}, [a]_{\mathfrak{r}})$: On input zero test parameters ztpp and an encoding $[a]_{\mathfrak{r}}$ at level $\mathfrak{r}$, where $\mathfrak{r} = \mathbf{rt}(T)$, output 0 if and only if $a = 0$.

We define degree of structured multilinear maps.

Definition 5 (Degree of SMMAP). *Consider a structured multilinear maps scheme given by* $\mathsf{SMMap} = (T = (V, E),\ \{\mathbf{G}_u\}_{u \in V})$. *The* **degree** *of* SMMap *is defined recursively as follows.*
We assign degree to every node in the tree as follows:

- *Degree of every leaf node u is 1.*

- *Consider a non leaf node w. Let u and v be its children. The degree of w is the sum of degree of u and degree of v.*

The degree of SMMap *is defined to be the degree of the root node.*

Remark 5. If we restrict ourselves to only binary trees (which is the case in our work) and if d is the depth of the binary tree T then the degree of SMMap, associated with $(T, \{\mathbf{G}_u\}_{u \in V})$ is 2^d.

Useful Notation: We employ the following notation that will be helpful later. Suppose $[v_1]_{\mathbf{i}}, \ldots, [v_m]_{\mathbf{i}}$ be a vector of encodings and let $\mathbf{v} = (v_1, \ldots, v_m) \in \mathbb{Z}_N^m$. Then, $[\mathbf{v}]_{\mathbf{i}}^m$ denotes $([v_1]_{\mathbf{i}}, \ldots, [v_m]_{\mathbf{i}})$. If the dimension of the vector is clear, we just drop m from the subscript and write $[\mathbf{v}]_{\mathbf{i}}$.

3.2 Instantiations of Structured Multilinear Maps

We can instantiate structured multilinear maps using the 'asymmetric' version of existing multilinear map candidates [GGH13a, CLT13]. For example, in asymmetric GGH, every encoding is associated with set S. Two encodings associated with the same set can be added. If there are two encodings associated with sets S_1 and S_2 respectively, then they can be paired if and only if $S_1 \cap S_2 = \emptyset$. The encoding at the final level is associated with the universe set, that is the union of all the sets.

To construct a structure multilinear map associated with $(T = (V, E), \phi)$, we can start with a universal set $U = \{1, \ldots, |V'|\}$, where $V' \subseteq V$ is the set of leaves in T. That is, there are as many elements as the number of leaves in V. We then design a bijection $\psi : U \rightarrow [V']$. An encoding is encoded at a leaf node u under the set $S_u = \{\psi^{-1}(u)\}$. For a non leaf node w, the encoding is performed under the set $S_w = S_u \cup S_v$, where u and v are the children of w.

3.3 Definition

A L-slotted encoding SEnc is a type of structured multilinear maps $\mathsf{SMMap} = (T = (V, E),\ \{\mathbf{G}_u\}_{u \in V})$ associated with ring R and is additionally parameterized by L. It consists of the following algorithms:

- **Secret Key Generation,** $\mathsf{Gen}(1^\lambda)$: It outputs secret key $\mathbf{sk}$ and zero test parameters ztpp.
- **Encoding,** $\mathsf{Encode}(\mathbf{sk}, a_1, \ldots, a_L, u \in V)$: In addition to secret key $\mathbf{sk}$, it takes as input $a_1, \ldots, a_L \in R$ and a node $u \in V$. If u is not the root node, it outputs an encoding $[a_1 | \cdots | a_L]_{\mathbf{u}}$. If u is indeed the root node, it outputs an encoding $\left[\sum_{i=1}^{L} a_i\right]_{\mathbf{u}}$.
- **Add,** $[a_1 | \cdots | a_L]_{\mathbf{u}} + [b_1 | \cdots | b_L]_{\mathbf{u}} = [a_1 + b_1 | \cdots | a_L + b_L]_{\mathbf{u}}$. Note that only elements corresponding to the same node in the tree can be added. Further, the elements in the vector are added component-wise.

- **Multiply:** Suppose $w = \mathbf{par}(u)$ and $w = \mathbf{par}(v)$.

$$[a_1|\cdots|a_L]_{\mathbf{u}} \circ [b_1|\cdots|b_L]_{\mathbf{v}} = \begin{cases} [a_1b_1|\cdots|a_Lb_L]_{\mathbf{w}} & \text{if } \mathbf{rt}(T) \neq w \\ \left[\sum\limits_{i=1}^{L} a_ib_i\right]_{\mathbf{w}} & \text{otherwise} \end{cases}$$

  The elements in the vectors are multiplied component-wise.
- **Zero Test,** $\mathsf{ZeroTest}(\mathsf{ztpp}, [a]_{\mathfrak{r}})$: On input zero test parameters ztpp and an encoding $[a]_{\mathfrak{r}}$ at level $\mathfrak{r}$, where $\mathfrak{r} = \mathbf{rt}(T)$, output 0 if and only if $a = 0$.

Remark 6. The degree of slotted encodings can be defined along the same lines as the degree of structured multilinear maps.

3.4 Evaluation of Polynomials on Slotted Encodings

We consider the homomorphic evaluation of (T, ϕ)-respecting polynomials on slotted encodings. We first define evaluation of (T, ϕ)-respecting monomials on slotted encodings and then using this notion define evaluation of $(T, \overrightarrow{\phi})$-respecting polynomials on slotted encodings.

$\mathsf{HomEval}\,(t, \mathsf{SMMap}, \{\mathsf{E}_{1,u}\}_{u \in V}, \ldots, \{\mathsf{E}_{n,u}\}_{u \in V})$: The input to this algorithm is (T, ϕ)-respecting monomial $t \in \mathbb{F}_{\mathbf{p}}[y_1, \ldots, y_n]$, slotted encoding scheme $\mathsf{SMMap} = (T = (V, E),\ \{\mathbf{G}_u\}_{u \in V})$ and slotted encodings $\mathsf{E}_{i,u}$, for every $i \in [n]$ and every $u \in V$, encoded under $\mathbf{G}_u$.

The evaluation proceeds recursively as follows: for every non leaf node $u \in V$, set $\widetilde{\mathsf{E}_u} = \mathsf{E}_{\phi(u),u}$. Consider the case when u is a non-leaf node and let v and w be the children of u. Compute encoding associated with node u as $\widetilde{\mathsf{E}_u} = \widetilde{\mathsf{E}_v} \circ \widetilde{\mathsf{E}_w}$. Let $\mathbf{rt}$ be the root of T. Output the encoding $\widetilde{\mathsf{E}_{\mathbf{rt}}}$ associated with $\mathbf{rt}$.

$\mathsf{HomEval}\,(p, \mathsf{SMMap}, \{\mathsf{E}_{1,u}\}_{u \in V}, \ldots, \{\mathsf{E}_{n,u}\}_{u \in V})$: The input to this algorithm is $(T, \overrightarrow{\phi})$-respecting polynomial $p \in \mathbb{F}_{\mathbf{p}}[y_1, \ldots, y_n]$, slotted encoding scheme $\mathsf{SMMap} = (T = (V, E),\ \{\mathbf{G}_u\}_{u \in V})$ and slotted encodings $\mathsf{E}_{i,u}$, for every $i \in [n]$ and every $u \in V$, encoded under $\mathbf{G}_u$.

Let $p = \sum_{i=1}^{n} c_i t_i$, for $c_i \in \mathbb{F}_{\mathbf{p}}$ and t_i is a (T, ϕ_i)-respecting monomial for every $i \in [n]$. The evaluation proceeds as follows: for every $i \in [n]$, execute $\widetilde{\mathsf{E}_{\mathbf{rt}}}^{(i)} \leftarrow \mathsf{HomEval}(t_i, \mathsf{SMMap}, \{\mathsf{E}_{1,u}\}_{u \in V}, \ldots, \{\mathsf{E}_{n,u}\}_{u \in V})$. Compute $\mathsf{E}_{\mathbf{rt}} = \sum\limits_{i=1}^{n} c_i \widetilde{\mathsf{E}_{\mathbf{rt}}}^{(i)}$. Output the encoding $\mathsf{E}_{\mathbf{rt}}$.

Remark 7. Based on the current implementation of multilinear maps, given an encoding of an element $a \in \mathbb{F}_{\mathbf{p}}$, we don't know how to securely obtain encoding of $c{\cdot}a$ for some scalar $c \in \mathbb{F}_{\mathbf{p}}$ of our choice. But instead, we can still obtain encoding of $c \cdot a$, when c is small (for instance, polynomial in security parameter). This can achieved by adding encoding of a, c number of times.

4 Projective Arithmetic FE from Slotted Encodings

We show how to construct projective arithmetic FE starting from the notion of slotted encodings defined in Sect. 3.3.

Consider a L-slotted encoding scheme SEnc, defined with respect to structured multilinear maps $\mathsf{SMMap} = (T = (V, E),\ \{\mathbf{G}_u\}_{u \in V})$ and is parameterized by L. We construct a multi-key secret key projective arithmetic functional encryption scheme PAFE for a function class $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ as follows. Here, $\mathcal{C}_\lambda$ consists of functions with input length λ and output length $\mathrm{poly}(\lambda)$.

$\underline{\mathsf{Setup}(1^\lambda)}$: On input security parameter λ,

- It executes the secret key generation algorithm of the slotted encoding scheme to obtain $\mathbf{sk} \leftarrow \mathsf{Gen}(1^\lambda)$.
- Sample values $\alpha_{i,u} \in \mathbb{F}_{\mathbf{p}}$ for every $i \in [\ell_{\mathsf{inp}}], u \in V$ at random. We define ℓ_{inp} later. Denote $\overrightarrow{\alpha} = (\alpha_{i,u})_{i \in [\ell_{\mathsf{inp}}], u \in V}$.
- Sample a random value $S \in \mathbb{F}_{\mathbf{p}}$.

It outputs $\mathsf{MSK} = (\mathbf{sk}, \overrightarrow{\alpha}, S)$.

$\underline{\mathsf{KeyGen}(\mathsf{MSK}, p)}$: It takes as input master secret key MSK and a T-respecting polynomial $p \in \mathbb{F}_{\mathbf{p}}[y_1, \ldots, y_{\ell_{\mathsf{inp}}}]$ associated with an arithmetic circuit C, where T is the same tree associated with the structured multilinear maps. Since p is T-respecting, we have the following: There exists $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_K)$ with $\phi_i : [V] \rightarrow [\ell_{\mathsf{inp}}]$ such that:

- $p = \sum_{j=1}^{K} c_i t_i$, where $c_i \in \mathbb{F}_{\mathbf{p}}$.
- t_i is a (T, ϕ_i)-respecting monomial in ℓ_{inp} variables.

Let δ_i be obtained by first assigning $\alpha_{\phi_i(u),u}$ to every leaf node u and then evaluating T[13]. That is, δ_i is the value obtained at the root of T. Assign $\varDelta = \sum_{i=1}^{K} c_i \cdot \delta_i$.

Let $\mathbf{rt}$ be the root of T and let $\mathfrak{u}$ be its left child and $\mathfrak{v}$ be its right child. Compute $\mathsf{E}^C = \mathsf{Encode}\left(\mathbf{sk}, (0, \varDelta \cdot S, p(0;0)), \mathfrak{u}\right)$ for every $i \in [n]$. Output $sk_C = (C, \mathsf{E}^C)$.

$\underline{\mathsf{Enc}(\mathsf{MSK}, x)}$: It takes as input master secret key MSK and input $x \in \{0,1\}^{\ell_x}$. Let $\mathsf{inp} = x$ and $\ell_{\mathsf{inp}} = |x|$.

It also samples an element $\gamma \in \mathbb{F}_{\mathbf{p}}$ at random. For every $i \in [\ell_{\mathsf{inp}}], u \in V$ and u is a leaf node, encode the tuple $(\mathsf{inp}_i, \gamma \cdot \alpha_{i,u}, 0)$ with inp_i denoting the i^{th} bit of inp, as follows: $\mathsf{E}_{i,u}^{\mathsf{inp}} = \mathsf{Encode}\left(\mathsf{MSK}, (\mathsf{inp}_i, \gamma \cdot \alpha_{i,u}, 0), u\right)$. Also encode γ^D under group $\mathbf{G}_{\mathfrak{v}}$, where $\mathfrak{v}$ is the right child of $\mathbf{rt}$: $\mathsf{E}_\gamma = \mathsf{Encode}(\mathsf{MSK}, (0, \gamma^D \cdot S^{-1}, 0), \mathfrak{v})$. Recall that D is the degree of homogeneity of RP.

Output the ciphertext $\mathsf{CT} = \left((\mathsf{E}_{i,u})_{i \in [\mathsf{inp}], u \in V}, \mathsf{E}_\gamma\right)$.

$\underline{\mathsf{ProjectDec}(sk_C, \mathsf{CT})}$: It takes as input functional key sk_C and ciphertext CT. It parses sk_C as (C, E^C) and CT as $\left((\mathsf{E}_{i,u})_{i \in [\mathsf{inp}], u \in V}, \mathsf{E}_\gamma\right)$. It executes the following:

[13] Note that every non leaf node is treated as a multiplication gate.

- Compute $\mathsf{out}_1 = \mathsf{HomEval}(p, \mathsf{SMMap}, (\mathsf{E}_{i,u})_{i \in [\mathsf{inp}], u \in V})$.
- Compute $\mathsf{out}_2 = \mathsf{E}^C \circ \mathsf{E}_\gamma$.

Output the partial decrypted value $\iota = \mathsf{out}_1 - \mathsf{out}_2$.

$\underline{\mathsf{Recover}(c_1, \iota_1, \ldots, c_{\ell_f}, \iota_{\ell_f})}$: On input co-efficients $c_i \in \mathbb{F}_\mathbf{p}$, partial decrypted values ι_i, it first computes:

$$temp = c_1 \iota_1 + \cdots + c_{\ell_f} \iota_{\ell_f}$$

The addition carried out above corresponds to the addition associated with the slotted encodings scheme. Now, perform $\mathsf{ZeroTest}(\mathsf{ztpp}, temp)$ and output the result. Note that the output is either in $\{0, \ldots, B\}$ or its $\perp$.

(B, B')-Correctness. From the correctness of HomEval and slotted encodings, it follows that out_1 is an encoding of $(p(x), \gamma^D \cdot p(\{\alpha_{i,u}\}), 0)$. Further, out_2 is an encoding of $(0, \gamma^D \cdot p(\{\alpha_{i,u}\}), 0)$. Thus, the partial decrypted value $\mathsf{out}_1 - \mathsf{out}_2$ is an encoding of $(p(x), 0, 0)$.

With this observation, we remark that for many polynomials $p_1, \ldots, p_N$, the decryption of functional key of p_i on encryption of x yields as partial decrypted values, encodings of $(p_i(x), 0, 0)$. Thus, sum of all encodings of $(c_i \cdot p_i(x), 0, 0)$, where $c_i \in B'$ and $B' = \{0, \ldots, \text{poly}(\lambda)\}$, yields a successful zero test query if and only if $\sum_{i=1}^{N} c_i p_i(x) = 0$.

We remark that if ztpp just contains parameters to test whether a top level encoding is zero or not, then the above construction only supports $B = \{0\}$. If it additionally contains encoding of 1, then we can set $B = \text{poly}(\lambda)$.

Encryption Complexity: Multiplicative Overhead. We calculate the encryption complexity as follows.

$$|\mathsf{Enc}(\mathsf{MSK}, x)| = |x| \cdot (\text{Number of groups in } \mathsf{SMMap}) \cdot \text{poly}(\lambda)$$

Thus, the above scheme satisfies the multiplicative overhead property.

4.1 Proof of Security

Semi-Functional Algorithms: We describe the semi-functional encryption and the key generation algorithms. We start with the semi-functional key generation algorithm.

$\underline{\mathsf{sfKG}(\mathsf{MSK}, p, \theta)}$: Parse MSK as $(\mathbf{sk}, \overrightarrow{\alpha}, S)$. In addition, it takes as input a (T, ϕ)-respecting polynomial p and value θ to be hardwired in the third slot. Let $p = \sum_{j=1}^{K} c_j t_j$, where t_j is a (T, ϕ_j)-respecting monomial in ℓ_{inp} variables. Let δ_j be obtained by first assigning $\alpha_{\phi_j(u),u}$ to every leaf node u and then evaluating T. That is, δ_j is the value obtained at the root of T. Assign $\Delta = \sum_{j=1}^{K} c_{i,j} \cdot \delta_j$.

Let $\mathbf{rt}$ be the root of T and let $\mathfrak{u}$ be its left child and $\mathfrak{v}$ be its right child. Compute $\mathsf{E}^p = \mathsf{Encode}(\mathbf{sk}, (0, \Delta \cdot S, p(0; 0) - \theta), \mathfrak{u})$ for every $i \in [n]$. Output $sk_C = (p, \mathsf{E}^p)$.

We now describe the semi-functional encryption algorithm.

$\underline{\mathsf{sfEnc}(\mathsf{MSK}, 1^{\ell_{\mathsf{inp}}})}$: Parse MSK as $(\mathbf{sk}, \overrightarrow{\alpha})$. It samples an element $\gamma \in \mathbb{F}_{\mathbf{p}}$ at random.

For every $i \in [\ell_{\mathsf{inp}}], u \in V$ and u is a leaf node, encode the tuple $(0, \gamma \cdot \alpha_{i,u}, 0)$ as follows: $\mathsf{E}_{i,u}^{\mathsf{inp}} = \mathsf{Encode}\left(\mathsf{MSK}, (0, \gamma \cdot \alpha_{i,u}, 0), u\right)$. Also encode γ^D under group $\mathbf{G}_{\mathfrak{v}}$, where $\mathfrak{v}$ is the right child of $\mathbf{rt}$: $\mathsf{E}_{\gamma} = \mathsf{Encode}(\mathsf{MSK}, (0, \gamma^D, 1), \mathfrak{v})$. Recall that D is the degree of homogeneity of RP.

Output the ciphertext $\mathsf{CT} = \left((\mathsf{E}_{i,u})_{i \in [\mathsf{inp}], u \in V}, \mathsf{E}_{\gamma}\right)$.

We now prove the indistinguishability of semi-functional ciphertexts and indistinguishability of functional keys properties. Before that we state the assumptions on the slotted encodings upon which we prove the security of our scheme.

Assumptions. We define the following two assumptions.

__Assumption #1:__ For all (i) *inputs* $\mathbf{x} = (x_1, \ldots, x_\mu) \in \{0,1\}^{\mu \cdot \ell_x}$, (ii) *polynomials* $p \in \mathbb{F}_{\mathbf{p}}[y_1, \ldots, y_n], \mathbf{q} = (q_1, \ldots, q_N) \in \mathbb{F}_{\mathbf{p}}[y_1, \ldots, y_n]^N$ *be* (T, ϕ)*-respecting polynomials,* (iii) *subset* $I \subseteq [n]$ *and finally,* (iv) *values* $\theta \in \mathbb{F}_{\mathbf{p}}, \Theta = (\theta_i)_{i \in I} \in \mathbb{F}_{\mathbf{p}}^{|I|}$ *and for every sufficiently large* $\lambda \in \mathbb{N}$, *the following holds:*

$$\{\ \mathsf{KeyGen}(\mathsf{MSK}, p),\ aux[\mathbf{x}, \mathbf{q}, I, \Theta]\ \}\ \cong_c\ \{\ \mathsf{sfKG}(\mathsf{MSK}, p, \theta),\ aux[\mathbf{x}, \mathbf{q}, I, \Theta]\}$$

- $\mathsf{MSK} \leftarrow \mathsf{Setup}(1^\lambda)$
- $aux[\mathbf{x}, \mathbf{q}, I, \Theta] = (\mathsf{CT}_1, \ldots, \mathsf{CT}_\mu, sk_1, \ldots, sk_N)$ *consists of two components:*
  1. *For every* $i \in [n]$, *compute* $\mathsf{CT}_i \leftarrow \mathsf{Enc}(\mathsf{MSK}, x_i)$.
  2. *For every* $i \in [N]$ and $i \in I$, *compute* $sk_i \leftarrow \mathsf{sfKG}(\mathsf{MSK}, q_i, \theta_i)$. *Else if* $i \notin I$, *compute* $sk_i \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, q_i)$.

__Assumption #2:__ For all (i) *inputs* $x^* \in \{0,1\}^{\ell_x}, \mathbf{x} = (x_1, \ldots, x_\mu) \in \{0,1\}^{\mu \cdot \ell_x}$, (ii) *polynomials* $\mathbf{q} = (q_1, \ldots, q_N) \in \mathbb{F}_{\mathbf{p}}[y_1, \ldots, y_n]^N$ *be* (T, ϕ)*-respecting polynomials and finally,* (iii) *values* $\Theta = (\theta_i)_{i \in [N]}$ *and for every sufficiently large* $\lambda \in \mathbb{N}$, *the following holds:*

$$\left\{\ \mathsf{sfEnc}(\mathsf{MSK}, 1^{\ell_{\mathsf{inp}}}),\ aux[\mathbf{x}, \mathbf{q}, \Theta]\ \right\}\ \cong_c\ \{\ \mathsf{Enc}(\mathsf{MSK}, x^*),\ aux[\mathbf{x}, \mathbf{q}, \Theta]\}$$

- $\mathsf{MSK} \leftarrow \mathsf{Setup}(1^\lambda)$
- $aux[\mathbf{x}, \mathbf{q}, \Theta] = (\mathsf{CT}_1, \ldots, \mathsf{CT}_\mu, sk_1, \ldots, sk_N)$ *is computed in the following way:*
  1. *For every* $i \in [n]$, *compute* $\mathsf{CT}_i \leftarrow \mathsf{Enc}(\mathsf{MSK}, x_i)$.
  2. *For every* $i \in [N]$ $\theta_i = q_i(x^*)$.
  3. *For every* $i \in [N]$, *compute* $sk_i \leftarrow \mathsf{sfKG}(\mathsf{MSK}, q_i, \theta_i)$.

The following two theorems directly follow from the above two assumptions.

Theorem 4. *The scheme* PAFE *satisfies indistinguishability of semi-functional keys under Assumption #1.*

Theorem 5. *The scheme* PAFE *satisfies indistinguishability of semi-functional ciphertexts under Assumption #2.*

From the above two theorems, we have the following theorem.

Theorem 6. *The* PAFE *satisfies semi-functional security under Assumptions #1 and #2.*

References

[AAB+15] Agrawal, S., Agrawal, S., Badrinarayanan, S., Kumarasubramanian, A., Prabhakaran, M., Sahai, A.: On the practical security of inner product functional encryption. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 777–798. Springer, Heidelberg (2015). doi:10.1007/978-3-662-46447-2_35

[AB15] Applebaum, B., Brakerski, Z.: Obfuscating circuits via composite-order graded encoding. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015. LNCS, vol. 9015, pp. 528–556. Springer, Heidelberg (2015). doi:10.1007/978-3-662-46497-7_21

[ABCP15] Abdalla, M., Bourse, F., Caro, A., Pointcheval, D.: Simple functional encryption schemes for inner products. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 733–751. Springer, Heidelberg (2015). doi:10.1007/978-3-662-46447-2_33

[ABCP16] Abdalla, M., Bourse, F., De Caro, A., Pointcheval, D.: Better security for functional encryption for inner product evaluations. IACR Cryptology ePrint Archive 2016:11 (2016)

[ABSV15] Ananth, P., Brakerski, Z., Segev, G., Vaikuntanathan, V.: From selective to adaptive security in functional encryption. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9216, pp. 657–677. Springer, Heidelberg (2015). doi:10.1007/978-3-662-48000-7_32

[AIK06] Applebaum, B., Ishai, Y., Kushilevitz, E.: Computationally private randomizing polynomials and their applications. Comput. Compl. **15**(2), 115–162 (2006)

[AJ15] Ananth, P., Jain, A.: Indistinguishability obfuscation from compact functional encryption. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9215, pp. 308–326. Springer, Heidelberg (2015). doi:10.1007/978-3-662-47989-6_15

[AJS15] Ananth, P., Jain, A., Sahai, A.: Achieving compactness generically: Indistinguishability obfuscation from non-compact functional encryption. IACR Cryptology ePrint Archive 2015:730 (2015)

[AL16] Applebaum, B., Lovett, S.: Algebraic attacks against random local functions and their countermeasures. In: STOC, pp. 1087–1100 (2016)

[AS16] Ananth, P., Sahai, A.: Projective arithmetic functional encryption and indistinguishability obfuscation from degree-5 multilinear maps. Cryptology ePrint Archive, Report 2016/1097 (2016). http://eprint.iacr.org/2016/1097

[BCF16] Elisabetta, C., Baltico, Z., Catalano, D., Fiore, D.: Practical functional encryption for bilinear forms. Cryptology ePrint Archive, Report 2016/1104 (2016). http://eprint.iacr.org/2016/1104

[BGG+14] Boneh, D., Gentry, C., Gorbunov, S., Halevi, S., Nikolaenko, V., Segev, G., Vaikuntanathan, V., Vinayagamurthy, D.: Fully key-homomorphic encryption, arithmetic circuit ABE and Compact garbled circuits. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 533–556. Springer, Heidelberg (2014). doi:10.1007/978-3-642-55220-5_30

[BGI+01] Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S., Yang, K.: On the (im)possibility of obfuscating programs. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 1–18. Springer, Heidelberg (2001). doi:10.1007/3-540-44647-8_1

[BGJ+16] Bitansky, N., Goldwasser, S., Jain, A., Paneth, O., Vaikuntanathan, V., Waters, B.: Time-lock puzzles from randomized encodings. In: ITCS 2016

[BJK15] Bishop, A., Jain, A., Kowalczyk, L.: Function-hiding inner product encryption. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015. LNCS, vol. 9452, pp. 470–491. Springer, Heidelberg (2015). doi:10.1007/978-3-662-48797-6_20

[BLP16] Bitansky, N., Lin, H., Paneth, O.: On removing graded encodings from functional encryption. Cryptology ePrint Archive, Report 2016/962, 2016 http://eprint.iacr.org/2016/962

[BNPW16] Bitansky, N., Nishimaki, R., Passelègue, A., Wichs, D.: From cryptomania to obfustopia through secret-key functional encryption. Cryptology ePrint Archive, Report 2016/558 (2016). http://eprint.iacr.org/2016/558

[BNS13] Boneh, D., Nikolaenko, V., Segev, G.: Attribute-based encryption for arithmetic circuits. IACR Cryptology ePrint Archive 2013:669 (2013)

[BP15] Bitansky, N., Paneth, O.: ZAPs and non-interactive witness indistinguishability from indistinguishability obfuscation. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015. LNCS, vol. 9015, pp. 401–427. Springer, Heidelberg (2015). doi:10.1007/978-3-662-46497-7_16

[BPR15] Bitansky, N., Paneth, O., Rosen, A.: On the cryptographic hardness of finding a Nash equilibrium. In: FOCS (2015)

[BS15] Brakerski, Z., Segev, G.: Function-private functional encryption in the private-key setting. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015. LNCS, vol. 9015, pp. 306–324. Springer, Heidelberg (2015). doi:10.1007/978-3-662-46497-7_12

[BSW07] Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE (S&P 2007), pp. 321–334 (2007)

[BV15] Bitansky, N., Vaikuntanathan, V.: Indistinguishability obfuscation from functional encryption. In: FOCS, IEEE (2015)

[CGH+15] Coron, J.-S., Gentry, C., Halevi, S., Lepoint, T., Maji, H.K., Miles, E., Raykova, M., Sahai, A., Tibouchi, M.: Zeroizing without low-level zeroes: new MMAP attacks and their limitations. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9215, pp. 247–266. Springer, Heidelberg (2015). doi:10.1007/978-3-662-47989-6_12

[CHL+15] Cheon, J.H., Han, K., Lee, C., Ryu, H., Stehlé, D.: Cryptanalysis of the multilinear map over the integers. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 3–12. Springer, Heidelberg (2015). doi:10.1007/978-3-662-46800-5_1

[CHN+16] Cohen, A., Holmgren, J., Nishimaki, R., Vaikuntanathan, V., Wichs, D.: Watermarking cryptographic capabilities. In: STOC (2016)

[CIJ+13] Caro, A., Iovino, V., Jain, A., O'Neill, A., Paneth, O., Persiano, G.: On the achievability of simulation-based security for functional encryption. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8043, pp. 519–535. Springer, Heidelberg (2013). doi:10.1007/978-3-642-40084-1_29

[CLT13] Coron, J.-S., Lepoint, T., Tibouchi, M.: Practical multilinear maps over the integers. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 476–493. Springer, Heidelberg (2013). doi:10.1007/978-3-642-40041-4_26

[CS02] Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 45–64. Springer, Heidelberg (2002). doi:10.1007/3-540-46035-7_4

[DDM16] Datta, P., Dutta, R., Mukhopadhyay, S.: Functional encryption for inner product with full function privacy. In: Cheng, C.-M., Chung, K.-M., Persiano, G., Yang, B.-Y. (eds.) PKC 2016. LNCS, vol. 9614, pp. 164–195. Springer, Heidelberg (2016). doi:10.1007/978-3-662-49384-7_7

[Gay16] Gay, R.: Functional encryption for quadratic functions, and applications to predicate encryption. Cryptology ePrint Archive, Report 2016/1106 (2016). http://eprint.iacr.org/2016/1106

[GGG+14] Goldwasser, S., Gordon, S.D., Goyal, V., Jain, A., Katz, J., Liu, F.-H., Sahai, A., Shi, E., Zhou, H.-S.: Multi-input functional encryption. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 578–602. Springer, Heidelberg (2014). doi:10.1007/978-3-642-55220-5_32

[GGH13a] Garg, S., Gentry, C., Halevi, S.: Candidate multilinear maps from ideal lattices. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 1–17. Springer, Heidelberg (2013). doi:10.1007/978-3-642-38348-9_1

[GGH+13b] Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: FOCS, pp. 40–49 (2013)

[GGH15] Gentry, C., Gorbunov, S., Halevi, S.: Graph-induced multilinear maps from lattices. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015. LNCS, vol. 9015, pp. 498–527. Springer, Heidelberg (2015). doi:10.1007/978-3-662-46497-7_20

[GGHR14] Garg, S., Gentry, C., Halevi, S., Raykova, M.: Two-round secure MPC from indistinguishability obfuscation. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 74–94. Springer, Heidelberg (2014). doi:10.1007/978-3-642-54242-8_4

[GGHZ14] Garg, S., Gentry, C., Halevi, S., Zhandry, M.: Fully secure attribute based encryption from multilinear maps. IACR Cryptology ePrint Archive 2014:622 (2014)

[GLSW15] Gentry, C., Lewko, A.B., Sahai, A., Waters, B.: Indistinguishability obfuscation from the multilinear subgroup elimination assumption. In: FOCS, pp. 151–170 (2015)

[GMM+16] Garg, S., Miles, E., Mukherjee, P., Sahai, A., Srinivasan, A., Zhandry, M.: Secure obfuscation in a weak multilinear map model. Cryptology ePrint Archive, Report 2016/817 (2016). http://eprint.iacr.org/2016/817

[GMR89] Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM J. Comput. **18**(1), 186–208 (1989)

[GPSW06] Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: ACM CCS (2006)

[GS08] Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432. Springer, Heidelberg (2008). doi:10.1007/978-3-540-78967-3_24

[HSW14] Hohenberger, S., Sahai, A., Waters, B.: Replacing a random oracle: full domain hash from indistinguishability obfuscation. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 201–220. Springer, Heidelberg (2014). doi:10.1007/978-3-642-55220-5_12

[IK00] Ishai, Y., Kushilevitz, E.: Randomizing polynomials: a new representation with applications to round-efficient secure computation. In: FOCS, pp. 294–304 (2000)

[KSW08] Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 146–162. Springer, Heidelberg (2008). doi:10.1007/978-3-540-78967-3_9

[Lin16] Lin, H.: Indistinguishability obfuscation from constant-degree graded encoding schemes. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9665, pp. 28–57. Springer, Heidelberg (2016). doi:10.1007/978-3-662-49890-3_2

[LOS+10] Lewko, A., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully secure functional encryption: attribute-based encryption and (hierarchical) inner product encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 62–91. Springer, Heidelberg (2010). doi:10.1007/978-3-642-13190-5_4

[LV16] Lin, H., Vaikuntanathan, V.: Indistinguishability obfuscation from DDH-like assumptions on constant-degree graded encodings. In FOCS (2016)

[MSZ16] Miles, E., Sahai, A., Zhandry, M.: Annihilation attacks for multilinear maps: cryptanalysis of indistinguishability obfuscation over GGH13. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9815, pp. 629–658. Springer, Heidelberg (2016). doi:10.1007/978-3-662-53008-5_22

[OT08] Okamoto, T., Takashima, K.: Homomorphic encryption and signatures from vector decomposition. In: Galbraith, S.D., Paterson, K.G. (eds.) Pairing 2008. LNCS, vol. 5209, pp. 57–74. Springer, Heidelberg (2008). doi:10.1007/978-3-540-85538-5_4

[OT09] Okamoto, T., Takashima, K.: Hierarchical predicate encryption for inner-products. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 214–231. Springer, Heidelberg (2009). doi:10.1007/978-3-642-10366-7_13

[OW14] O'Donnell, R., Witmer, D.: Goldreich's PRG: evidence for near-optimal polynomial stretch. In: CCC, pp. 1–12 (2014)

[SW05] Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005). doi:10.1007/11426639_27

[SW08] Sahai, A., Waters, B.: Slides on functional encryption. Powerpoint presentation (2008)

[SW14] Sahai, A., Waters, B.: How to use indistinguishability obfuscation: deniable encryption, and more. In STOC, pp. 475–484 (2014)

[Wat09] Waters, B.: Dual system encryption: realizing fully secure IBE and HIBE under simple assumptions. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 619–636. Springer, Heidelberg (2009). doi:10.1007/978-3-642-03356-8_36

[Zim15] Zimmerman, J.: How to obfuscate programs directly. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 439–467. Springer, Heidelberg (2015). doi:10.1007/978-3-662-46803-6_15